Robert Meersman
Zahir Tari et al. (Eds.)

On the Move to Meaningful Internet Systems 2006: CoopIS, DOA, GADA, and ODBASE

LNCS 4275

OTM Confederated International Conferences
CoopIS, DOA, GADA, and ODBASE 2006
Montpellier, France, October/November 2006
Proceedings, Part I

Lecture Notes in Computer Science 4275

Commenced Publication in 1973
Founding and Former Series Editors:
Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

Robert Meersman Zahir Tari et al. (Eds.)

On the Move to Meaningful Internet Systems 2006: CoopIS, DOA, GADA, and ODBASE

OTM Confederated International Conferences
CoopIS, DOA, GADA, and ODBASE 2006
Montpellier, France, October 29 – November 3, 2006
Proceedings, Part I

Springer

Volume Editors

Robert Meersman
Vrije Universiteit Brussel (VUB), STARLab
Bldg G/10, Pleinlaan 2, 1050 Brussels, Belgium
E-mail: meersman@vub.ac.be

Zahir Tari
RMIT University, School of Computer Science and Information Technology
Bld 10.10, 376-392 Swanston Street, VIC 3001, Melbourne, Australia
E-mail: zahirt@cs.rmit.edu.au

Library of Congress Control Number: 2006934986

CR Subject Classification (1998): H.2, H.3, H.4, C.2, H.5, I.2, D.2.12, K.4

LNCS Sublibrary: SL 3 – Information Systems and Application, incl. Internet/Web and HCI

ISSN 0302-9743
ISBN-10 3-540-48287-3 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-48287-1 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 11914853 06/3142 5 4 3 2 1 0

Volume Editors

Robert Meersman
Zahir Tari

CoopIS

Mike Papazoglou
Louiqa Raschid
Rainer Ruggaber

DOA

Judith Bishop
Kurt Geihs

GADA

Pilar Herrero
María S. Pérez
Domenico Talia
Albert Zomaya

ODBASE

Maurizio Lenzerini
Erich Neuhold
V.S. Subrahmanian

OTM 2006 General Co-chairs' Message

Dear OnTheMove Participant or Reader of these Proceedings,

The General Chairs of OnTheMove 2006, Montpellier, France, are happy to observe that the conference series that was started in Irvine, California in 2002 and subsequently held in Catania, Sicily in 2003 and in Cyprus in 2004 and 2005 clearly continues to attract a growing representative selection of today's worldwide research on the scientific concepts underlying distributed, heterogeneous and autonomous yet meaningfully collaborative computing, with the Internet and the WWW as its prime epitomes.

Indeed, as such large, complex and networked intelligent information systems become the focus and norm for computing, it is clear that there is an acute and increasing need to address and discuss in an integrated forum the implied software and system issues as well as methodological, theoretical and application issues. As we all know, e-mail, the Internet, and even video conferences are not sufficient for effective and efficient scientific exchange. This is why the OnTheMove (OTM) Federated Conferences series has been created to cover the increasingly wide yet closely connected range of fundamental technologies such as data and Web semantics, distributed objects, Web services, databases, information systems, workflow, cooperation, ubiquity, interoperability, mobility, grid and high-performance. OnTheMove aspires to be a primary scientific meeting place where all aspects of the development of Internet- and Intranet-based systems in organizations and for e-business are discussed in a scientifically motivated way. This fifth 2006 edition of the OTM Federated Conferences event therefore again provided an opportunity for researchers and practitioners to understand and publish these developments within their individual as well as within their broader contexts.

The backbone of OTM was originally formed by the co-location of three related, complementary and successful main conference series: DOA (Distributed Objects and Applications, since 1999), covering the relevant infrastructure-enabling technologies, ODBASE (Ontologies, DataBases and Applications of SEmantics, since 2002) covering Web semantics, XML databases and ontologies, CoopIS (Cooperative Information Systems, since 1993) covering the application of these technologies in an enterprise context through, for example, workflow systems and knowledge management. For the 2006 edition, these were strengthened by a fourth conference, GADA (Grid computing, high-performAnce and Distributed Applications, a successful workshop at OTM since 2004), covering the large-scale integration of heterogeneous computing systems and data resources with the aim of providing a global computing space. Each of these four conferences encourages researchers to treat their respective topics within a framework that incorporates jointly (a) theory , (b) conceptual design and development, and (c) applications, in particular case studies and industrial solutions.

Following and expanding the model created in 2003, we again solicited and selected quality workshop proposals to complement the more "archival" nature of the main conferences with research results in a number of selected and more "avant garde" areas related to the general topic of distributed computing. For instance, the so-called Semantic Web has given rise to several novel research areas combining linguistics, information systems technology, and artificial intelligence, such as the modeling of (legal) regulatory systems and the ubiquitous nature of their usage. We were glad to see that several earlier successful workshops (notably WOSE, MIOS-INTEROP, AweSOMe, CAMS, SWWS, SeBGIS, ORM) re-appeared in 2006 with a second, third or sometimes fourth edition, and that not less than seven new workshops could be hosted and successfully organized by their respective proposers: IS (International Workshop on Information Security), COMINF (International Workshop on Community Informatics), KSinBIT (International Workshop on Knowledge Systems in Bioinformatics), MONET (International Workshop on MObile and NEtworking Technologies for social applications), OnToContent (Ontology content and evaluation in Enterprise), PerSys (International Workshop on Pervasive Systems), and RDDS (International Workshop on Reliability in Decentralized Distributed Systems). We know that as before, their audiences will mutually productively mingle with those of the main conferences, as is already visible from the overlap in authors! The OTM organizers are especially grateful for the leadership and competence of Pilar Herrero in managing this complex process into a success for the second year in a row.

A special mention for 2006 is again due for the third and enlarged edition of the highly attractive OnTheMove Academy (formerly called Doctoral Consortium Workshop). Its 2006 Chairs, Antonia Albani, Gábor Nagypál and Johannes Maria Zaha, three young and active researchers, further refined the original set-up and interactive formula to bring PhD students together: they call them to submit their research proposals for selection; the resulting submissions and their approaches are presented by the students in front of a wider audience at the conference, where they are then independently and extensively analyzed and discussed in public by a panel of senior professors. This year these were Johann Eder, Maria Orlowska, and of course Jan Dietz, the Dean of the OnTheMove Academy, who provided guidance, support and help for the team. The successful students are also awarded free access to all other parts of the OTM program, and only pay a minimal fee for the Doctoral Symposium itself (in fact their attendance is largely sponsored by the other participants!). The OTM organizers expect to further expand the OnTheMove Academy in future editions of the conferences and so draw an audience of young researchers into the OTM forum.

All four main conferences and the associated workshops share the distributed aspects of modern computing systems, and the resulting application-pull created by the Internet and the so-called Semantic Web. For DOA 2006, the primary emphasis was on the distributed object infrastructure; for ODBASE 2006, it became the knowledge bases and methods required for enabling the use of formal semantics; for CoopIS 2006, the topic was the interaction of such

technologies and methods with management issues, such as occur in networked organizations, and for GADA 2006, the topic was the scalable integration of heterogeneous computing systems and data resources with the aim of providing a global computing space. These subject areas naturally overlap and many submissions in fact also treat an envisaged mutual impact among them. As for the earlier editions, the organizers wanted to stimulate this cross-pollination by a shared program of famous keynote speakers: this year we were proud to announce Roberto Cencioni (European Commission), Alois Ferscha (Johannes Kepler Universität), Daniel S. Katz (Louisiana State University and Jet Propulsion Laboratory), Frank Leymann (University of Stuttgart), and Marie-Christine Rousset (University of Grenoble)! We also encouraged multiple event attendance by providing all authors, also those of workshop papers, with free access or discounts to one other conference or workshop of their choice.

We received a total of 361 submissions for the four main conferences and an impressive 493 (compared to the 268 in 2005 and 170 in 2004!) submissions for the workshops. Not only may we indeed again claim success in attracting an increasingly representative volume of scientific papers, but such a harvest of course allows the Program Committees to compose a higher quality cross-section of current research in the areas covered by OTM. In fact, in spite of the larger number of submissions, the Program Chairs of each of the three main conferences decided to accept only approximately the same number of papers for presentation and publication as in 2003, 2004 and 2005 (i.e., average one paper out of four submitted, not counting posters). For the workshops, the acceptance rate varies but was much stricter than before, about one in two to three, to less than one quarter for the IS (Information Security) international workshop. Also for this reason, we separated the proceedings into two books with their own titles, with the main proceedings in two volumes, and we are grateful to Springer for their suggestions and collaboration in producing these books and CDROMs. The reviewing process by the respective Program Committees as usual was performed very professionally and each paper in the main conferences was reviewed by at least three referees, with arbitrated e-mail discussions in the case of strongly diverging evaluations. It may be worthwhile to emphasize that it is an explicit OnTheMove policy that all conference Program Committees and Chairs make their selections completely autonomously from the OTM organization itself. Continuing a costly but nice tradition, the OnTheMove Federated Event organizers decided again to make all proceedings available to all participants of conferences and workshops, independently of one's registration to a specific conference or workshop. Each participant also received a CDROM with the full combined proceedings (conferences + workshops).

The General Chairs are once more especially grateful to all the many people directly or indirectly involved in the setup of these federated conferences who contributed to making it a success. Few people realize what a large number of people have to be involved, and what a huge amount of work, and sometimes risk, the organization of an event like OTM entails. Apart from the persons in the roles mentioned above, we therefore in particular wish to thank our 12 main conference

PC Co-chairs (GADA 2006: Pilar Herrero, María S. Pérez, Domenico Talia, Albert Zomaya; DOA 2006: Judith Bishop, Kurt Geihs; ODBASE 2006: Maurizio Lenzerini, Erich Neuhold, V.S. Subrahmanian; CoopIS 2006: Mike Papazoglou, Louiqa Raschid, Rainer Ruggaber) and our 36 workshop PC Co-chairs (Antonia Albani, George Buchanan, Roy Campbell, Werner Ceusters, Elizabeth Chang, Ernesto Damiani, Jan L.G. Dietz, Pascal Felber, Fernando Ferri, Mario Freire, Daniel Grosu, Michael Gurstein, Maja Hadzic, Pilar Herrero, Terry Halpin, Annika Hinze, Skevos Evripidou, Mustafa Jarrar, Arek Kasprzyk, Gonzalo Méndez, Aldo de Moor, Bart De Moor, Yves Moreau, Claude Ostyn, Andreas Persidis, Maurizio Rafanelli, Marta Sabou, Vitor Santos, Simao Melo de Sousa, Katia Sycara, Arianna D'Ulizia, Eiko Yoneki, Esteban Zimányi).

All, together with their many PC members, did a superb and professional job in selecting the best papers from the large harvest of submissions.

We also heartily thank Zohra Bellahsene of LIRMM in Montpellier for the considerable efforts in arranging the venue at their campus and coordinating the substantial and varied local facilities needed for a multi-conference event such as ours. And we must all also be grateful to Mohand-Said Hacid of the University of Lyon for researching and securing the sponsoring arrangements, to Gonzalo Méndez, our excellent Publicity Chair, to our extremely competent and experienced Conference Secretariat and technical support staff Daniel Meersman, Ana-Cecilia Martinez Barbosa, and Jan Demey, and last but not least to our hyperactive Publications Chair and loyal collaborator of many years, Kwong Yuen Lai, this year bravely assisted by Peter Dimopoulos.

The General Chairs gratefully acknowledge the academic freedom, logistic support and facilities they enjoy from their respective institutions, Vrije Universiteit Brussel (VUB) and RMIT University, Melbourne, without which such an enterprise would not be feasible.

We do hope that the results of this federated scientific enterprise contribute to your research and your place in the scientific network... We look forward to seeing you again at next year's edition!

August 2006

Robert Meersman, Vrije Universiteit Brussel, Belgium
Zahir Tari, RMIT University, Australia
(General Co-chairs, OnTheMove 2006)

Organization Committee

The OTM (On The Move) 2006 Federated Conferences, which involve CoopIS (Cooperative Information Systems), DOA (distributed Objects and Applications), GADA (Grid computing, high-performAnce and Distributed Applications), and ODBASE (Ontologies, Databases and Applications of Semantics), are proudly supported by CNRS (Centre National de la Researche Scientifique, France), the City of Montpellier (France), Ecole Polytechnique Universitaire de Montepellier, Université de Montpellier II (UM2), Laboratoire d'Informatique de Robotique et de Microélectronique de Montpellier (LIRMM), RMIT University (School of Computer Science and Information Technology), and Vrije Universiteit Brussel (Department of Computer Science).

Executive Committee

OTM 2006 General Co-chairs:	Robert Meersman (Vrije Universiteit Brussel, Belgium) and Zahir Tari (RMIT University, Australia).
GADA 2006 PC Co-chairs:	Pilar Herrero (Universidad Politécnica de Madrid, Spain), María S. Pérez (Universidad Politécnica de Madrid, Spain), Domenico Talia (Universitá della Calabria, Italy), and Albert Zomaya (The University of Sydney, Australia).
CoopIS 2006 PC Co-chairs:	Mike Papazoglou (Tilburg University, Netherlands), Louiqa Raschid (University of Maryland, USA), and Rainer Ruggaber (SAP Research Center, Germany).
DOA 2006 PC Co-chairs:	Judith Bishop (University of Pretoria, South Africa) and Kurt Geihs (University of Kassel, Germany).
ODBASE 2006 PC Co-chairs:	Maurizio Lenzerini (Universitá di Roma "La Sapienza," Italy), Erich Neuhold (Darmstadt University of Technology, Germany), and V.S. Subrahmanian (University of Maryland College Park, USA).
Publication Co-chairs:	Kwong Yuen Lai (RMIT University, Australia) and Peter Dimopoulos (RMIT University, Australia).
Organizing Chair:	Zohra Bellahsene (LIRMM CNRS/University of Montpellier II, France).
Publicity Chair:	Mohand-Said Hacid (Université Claude Bernard Lyon I, France).

Secretariat: Ana-Cecilia Martinez Barbosa, Jan Demey, and Daniel Meersman.

CoopIS 2006 Program Committee

Marco Aiello
Alistair Barros
Boualem Benatallah
Salima Benbernou
Arne Berre
Elisa Bertino
Klemens Böhm
Alex Borgida
Luc Bouganim
Stephane Bressan
Laura Bright
Fabio Casati
Mariano Cilia
Vincenzo D'Andrea
Umesh Dayal
Alex Delis
Marlon Dumas
Schahram Dustdar
Johann Eder
Klaus Fischer
Avigdor Gal
Fausto Giunchiglia
Paul Grefen
Mohand-Said Hacid
Manfred Hauswirth
Willem-Jan van den Heuvel
Martin Hepp
Carsten Holtmann
Nenad Ivezic
Paul Johannesson
Manolis Koubarakis
Bernd Krämer
Winfried Lamersdorf
Steven Laufmann
Qing Li
Tiziana Margaria
Marta Mattoso
Massimo Mecella
Nikolay Mehandjiev
Brahim Medjahed
Michele Missikoff
Michael zur Muehlen
Jörg Müller
David Munro
Wolfgang Nejdl
Cesare Pautasso
Mourad Ouzzani
Manfred Reichert
Stefanie Rinderle
Uwe Riss
Timos Sellis
Anthony Tomasic
Farouk Toumani
Patrick Valduriez
Wil van der Aalst
Maria Esther Vidal
Mathias Weske
Jian Yang
Vladimir Zadorozhny

GADA 2006 Program Committee

Akshai Aggarwal
Alan Sussman
Alastair Hampshire
Alberto Sanchez
Alvaro A.A. Fernandes
Angelos Bilas
Antonio Garcia Dopico
Artur Andrzejak
Azzedine Boukerche
Beniamino Di Martino
Bhanu Prasad
Bing Bing Zhou

Carmela Comito
Carole Goble
Costin Badica
Dana Petcu
Daniel S. Katz
David Walker
Domenico Laforenza
Eduardo Huedo
Elghazali Talbi
Enrique Soler
Fatos Xhafa
Felix Garcia
Francisco Luna
Franciszek Seredynski
Gregorio Martinez
Hamid Sarbazi-Azad
Heinz Stockinger
Ignacio M. Llorente
Jack Dongarra
Jan Humble
Jemal Abawajy
Jesús Carretero
Jinjun Chen
Jon Maclaren
Jose L. Bosque
Jose M. Peña
Juan A. Botá Blaya
Kostas Karasavvas
Kurt Stockinger
Laurence T. Yang
Manish Parashar
Manuel Salvadores
Marcin Paprzycki
Maía Eugenia de Pool
Maria Ganzha
Mario Cannataro
Marios Dikaiakos
Mark Baker
Mirela Notare
Mohamed Ould-Khaoua
Neil P. Chue Hong
Omer F. Rana
Panayiotis Periorellis
Pascal Bouvry
Rainer Unland
Rajkumar Buyya
Reagan Moore
Rizos Sakellariou
Rosa M. Badia
Ruben S. Montero
Santi Caballé Llobet
Sattar B. Sadkhan Almaliky
Savitri Bevinakoppa
Stefan Egglestone
Thierry Priol
Toni Cortes
Valdimir Getov
Víctor Robles

ODBASE 2006 Program Committee

Sibel Adali
Maristella Agosti
Bill Andersen
Juergen Angele
Franz Baader
Sonia Bergamaschi
Alex Borgida
Christoph Bussler
Marco Antonio Casanova
Silvana Castano
Tiziana Catarci
Giuseppe De Giacomo
Stefan Decker
Rainer Eckstein
Johann Eder
Mohand Said Hacid
Jeff Heflin
Jim Hendler
Edward Hung
Arantza Illarramendi
Vipul Kashyap
Larry Kerschberg
Ross King
Roger

Harumi Kuno
Georg Lausen
Michele Missikoff
John Mylopoulos
Wolfgang Nejdl
Christine Parent
Thomas Risse
Heiko Schuldt
Peter Schwarz
Peter Spyns
York Sure
Sergio Tessaris
David Toman
Guido Vetere
Chris Welty

DOA 2006 Program Committee

Cristiana Amza
Matthias Anlauff
Mark Baker
Guruduth Banavar
Gordon Blair
Harold Carr
Geoff Coulson
Francisco "Paco" Curbera
Frank Eliassen
Tomoya Enokido
Patrick Eugster
Pascal Felber
Jeff Gray
Stefan Gruner
Mohand-Said Hacid
Franz Hauck
Naohiro Hayashibara
Hui-Huang Hsu
Mehdi Jazayeri
Eric Jul
Bettina Kemme
Fabio Kon
Joe Loyall
Peter Loehr
Frank Manola
Keith Moore
Francois Pacull
Simon Patarin
Peter Pietzuch
Joao Pereira
Arno Puder
Rajendra Raj
Andry Rakotonirainy
Luis Rodrigues
Isabelle Rouvellou
Rick Schantz
Heinz-W. Schmidt
Douglas Schmidt
Richard Soley
Michael Stal
Jean-Bernard Stefani
Stefan Tai
Hong Va Leong
Steve Vinoski
Norbert Voelker
Andrew Watson
Torben Weis
Doug Wells
Michael Zapf

OTM Conferences 2006 Additional Reviewers

Carola Aiello
Reza Akbarinia
Nicolas Anciaux
Samuil Angelov
Fulvio D'Antonio
Philipp Baer
Gabriele Barchiesi
Carlo Bellettini
Domenico Beneventano
Jesus Bermudez

Devis Bianchini
Steffen Bleul
Ralph Bobrik
Silvia Bonomi
Abdelhamid Bouchachia
Jose de Ribamar Braga
Vanessa Braganholo
Lars Braubach
Gert Brettlecker
Andrea Cali
Ken Cavanaugh
Emmanuel Coquery
Felix J. Garcia Clemente
Fabio Coutinho
Georges DaCosta
Antonio De Nicola
Fabien Demarchi
Henry Detmold
Christoph Dorn
Viktor S. Wold Eide
Hazem Elmeleegy
Mohamed Y. ElTabakh
Michael Erdmann
Rik Eshuis
Katrina Falkner
Alfio Ferrara
Ilya Figotin
Anna Formica
Nadine Froehlich
Jochen Fromm
Mati Golani
Gangadharan Gr
Tim Grant
Francesco Guerra
Pablo Guerrero
Yuanbo Guo
Peter Haase
Hakim Hacid
Christian Hahn
Bjorn-Oliver Hartmann
Martin Henkel
Eelco Herder
Edward Ho
Thomas Hornung
Sergio Ilarri
Markus Kalb
Marijke Keet
Kyong Hoon Kim
Mirko Knoll
Natallia Kokash
Iryna Kozlova
Christian Kunze
Steffen Lamparter
Christoph Langguth
Marek Lehmann
Domenico Lembo
Elvis Leung
Mario Lezoche
Baoping Lin
An Liu
Hai Liu
Andrei Lopatenko
Carsten Lutz
Linh Thao Ly
Jurgen Mangler
Vidal Martins
Pietro Mazzoleni
Massimo Mecella
Michele Melchiori
Eduardo Mena
Marco Mesiti
Tommie Meyer
Hugo Miranda
Jose Mocito
Thorsten Moeller
Stefano Montanelli
Cristian Madrigal Mora
Francesco Moscato
Anan Mrey
Dominic Müller
Sharath Babu Musunoori
Meenakshi Nagarajan
Dirk Neumann
Johann Oberleitner
Nunzia Osimi
Zhengxiang Pan
Paolo Perlasca
Illia Petrov
Horst Pichler
Christian Platzer

Antonella Poggi
Alexander Pokahr
Konstantin Pussep
Abir Qasem
Michael von Riegen
Francisco Reverbel
Haggai Roitman
Kurt Rohloff
Dumitru Roman
Florian Rosenberg
Nicolaas Ruberg
Kai Sachs
Martin Saternus
Monica Scannapieco
Daniel Schall
Sergio Serra
Kai Simon
Esteban Lean Soto
Stefano Spaccapietra
Michael Springmann
Iain Stalker
Nenad Stojanovic
Umberto Straccia
Gerd Stumme
Francesco Taglino
Robert Tairas
Wesley Terpstra
Eran Toch
Anni-Yasmin Turhan
Martin Vasko
Salvatore Venticinque
Maurizio Vincini
Johanna Voelker
Jochem Vonk
Denny Vrandecic
Andrzej Walczak
Ting Wang
Thomas Weise
Thomas Weishupl
Christian von der Weth
Karl Wiggisser
Hui Wu
Jianming Ye
Sonja Zaplata
Weiliang Zhao
Uwe Zdun
Ingo Zinnikus

Table of Contents – Part I

Cooperative Information Systems (CoopIS) 2006 International Conference

Keynote

Distributed Information Systems I

Distributed Information Systems II

Workflow Modelling

Workflow Management and Discovery

Dynamic and Adaptable Workflows

Services Metrics and Pricing

Formal Approaches to Services

Trust and Security in Cooperative IS

P2P Systems

Collaborative Systems Design and Development

Collaborative Systems Development

Cooperative IS Applications

Ontologies, Databases and Applications of Semantics (ODBASE) 2006 International Conference

Keynote

Foundations

Metadata

Design

Ontology Mappings

Information Integration

Agents

Contexts

Similarity and Matching

Errata

Table of Contents – Part II

Grid Computing, High Performance and Distributed Applications (GADA) 2006 International Conference

Keynote

Resource Selection and Management

P2P-Based Systems

Grid File Transfer

Parallel Applications

Scheduling in Grid Environments

Autonomous and Autonomic Computing

Grid Infrastructures for Data Analysis

Access Control and Security

Programming Aspects for Developing Scientific Grid Components

Databases and Data Grids

Distributed Applications

Evaluation

Distributed Objects and Applications (DOA) 2006 International Conference

Keynote

Services

Communications

Searching Techniques

Types and Notations

Adaptivity

Middleware

Distribution Support

Self-organisation

CoopIS 2006 International Conference (International Conference on Cooperative Information Systems) PC Co-chairs' Message

Welcome to the Proceedings of the 14th International Conference on Cooperative Information Systems (CoopIS 2006), which was held in Montpellier, France, from October 29 to November 3, 2006.

The CoopIS conferences provide a forum for exchanging ideas and results on scientific research from a variety of areas, such as CSCW, Internet data management, electronic commerce, human-computer interaction, workflow management, agent technologies, P2P systems, and software architectures, to name but a few. We encourage the participation of both researchers and practitioners in order to facilitate exchange and cross-fertilization of ideas and to support the transfer of knowledge to research projects and products. Towards this goal, we accepted both research and experience papers.

This year's conference included sessions that cover the following topics: distributed information systems; workflow modelling, management and discovery; dynamic and adaptable workflows; formal approaches to services and service metrics and pricing; trust and security in cooperative IS; P2P systems; collaborative systems design and development and cooperative IS applications.

This high-quality program would not have been possible without the authors who chose CoopIS as a venue to submit their publications. Out of 131 submitted papers, we selected 38 full papers and 7 short papers / posters. To round up this excellent program, Frank Leymann from the University of Stuttgart agreed to be our keynote speaker.

We are grateful for the dedicated work of the 59 experts in the field (including their co-reviewers) who served on the Program Committee.

Finally, we are deeply indebted to Kwong Yuen Lai for his hard work, enthusiasm and almost round-the-clock availability. He was key to facilitating the paper management process and making sure that the review process stayed on schedule. We would also like to thank Robert Meersman and Zahir Tari for their support and response to our questions.

We hope that you enjoy this year's selection of contributions for CoopIS and we look forward to meeting you in future conferences.

August 2006

Mike Papazoglou, Tilburg University, Netherlands
Louiqa Raschid, University of Maryland, USA
Rainer Ruggaber, SAP Research Center, Germany

R. Meersman, Z. Tari et al. (Eds.): OTM 2006, LNCS 4275, p. 1, 2006.

Workflow-Based Coordination and Cooperation in a Service World

Frank Leymann

Institute of Architecture of Application Systems
University of Stuttgart
Universitätsstr. 38
70569 Stuttgart
Germany
Leymann@iaas.uni-stuttgart.de

Abstract. One of the most important roles of workflow technology in a service oriented environment is that of providing an easy to use technology for service composition (so-called "orchestration"). Another important composition model in this domain is based on the technology of "coordination protocols". We sketch the relation between orchestration and coordination protocols by describing two application areas of both technologies: the introduction of subprocesses to the service oriented world, and facilitating outsourcing by making splitting processes much easier. Cooperation aspects of workflow technology are emphasized by sketching the inclusion of human tasks in orchestrations. Finally, the benefit of combining semantic technologies with orchestrations is outlined ("semantic processes") which aims in simplifying the creation of orchestrations.

1 Introduction

Service oriented computing (SOC) is a major step forward in mastering heterogeneity of IT artifacts (e.g. [1], [10]). The corresponding architectural style (SOA, i.e. Service Oriented Architecture) is widely adopted in both, industry and academia. Various standards (WS*, i.e. Web Service Standards) have been proposed to support service orientation in an interoperable manner [16].

Workflow technology has been established as a key contributor to the success of the service environment: via BPEL workflow technology allows to easily compose new services out of existing services ("orchestration" – cf. section 3). This enables new business models for software and enables non-IT professionals to create services.

To make service composition (cf. section 2) even more easier and more productive extensions of BPEL are needed to support subprocesses (cf. section 5) and to semantically discover services to be composed in business terms (cf. section 8). Also outsourcing of process fragments via simple splits must be made much simpler (cf. section 6). Finally business processes often require human interactions (cf. section 7).

R. Meersman, Z. Tari et al. (Eds.): OTM 2006, LNCS 4275, pp. 2 – 16, 2006.

Most of these advanced features of orchestration require another service composition model as their underpinning: coordination (cf. section 4). We discuss orchestration and coordination as well as their relation w.r.t. the advanced features in this paper.

2 Service Composition Models

In this section, we briefly sketch various approaches of how services can be composed.

2.1 The Notion of Services and Composition

A *service* is a function that is made available at a network endpoint via certain transports and serialization formats. Furthermore, a service supports or requires certain non-functional characteristic like message-level security, transactional context, privacy, etc. Finally, a service is made available to a specific requestor under certain terms and conditions like price and payment method, or more sophisticated service level agreements (SLAs).

An important aspect of a service is its "always on" semantics: in contrast to the object paradigm a user of a service has no need to create a service or to destroy a service after its use. Thus, a service is "just there" whenever needed similar to gas, water, power etc. In this sense, services show a lot of characteristics of traditional utilities and enable a new business model called "Software as a Service" (SaaS).

Often, a service may not solve a complete business problem. This is an aspect the service paradigm has in common with the component paradigm. I.e. in order to solve a real-world business problem the functionality provided by a certain service typically has to be extended by functionality provided by additional services. Thus, services often have to be composed to solve complete business problems. *Composition* provides a means to specify how the corresponding services have to interact in order to solve a complete business problem. The result of a composition may again be a service, which in turn may be composed with yet other services.

2.2 Composition Taxonomy

Figure 1 is a modified version of the taxonomy of service aggregations introduced in [6]: services may be composed in a typed manner or in an untyped manner. Typed composition already defines the types of services to be composed, i.e. their port types. Untyped composition does not make any assumption about the type of services that interact to solve a business problem.

Both kinds of composition have constrained and unconstrained variants. A cons - trained composition model specifies the rules of the interaction between the aggregated services, while an unconstrained composition model leaves the details of the interaction open. Examples of typed constrained composition models are "orchestration" (cf. section 3.1) and "choreography" (cf. section 3.3) which both restrict the ordering of application messages exchanged between the composed

services. An example of an untyped constrained composition model is a “coordination type” (cf. section 4.1) which does not make any assumption about the application level messages exchanged but does specify how agreement about the success of the interaction between the composed services is reached out-of-band.

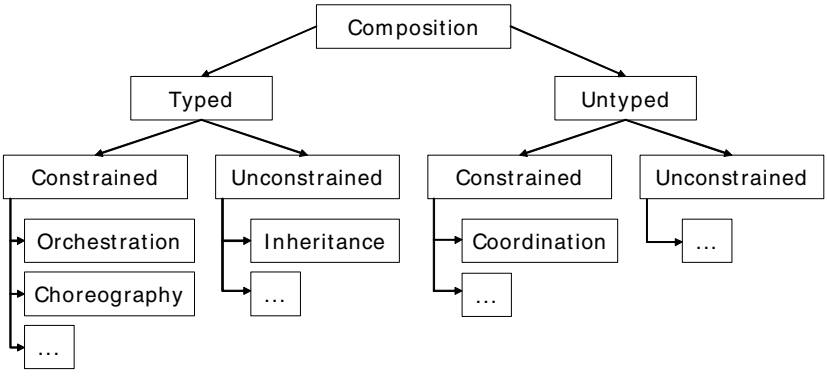


Fig. 1. Taxonomy of Service Composition Models

3 Orchestration

Since the late 80ies workflow technology (i.e. workflow modeling and workflow execution [3], [14]) has been used to compose higher-level business functionality out of individual (composed or non-composed) functions. Business processes are represented as such workflows. Thus, it is not surprisingly that this technology became the most widespread means of composition on the service world.

3.1 Workflow-Based Composition

A workflow defines the potential flow of control and data amongst a set of activities. An *activity* is an individual step to be performed to contribute to the overall goal of the workflow. The *control flow* specifies a partial order on the set of activities corresponding to a graph structure with the activities as the nodes of the graph. Business *rules* are used to evaluate which path in the graph is to be followed in the actual context of a workflow. The actual *context* is defined as the data used as input for the activities and rules, and the data produced as output by the activities; especially, the context includes the messages exchanged between the workflow and the outside world. The way how input data is assembled from the context and output data is disassembled into the context is a matter of the *data flow* specification of a workflow. Messages that are consumed by a workflow and that may result in a response message are the basis for defining individual operations and, thus, rendering a workflow as a set of services provided by the workflow to the outside world.

An activity specifies especially which kind of function is needed to perform the individual step the activity represents. In a service world, an activity refers to a

service providing the corresponding individual function. Typically, an activity does not point directly to a certain endpoint implementing the service but to the type of service needed (i.e. an operation of a port type). When the model of a workflow is deployed additional information is provided that allows to discover a concrete endpoint during runtime. This way, a workflow model becomes adaptable to the specific situations of particular users of a workflow model.

Modeling a workflow (i.e. a business process) is often done by experts from the corresponding business domain. Modeling business processes does not require detailed information technology skills as needed for creating programs in a programming language. Creating programs based on a programming language is referred to as *programming in the small*. In contrast to this, modeling a business process is often called *programming in the large*. Thus, business domain experts do programming in the large by using workflow technology to compose new services out of existing services. In a service world, workflow-based composition of services from services is referred to as *orchestration*.

3.2 BPEL and Its Extensibility

Business Process Execution Language (BPEL – [17], [18] and [15] for an overview) is the established standard for specifying orchestrations. The corresponding specification defines an XML-based language to define an orchestration and an associated operational semantics to define how to perform an orchestration. Thus, effectively, BPEL specifies a metamodel for orchestration. Modeling tools support BPEL directly or allow to export workflow models in the BPEL XML format. Workflow systems support importing BPEL XML format and execute it.

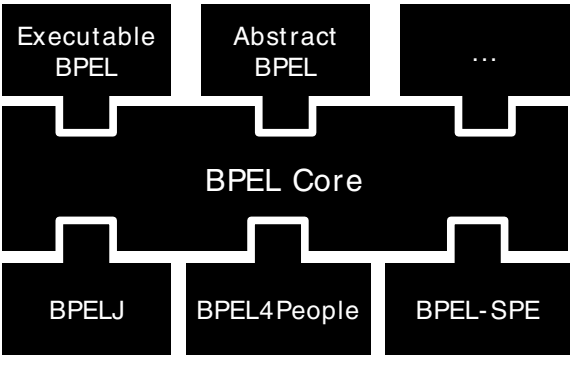


Fig. 2. Language Structure of BPEL Supporting Extensibility

Workflow technology has many different facets. These many facets can only be standardized over time. Because of this, BPEL itself is extensible by defining a core which is built to support different aspects of workflow technology on top of it (cf. Figure 2). The core part defines the language and metamodel elements of orchestrations (note: the current draft of BPEL 2.0 does not longer talk explicitly of a

core part although the extensibility mechanism is still the same). On top of the core additional language elements are defined that are required to specify *executable* process models. Another extension of the core supports specifying abstract processes: an *abstract* process is not necessarily executable but describes externally observable behavior of an executable process, for example. The kind of relation (indicated by "R" in Figure 3 between the executable process E^1 and associated abstract process A^1) between an abstract process and an executable process is part of the definition of the kind of abstract process: an abstract process may define views on executable processes, or an abstract process may define ordering constraints on the use of port type operations etc [8].

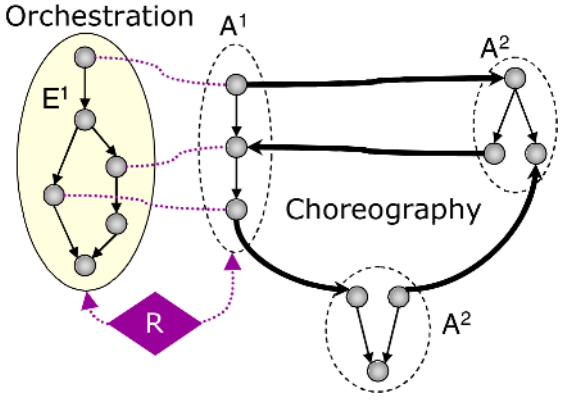


Fig. 3. Choreography, Orchestration, and Abstract/Executable Processes

Other extensions are proposed as separate specifications. For example, activities that are performed by people are envisioned by BPEL4 People; more details will be discussed in section 7. Activities that are implemented by another process (a.k.a. subprocess) are sketched by BPEL Subprocess Extension (BPEL-SPE); more details will be discussed in section 5.

3.3 Orchestration vs. Choreography

Both, abstract processes as well as executable processes are referred to as orchestrations. An important aspect of the definition of an orchestration is that it describes what happens at a single partner side. I.e. an orchestration defines the workflow at a single partner as well as the service types expected from its partners (so-called "partner links" in BPEL). But an orchestration does not care whether the partner services are implemented by orchestrations or by other means; i.e. the partner ends are opaque for an orchestration.

An orchestration sends messages to or expects messages from its partners in the order defined by the corresponding workflow. From an orchestration point of view the partners are expected to behave accordingly. But defining the workflows at the

partner ends such that they fit together is highly non-trivial. To enable proper definitions all partner ends of an interaction between multiple partners should be transparent. Then, the various activities of the partner workflows can be "wired" such that it is defined which activity at which partner sends or receives messages in which order from which partner (cf. Figure 3). Such a definition is referred to as *choreography*. A choreography defines the message exchange between multiple partners by specifying (1) for each partner one or more orchestrations defining the ordering behavior of the particular partner, and (2) the wiring between the activities of the orchestrations of the partners. The latter is sometimes referred to as the *global model* of interaction between partners [23].

There is a proposal by W3C for a choreography standard called WS-CDL [21]. This proposal does not only define a wiring mechanism between orchestrations but it also defines orchestration features in parallel to BPEL. But this violates the principles of modularization and composability that Web service standards typically follow [16]. Separate wiring mechanisms that fit together with BPEL would comply with these principles and, thus, would better fit into the overall WS* stack of standards. Research is going on to investigate such separate and composable wiring.

4 Coordination

The concept of coordination is used in distributed transaction processing since the early 70ies. In the service world the applicability of coordination technology is extended to reach agreement in any kind of distributed computations.

4.1 Abstract Coordination Model

In abstracting the use of coordination from transaction processing, the following model of coordination results (cf. Figure 4): A distributed *computation* is performed by a set of *participants* each of which executes certain application specific services (indicated by "application ports ap" in the figure) on behalf of the computation. All participants contribute to the overall success of the distributed computation, thus, they have to jointly agree on its outcome. For the purpose of outcome agreement each participant provides additional ports (called "protocol ports pp" in the figure) which are used to communicate messages of the supported *agreement protocol*. Since different participants may play different roles in the outcome agreement they may support different agreement protocols, i.e. different types of protocol ports. The collection of agreement protocols needed for successfully agreeing on the outcome of a distributed computation is referred to as *coordination type*.

Typically, the participants contributing to a distributed computation don't know each other – this is the result of the loose-coupling aspect of SOC. In order to run an agreement protocol between initially unknown participants the role of a *coordinator* is introduced. On request of one of the participants (typically the first participant initiating a new computation) the coordinator creates a new *coordination context*. This context includes a unique identifier (indicated by the `ID` field in the figure) for the new distributed computation. Each computation is associated with a coordination type used to reach agreement on the outcome between the participants; the name of

the coordination type is part of the coordination context too. Each participant of the computation has to register with the coordinator (the address of which is also part of the coordination context) for one of the agreement protocols specified by the coordination type of the computation. A participant is implicitly requested to join a computation and register with the coordinator as soon as the participant receives a new coordination context either with a request message or out-of-band (so-called *infection*). When the computation ends the coordinator runs the participant specific agreement protocol with each of the participants to finally determine the overall outcome of the computation.

For example, in Figure 4 the computation amongst participants P_1, P_2, and P_3 is identified by identifier `ID=42`. Participants P_1 and P_2 use agreement protocol R^Y to communicate with the coordinator about the outcome of the computation. Participant P_3 use agreement protocol R^X for that purpose. The coordination type of the computation is $\{R^X, R^Y\}$.

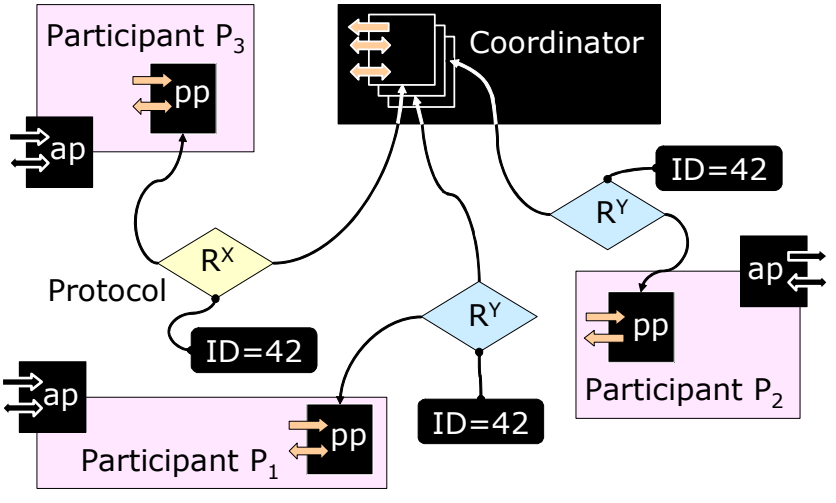


Fig. 4. Coordination Protocols and Roles

4.2 Use of Coordination

In the service world coordination is used to run distributed transactions between services in heterogeneous environments ([19], [20]). Several coordination types have be proposed spanning a spectrum from variants of the classical two-phase-commit protocol to long running transactions based on compensation based recovery. For example, the long running transaction model encompassed in BPEL has been specified as a separate coordination type.

But the abstract coordination model can be used in very different scenarios. For example, coordination types can be defined that support various kinds of auctions

(e.g. [13]). Also, the lifecycle of subprocesses can be coupled to the lifecycle of corresponding parent processes by means of coordination types: this is discussed next.

5 Subprocesses

An activity of a process can be implemented by another process. When a process is used to provide the service to be performed by an activity within another process the former process is referred to as *subprocess*. Subprocesses are an important mechanism for reuse in workflow technology: new process models may make use of already existing process models reducing the burden of modelling new processes and improving their quality.

5.1 Process Lifecycle as Coordination Type

A process model can be instantiated both, as a standalone process or as a subprocess. A process instantiating another process as one of its subprocesses is referred to as a *parent process*. A process instantiated as subprocess must give up some of its autonomy because conceptually a subprocess is part of its parent process. This implies, that a subprocess must terminate when its parent process terminates, or that a subprocess must be compensated when the activity it implements is compensated, for example.

The request to give up its lifecycle autonomy is implicitly passed to the subprocess when it is instantiated by the parent process: a coordination context corresponding to the agreement protocol for lifecycle outcome is created by the parent process and passed to the subprocess (cf. Figure 5). Thus, the subprocess is infected by the coordination context and registers as participant: from now on, the lifecycle of the subprocess is subordinated to the lifecycle of the parent process.

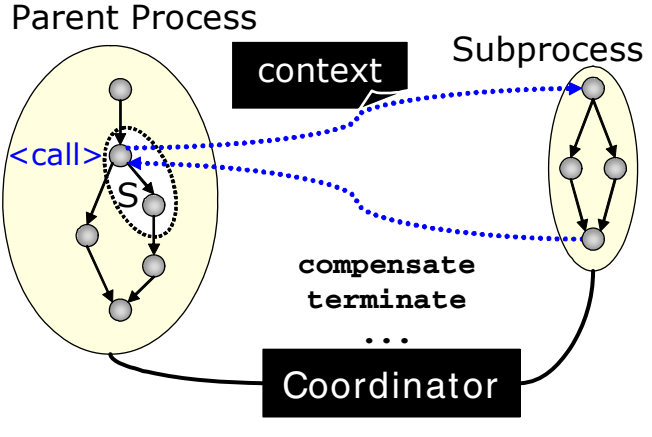


Fig. 5. Coordination Protocol between Parent Process and Subprocess

5.2 Calling a Subprocess

An envisioned extension of BPEL proposes such a coordination type (cf. Figure 5) [28]. Also, a new <call> activity is introduced to be used within a process to instantiate a subprocess. When encountering a <call> activity the encompassing process understands that it is about to become a parent process: It will create a corresponding coordination context and pass it to the new subprocess to instantiate it.

The subprocess gets infected and registers as participant. When the subprocess reaches its end it returns a response message to the parent process. Usually, a process is completed when it reaches its end and, thus, it can be garbage collected. But a subprocess must hold all of its relevant data because it may happen that the parent process decides to compensate the subprocess, for example. On reception of the response message the <call> activity within the parent process completes and navigation continues.

When the parent process decides to compensate the <call> activity a corresponding compensate protocol message is sent to the subprocess. On receiving this message compensation of the whole subprocess is performed. When this finishes the <call> activity is considered to have compensated in the parent process and navigation continues. Effectively, a <call> activity is compensated in a "deep" manner [14] by the default compensation mechanism specified in BPEL.

It should be obvious now that calling a subprocess is different from "just" invoking a service that is implemented by another process: the fact whether an invoked service is realized by another process or not is opaque to the invoking process, the invoked process does not give up its autonomy, and no agreement protocol is run between the invoking process and the invoked service.

6 Splitting Processes

One scenario for using a subprocess from another process is outsourcing the functionality corresponding to the subprocess to an external partner. When the outsourcing needs are known at the time the parent process is modelled subprocesses provide an excellent technology for that purpose. But often, outsourcing requirements are determined long time after a process is used in production.

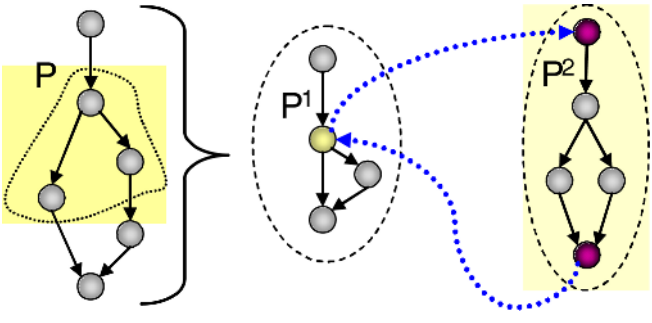


Fig. 6. Outsourcing a Fragment of a Process

6.1 Outsourcing Process Fragments

Monitoring or analysis techniques are used to determine potentials to improve processes of an enterprise. This may result in identifying fragments of a process that should be outsourced: the fragments have to be turned into subprocesses run by external partners, and the original process must be changed to call these subprocesses.

Figure 6 gives an example: process P contains a fragment (marked area) that is not competitive w.r.t. various metrics. This fragment is turned into a subprocess P^2. The original process P is modified by extracting the fragment and substituting it by a `<call>` activity resulting in process P^1. [7] describes how the corresponding rewriting of the process models and fragments can be done automatically.

6.2 Resulting Coordination Needs

Fragments to be outsourced may contain activities that are included in structured activities like scopes or while loops. In this case splitting the process apart introduces coordination needs. The following example illustrates this:

In Figure 7 activities C, D, and E are identified to be outsourced. Activities B and C are contained in scope S, i.e. S is split into scopes S_1 and S_2 contained in two different process models run at different partners N_1 and N_2. The semantics of scope S in the left side of the figure mandates that as long as activities B and C are not completed, the link from S to D must not be followed. Since splitting a process has to maintain the semantics of the original process model the semantics of scope S must be reflected by the two scopes S_1 and S_2 collectively. Consequently, when C completes at partner N_2 following the link from S_2 to D requires that B at partner N_1 already completed. Thus, S_1 and S_2 are dependent on each other, they must agree on their own completion (and faulting, compensation etc – which is not covered here). The corresponding agreement protocol is run by a coordinator.

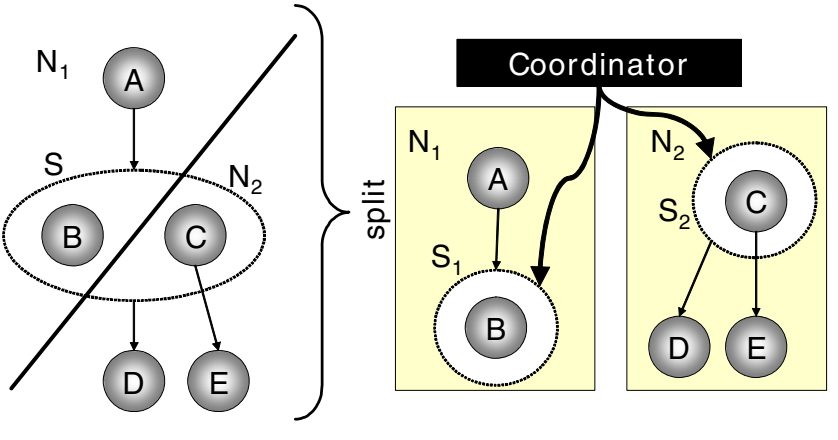


Fig. 7. Splitting Structured Activities and Resulting Coordination Needs

7 Human Tasks in Orchestrations

Activities that are performed by human beings are an integral aspect of workflow technology since its inception. BPEL itself does not include such activities because of its initial focus on composition of automatic activities. [27] describes how people interactions can be defined for BPEL using BPEL's extensibility.

7.1 People Links

Traditionally, people capable or required to perform a certain activity of a workflow are derived by a so-called "staff query" associated with the subject activity [14]. Such a staff query is evaluated on an organizational database of the enterprise performing the workflow.

Since BPEL process models are intended to be portable across different enterprises no assumption can be made on such a common organizational database. Because of this, BPEL4People [27] does not foresee to attach a staff query directly to a people facing activity but it introduces a level of indirection. This indirection is called a people link (see Figure 8).

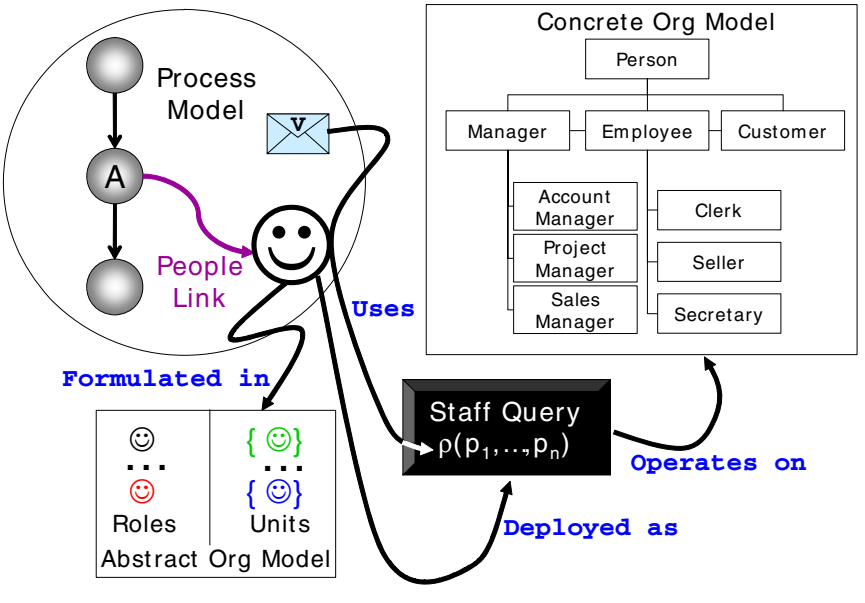


Fig. 8. Main Artifacts of the BPEL Extension for Activities Supporting People

A *people link* abstractly describes the kind of people that should be made aware that a certain task is awaiting their action. This description is given in an informal manner by just using text documenting this people. The text reflects an abstract organizational model corresponding to the business domain the business process

stems from. A people link may also refer to the process context to support more sophisticated people assignments: for example, a people link may refer to a variable to specify assignments like “sales person in region `reg_var`” where `reg_var` is a process variable.

During deployment a people link must be mapped to a query on an organizational database of the deploying enterprise: the element of the abstract organizational model is transformed into a corresponding filter expression, and parameters can be used by exploiting the elements of the process context passed through the people link. That way, a staff query is created and associated with a people facing activity.

7.2 Tasks

A *task* is the type of a piece of work to be performed by a human being. It refers to the information supporting the corresponding person in performing the proper work. Furthermore, a task defines data to be used for displaying the work request on a task list (a.k.a. worklist [14]).

Tasks can be instantiated. Basically, a task instance corresponds to what is traditionally called a workitem [14]. In BPEL4People tasks are reusable artifacts that are used to define people facing activities. When the workflow engine reaches such an activity it evaluates the staff query associated during deployment with its people link and distributes the work request to the resulting people. Hereby, the display information of the task is used.

The work requests are then rendered on corresponding task lists. People look for work they have to perform on task lists associated with them. A person can assume responsibility (“check out”) for a piece of work. When the work is done this is explicitly signaled to the workflow engine (“check in”) and result data produced by the task is passed back. The workflow engine then continues navigation.

8 Semantic Processes

Binding a service to a requestor is typically based on type information ([9], [22]). More sophisticated scenarios use descriptions of non-functional properties required from the services for discovery or may even establish service level agreements between the requestor an the service ([10], [11], [12]). However, the meaning of the functionality offered by the service must be known in advance.

8.1 Semantic Web Services

The concept of *semantic* Web services tries to overcome this ([2], [24]): a semantic Web service uses semantic Web technologies to describe the meaning of the functions provided, e.g. by annotating operations accordingly. A requestor, too, describes the functionality needed semantically, and this semantic description of the request is used for discovery of a matching semantic Web service. Based on such semantic descriptions request may even be mediated, e.g. be deriving appropriate message transformations. Corresponding infrastructures are subject of current research projects (e.g. [25]).

8.2 Semantics in Orchestrations

Workflow environments can benefit from semantics too [4].

First, when modeling business processes the semantics of the function performed by an activity may be specified instead of specifying the signature of that function. This allows business users or domain experts to define more aspects of a business process than before. The semantics specified by a business user may be used during modeling time to discover and propose appropriate service types. Going further, the semantics of a collection of activities may even be used to discover and propose a fragment of a process model realizing the specified business need ("auto-completion").

Second, the semantics specified by the domain experts may be used during runtime to discover and bind appropriate services: the workflow engine interacts with a service bus by passing the semantic description in and getting the response back [5].

An encompassing infrastructure covering both, modeling aspects as well as runtime aspects is currently under development [26]. A proposal of extending BPEL to support semantics is on its way too.

9 Conclusion

9.1 Summary

In this paper we described some aspects of service composition, coordination and cooperation of workflow technology in a service world.

We sketched a taxonomy for service composition models and positioned orchestration within that taxonomy. After reminding the basic ingredients of a workflow language we discussed the extensibility architecture of BPEL, i.e. the workflow language established in the service world. The notion of choreography has been contrasted with orchestration.

Coordination has been described as another composition model supporting outcome agreement in collections of dynamically assembled services. The role of coordination for realizing subprocesses and outsourcing arbitrary process fragments has been discussed.

Cooperation between human beings based on workflow technology in a service world has been outlined: the main idea is to abstract concrete staff assignments and defer binding to an enterprise organizational database until deployment time. That way, the guiding principle of portability is maintained by the corresponding BPEL extension.

Finally, semantic Web services and semantic business process have been sketched. These technologies are expected to significantly ease composition of services.

9.2 Future Work

We are currently working on algorithms to detect situations described in section 6.2 automatically and to rewrite the affected process models accordingly. Furthermore, a prototype is being built in which a BPEL engine interacts with a coordinator to maintain the semantics of split structured activities.

An extension of BPEL is in progress to combine semantic Web service technology and orchestration. We also investigate language elements needed on top of BPEL to support choreography in a modular and composable manner; a corresponding tool is under development.

References

1. G. Alonso, F. Casati, H. Kuno, V. Machiraju. Web Services, Springer 2004.
2. J. Cardoso, A. Sheth (ed.). Semantic Web Services, Processes and Applications, Kluwer 2006.
3. M. Dumas, W.M.P. van der Aalst, A.H.M. ter Hofstede. Process-Aware Information Systems, John Wiley & Sons, Inc., 2005.
4. M. Hepp, F. Leymann, J. Domingue, A. Wahler, D. Fensel. Semantic Business Process Management: Using Semantic Web Services for Business Process Management, Proc. IEEE ICEBE 2005 (Beijing, China, October 18-20, 2005).
5. D. Karastoyanova, F. Leymann, J. Nitzsche, B. Wetzstein, D. Wutke. Parameterized BPEL Processes: Concepts and Implementation, Proc. BPM'2006 (Vienna, Austria, September 5 – 7, 2006).
6. R. Khalaf, F. Leymann. On Web Services Aggregation, Proc. VLDB-TeS'03 (Berlin, Germany, September 2003).
7. R. Khalaf, F. Leymann. Role-Based Decomposition of Business Processes using BPEL, Proc. ICWS'2006 (Chicago, IL, USA, September 18 – 22, 2006).
8. R. Khalaf, A. Keller, F. Leymann. Web Services Business Processes: Architecture and Applications, IBM Systems Journal 45(2) (2006).
9. D. König, M. Kloppmann, F. Leymann, G. Pfau, D. Roller. Web Services Invocation Framework: A Step towards Virtualization Components, Proc. XMIDX 2003 (Berlin, Germany, February 16 -17, 2003), Lecture Notes in Informatics Volume P-24, GGI 2003.
10. F. Leymann. Web Services: Distributed applications without limits, Proc. BTW'03 (Leipzig, Germany, February 2003), Lecture Notes in Informatics Volume P-26, GI 2003.
11. F. Leymann. The Influence of Web Services on Software: Potentials and Tasks, Proc. 34th Annual Meeting of the German Computer Society (Ulm, Germany, September 20 – 24, 2004), Lecture Notes in Informatics Volume P-50, GI 2004.
12. F. Leymann. The (Service) Bus: Services Penetrate Everyday Life, Proc. 3rd Intl. Conf. on Service Oriented Computing ICSOC'2005, (Amsterdam, The Netherlands, December 13 – 16, 2005), LNCS 3826 Springer 2005.
13. F. Leymann, S. Pottinger. Rethinking the Coordination Models of WS-Coordination and WS-CF, Proc. IEEE ECOWS 2005 (Växjö, Sweden, November 14-16, 2005).
14. F. Leymann, D. Roller. Production Workflow: Concepts and Techniques, Prentice Hall 2000.
15. F. Leymann, D. Roller. Modeling Business Processes with BPEL4WS, Information Systems and e-Business Management (ISeB), Springer 2005.
16. S. Weerawarana, F. Curbera, F. Leymann, T. Storey, D.F. Ferguson. Web Services Platform Architecture, Prentice Hall 2005.

Links: (followed August 23, 2006)

17. BPEL - Business Process Execution Language For Web Services V1.1, BEA, IBM, Microsoft, SAP & Siebel, 2003, http://www-106.ibm.com/developerworks/library/ws-bpel/

18. OASIS BPEL Technical Committee, http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel
19. WS-C – Web Services Coordination V1.0 ftp://www6.software.ibm.com/ software/developer/library/WS-Coordination.pdf
20. OASIS WS-C Technical Committee http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=ws-tx
21. WS-CDL - Web Services Choreography Description Language V1.0 http://www.w3.org/TR/ws-cdl-10/
22. WSIF - Web Service Invocation Framework, http://ws.apache.org/wsif/
23. WSFL – Web Service Flow Language http://www-306.ibm.com/software/solutions/webservices/pdf/WSFL.pdf
24. SAWSDL – Semantic Annotations for WSDL http://www.w3.org/2002/ws/sawsdl/spec/SAWSDL.html
25. DIP Integrated Project, http://dip.semanticweb.org/
26. SUPER Integrated Project, http://www.ip-super.org/
27. M. Kloppmann, D. Koenig, F. Leymann, G. Pfau, A. Rickayzen, C. von Riegen, P. Schmidt, I. Trickovic. WS-BPEL Extension for People (BPEL4People), IBM, SAP 2005 http://www-128.ibm.com/developerworks/library/specification/ws-bpel4people/
28. M. Kloppmann, D. Koenig, F. Leymann, G. Pfau, A. Rickayzen, C. von Riegen, P. Schmidt, I. Trickovic. WS-BPEL Extension for Subprocesses (BPEL-SPE), IBM, SAP 2005 http://www-128.ibm.com/developerworks/library/specification/ws-bpelsubproc/

Distributed Triggers for Peer Data Management

Verena Kantere[1], Iluju Kiringa[2], Qingqing Zhou[3], John Mylopoulos[3], and Greg McArthur[3]

[1] National Technical University of Athens
vkante@dbnet.ece.ntua.gr
[2] University of Ottawa
iluju@site.uottawa.ca
[3] University of Toronto
{jm, qqzhou, greg}@cs.toronto.edu

Abstract. A network of peer database management systems differs from conventional multidatabase systems by assuming absence of any central control, no global schema, transient connection of participating peer DBMSs, and evolving coordination among databases. We describe distributed triggers to support data coordination in this setting. The execution of our triggers requires coordination among the involved peer databases. We present an SQL3 compatible trigger language for the P2P setting. We also extend the SQL3 processing mechanism to this setting. Our trigger processing mechanism consists of an execution semantics, a set of termination protocols to deal with peer transiency, and a set of protocols for managing peer acquaintances in presence of distributed triggers. We show preliminary experimental results about our mechanism.

1 Introduction

A P2P architecture is a node-to-node mode of communication over the Internet. Most of existing P2P applications, with some exceptions like [13, 17], do not deal with data management issues.

However, many application domains need the kind of advanced data management that would be offered by a P2P system to efficiently manage data residing in peer databases. One such domain is health care [8], where hospitals databases, independent doctors, pharmacists and pharmaceutical companies would like to participate in a P2P system in order to exchange information about medical histories of patients, medicines, symptoms of diseases, treatment methods etc. Another example is offered by genome databases. Many such databases are available today [11]. Sharing such data and exploiting information from these different databases would be useful for scientists by enhancing their capacity of acquiring new knowledge through coordination of the involved databases.

The anticipated wealth of applications mentioned above has prompted research in data management systems that can get together to build a network of peers that coordinate at run time the tasks of sharing and querying (mostly relational) data [11, 2, 17, 13]. Such a network is very similar to a conventional multidatabase. The latter relies on key concepts such as a global schema, central data management control, data integration, global access to multiple databases, and static setup of the network of participating

R. Meersman, Z. Tari et al. (Eds.): OTM 2006, LNCS 4275, pp. 17–35, 2006.

databases. Instead, in [2], an architecture for a network of peer DBMSs called Hyperion is proposed by assuming absence of any global schema, absence of any central control, evolving coordination rules among databases, and dynamic participation of peer databases. One of the major goals of the Hyperion project [2] is to augment a conventional (relational) DBMS with a P2P layer that enables the interoperability of heterogeneous peer databases. We call the DBMSs augmented with such a P2P layer **peer DBMSs**. Using their P2P layers, peer DBMSs may establish or abolish acquaintances among themselves. Two peers that have an acquaintance between them are 'acquainted' and are called 'acquaintees'. Once an acquaintance is established, peer DBMSs may coordinate and share their respective data. A peer DBMS manages both local data and the correspondences/mappings over its acquaintances. Using the latter, answering queries **locally** posed to a PDBMS, updating local data, and processing transactions are all done by taking both local data and the data in acquainted peers into account. Furthermore, coordination rules that are integrated into the the PDBMSs can enhance the interoperability of peer databases, by adding active functionality on local and remote data.

On a more practical side, a few companies such as iSpheres (www.ispheres.com), KnowNow (www.knownow.com), and IBM have recognized the importance of event-driven applications for today's increasingly complex execution of corporate business processes. These companies are developing languages for event-based computing to cope with the huge size of today's dynamic corporate information. Our research is pursuing a similar avenue by taking into account the semantical differences that almost always exist in practice between information sources that trigger reactive actions.

The contributions of this work are the following:

- We present an approach for using distributed triggers to enable and coordinate data exchange between peer databases by propagating appropriate updates to acquainted peer DBMSs. To do so, we extend the syntax of SQL3 to provide a rule language for the definition of Event-Condition-Action (ECA) rules in a peer database environment. We delineate a distributed trigger language with simple events, conditions destined to one single peer database, and actions destined to various peer databases. Though simple, this language nevertheless allows realistic coordination of peer data.
- We consider the processing of the distributed triggers in the absence of constraints. Our execution semantics extends the standard semantics of centralized SQL3 triggers by dealing with the heterogeneous and autonomous nature of peer databases.
- We propose a way of translating updates destined to remote peers using mapping tables and we describe appropriate acquaintance protocols to deal with the transient character of peer databases during the execution of distributed triggers.
- We have conducted experiments whose preliminary results show the viability of this solution for managing updates in peer DBMSs.

The next section gives a motivating example from the domain of health care. Section 3 introduces the main models used in the paper. Section 4 presents the syntax of the distributed triggers. Sections 5–6 describe the execution semantics and the various algorithms and protocols related to it. Section 7 presents preliminary experimental results. Related work is discussed in Section 8. Finally, we draw the conclusions in Section 9.

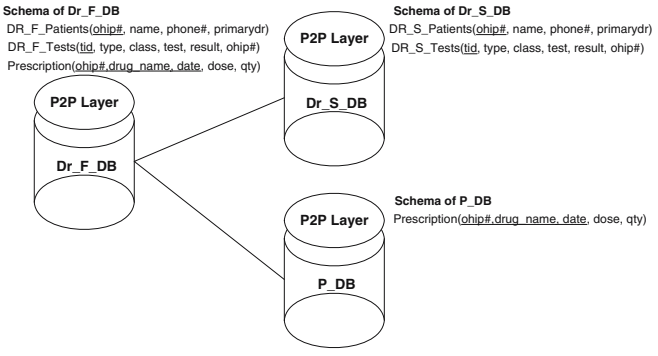


Fig. 1. A network of peer databases

2 Motivating Example

Consider the situation in which a family doctor, Dr. Smith, sees one of his regular patients, Ms. Watson, who is presenting with abdominal pains. Assume that Ms. Watson has had these pains for some time and that she has had some investigations done for this problem, but she is not sure which investigations. Moreover, she remembers to have been prescribed some medications, but does no longer know from which pharmacist.

A number of questions and issues may arise with respect to the patient's medical history such as: (1) Has the patient had a complete blood count? If so, when was that and what was the white blood cell count? (2) What medications was or is the patient taking? (3) Since Ms. Watson is one of Dr. Smith's regular patients and the latter keeps as complete a medical history as possible in his own database, Dr. Smith's data needs to be coordinated as it is updated. (4) Dr. Smith may want to have the information from any medication prescribed to Ms. Watson by pharmacists that he is aware of. (5) Dr. Smith may want to have any diagnosis from examinations done on Ms. Watson by hospitals that he is aware of.

In this situation, we assume a network of peer databases that belong to physicians, hospitals, medical laboratories, and pharmacies. Associated physicians establish acquaintances among themselves. Further acquaintances are established between physicians and associated laboratories, between the former and hospitals, between associated hospitals, and so on.

As an example consider a network of peer databases that comprises the following databases (Figure 1): a database *Dr_F_DB* that belongs to a family physician, Dr. F; a database *Dr_S_DB* that belongs to a specialist, Dr. S, associated with Dr F; a database *P_DB* that belongs to a pharmacist, P, associated with Dr F.

We use mapping tables [11] as a way of constraining data exchange between heterogeneous peer databases. Figure 2 shows two mapping tables for our domain. Intuitively, mapping tables are binary tables that provide a correspondence between data values of acquainted peer databases. At a certain extent, they also provide a simple schema-level correspondence. For example,the first mapping table of Figure 2 maps

FTest_2_STest

Dr_F_Test.test	Dr_S_Test.test
hemoglobin whitebloodcount	C0518015 C0427512

ohip_2_ohip

Dr_F_Patients.ohip#	Dr_s_Patients.ohip#
X	X

Fig. 2. Examples of mapping tables

Dr F's tests to Lab L's tests. Moreover, the second table of the same figure represents the identity function which maps each ohip [1] value of Dr F's database to itself in Dr S's database.

2.1 Distributed Triggers

The querying mechanism through which a pDBMS is able to gather data from the P2P network may not be sufficient. For example,if Dr F works in a walk-in clinic, he typically may not wish to keep complete records on his irregular patients, in which case this query and update time data coordination is enough for his instant need information. Yet, for his regular patients, Dr F may want to keep as complete a medical history as possible in *Dr_F_DB*. In this case, data updates must be complemented by a more general coordination mechanism.

Trigger 1. *If Dr F wants to be informed about any lab test performed on any of his patients by the associated Dr S, insertions alone would not be enough to express this: Dr F would like his database to be updated automatically when such new information occurs. This can be expressed with the following trigger, written in a rule language that extends SQL3 triggers:*

```
CREATE TRIGGER testInsertion
AFTER  INSERT ON Dr_S_Tests
    REFERENCING NEW AS NewTest IN Dr_S_DB
FOR EACH ROW
WHEN EXISTS SELECT P. ohip#
                FROM Dr_F_Patients P
                WHERE P.ohip# = NewTest.ohip# AND P.primarydr ='F'
BEGIN INSERT INTO Dr_F_Tests VALUES NewTest IN Dr_F_DB END
```

To express Trigger 1, we use a language that extends SQL3 triggers [12, 4]. An SQL3 trigger is a form of ECA rule with an explicit name. At definition time, each trigger is associated with a specific table. Only insertions, deletions, and updates on the associated table can activate the trigger. Let t be a SQL3 trigger; t is said to be *triggered* if a database operation on the associated table matches the event part of t. Moreover, t is said to be *activated* if it is triggered and its condition part evaluates to `true`.From now on, we use the verbs *trigger* and *activate* interchangeably. Events in SQL3 are kept simple, meaning that only single database operations are of interest. Conditions are SQL queries, and actions are SQL statements that are intended to be executed on the database instance. Triggers that are activated before their triggering (or else activating) event occurs are called BEFORE triggers; those that are activated

[1] Ontario Health Insurance Plan.

after their activating event has occurred are called AFTER triggers. SQL3 specifies an activation granularity, which is an indication of how many times the action part of an activated trigger may be executed. Two activation granularities are distinguished in SQL3: row-level, and statement-level activation. The former (expressed by FOR EACH ROW) means that the trigger action is executed for each of the rows that are affected by the database operation that activated the trigger. The latter (expressed by FOR EACH STATEMENT) means that the trigger action is executed for each statement that contains the database operation that activated the trigger, i.e. only once. Finally, SQL3 allows one to access both the current and the old state of the database (called **transition tables**) using keywords NEW_TABLE and OLD_TABLE, respectively. One may also access both the new and old values of tuples (called **transition values**) that are affected by triggering database operations using keywords NEW and OLD, respectively.

Trigger 1 is typically distributed, in the sense that its event is monitored in the database *Dr_S_DB*, and the action that it causes is executed in a different database, namely *Dr_F_DB*. The condition part of this rule is a simple select-join-project (SJP) query that will be evaluated by default (i.e. if not defined else-wise) in the database of the peer in which the trigger is created. Our distributed triggers extend typical centralized SQL3 triggers by explicitly mentioning the database in which the trigger event, condition, and actions occur.

3 Models

For our distributed triggers and their execution, we assume the relational data model [5]. In the following we use notations and concepts from [1], [19], [12].

Definition 1 (Peer to Peer Network). *A peer to peer network is a pair* $(\mathcal{N}, \mathcal{M})$*, Here,* $\mathcal{N} = (\mathcal{P}, \mathcal{L})$ *is an undirected graph, where* $\mathcal{P} = \{P_1, \cdots, P_n\}$ *is a set of peers,* $\mathcal{L} = \{(P_i, P_j) | P_i, P_j \in \mathcal{P}\}$ *is a set of acquaintances; each acquaintance* (P_i, P_j) *is associated with a set* $\mathcal{M}_{ij} \in \mathcal{M}$ *of mapping tables.*

To motivate our paper, we formulate our triggers by assuming that, though the peer schemas are heterogeneous, each peer makes all its schema visible to its acquaintees. However, this assumption is not realistic and we relax it in this section. We now assume that peers make only a subset of its schema visible to its peers.

Using notations from [16], we assume that each peer P_i maintains a set LS_i of attributes called *local schema*. We also assume that peer P_i exports a subset $ES_i \subseteq LS_i$ of attributes called *export schema* of P_i.

3.1 Local Update Model

Though a formal study of updates is beyond the scope of this paper, we formally introduce a few useful definitions and a formal model of updates for our P2P setting. For each peer P_i, we distinguish three sorts of updates: a **local** update is executed only on LS_i; a **remote** update is executed only on the ES_j of an immediate acquainted peer P_j; and a **global** update is executed both on LS_i and on the ES_j's of some or all of the immediate acquainted peers. A global update is therefore a family including a local update

and a set of remote updates. Below, we give the semantics of a local update. Section 3.2 will give the semantics of a remote update. Global updates are beyond the scope of this paper.

Consider a database update model in which users submit sequences of SQL database operations **insert**, **delete**, and **update** with their usual syntax, which we do not present due to lack of space. Each one of these operations is called an *SQL statement*. We consider the execution of SQL statements to be atomic.

Let S be an SQL statement. The set of affected rows (SAR) of S, denoted $\mathcal{A}(S)$, is the set of tuples changed by S. Formally, $\mathcal{A}(S)$ is defined as follows:

$\mathcal{A}(\textbf{insert}) = \{\langle v_1, \cdots, v_n \rangle\}$ (basic insertion)
$\mathcal{A}(\textbf{insert}) = \{t \in R_1 \bowtie R_2 \bowtie \cdots \bowtie R_m \mid$
t is an answer to the embedded SPJ query$\}$
$\mathcal{A}(\textbf{delete}) = \{t \in R \mid t \text{ satisfies } condition\}$
$\mathcal{A}(\textbf{update}) = \{\langle t, C, v \rangle \mid t \in R, C \in \textbf{att}(R), \text{and } v \in \textbf{dom}(C)\}$

Intuitively, $\langle t, C, v \rangle$ means that v is the new value of the attribute C in the tuple t. We will denote the SAR of a statement S on a specific database instance r by $\mathcal{A}(S, r)$.

The effect of a statement S, denoted $\mathcal{E}(S)$, is a tuple (I, D, U), where I, D, and U are sets of inserted, deleted, and updated tuples, respectively. Formally, we have: $\mathcal{E}(\textbf{insert})=(\mathcal{A}(\textbf{insert}), \emptyset, \emptyset)$; $\mathcal{E}(\textbf{delete})=(\emptyset, \mathcal{A}(\textbf{delete}), \emptyset)$; $\mathcal{E}(\textbf{update})=$ $(\emptyset, \emptyset, \mathcal{A}(\textbf{update}))$.

Each execution of a statement S corresponds to a set $\mathcal{T}$ of transitions $T_1, \cdots, T_n$, where $n = |\mathcal{A}(S)|$. We say that S causes the transitions in $\mathcal{T}$. Formally, a *transition* is a pair $(old\_tuple\_value, new\_tuple\_value)$ which is inductively defined as follows:

If S is an insert, then $T = (\emptyset, n)$, $n \in \mathcal{A}(\textbf{insert})$.
If S is a delete, then $T = (o, \emptyset)$, $o \in \mathcal{A}(\textbf{delete})$.
If S is an update, then $T = (o, n)$, where there is a tuple $(t, C, v) \in \mathcal{A}(\textbf{update})$, $o = t$, and n is the tuple t such that $t[C] = v$.

Definition 2 (Database State Change [12]). *A database state change is a triple* $(R, E, \mathcal{T})$*, where R is a relation,* $E \in \{\textbf{insert}, \textbf{delete}, \textbf{update}\}$*, and* $\mathcal{T}$ *is a set of transitions.*

3.2 Remote Update Model

Consider again a database update model with sequences of SQL operations **insert**, **delete**, and **update** with their usual syntax. For given acquainted peers P_i and P_j, we want to give a semantics for a remote update U_i over LS_i using mapping tables that are in place between P_i and P_j. That is, we want to translate the remote update U_i to an update U_j using the mapping tables. The intuition is to translate the remote update U_i such that the set of of affected rows of U_j only concerns tuples that are related to the set of affected rows of U_i through the mapping tables.

4 Distributed Trigger Language

A distributed SQL3 trigger is a syntactical straightforward extension of SQL3 triggers which mentions the databases in which event occurrence, condition evaluation, and

```
<trigger definition> ::= CREATE TRIGGER <trigger name>
      <trigger action time>
      <simple event> IN <db name>
     [REFERENCING <old/new values alias list>]
     [FOR EACH {ROW | STATEMENT}]
     [WHEN <trigger condition>]
     <trigger action>
<trigger action time> ::= BEFORE | [DETACHED] AFTER
<simple event> ::= INSERT | DELETE | UPDATE [OF <column name list>]
<trigger condition> ::= [NOT] <SQL query> IN <db name>
<trigger action> ::= {BEGIN ATOMIC {<SQL procedure stmt>;}+
                        IN <db name> END}+
<old/new values alias list> ::=
  OLD <table name> [AS] <identifier>
  | NEW <table name> [AS] <identifier>
  | OLD_TABLE <table name> [AS] <identifier>
  | NEW_TABLE <table name> [AS] <identifier>
```

Fig. 3. Syntax of Distributed Triggers

action execution take place. However, the parts of the distributed triggers may be associated with peer databases other than the one in which the trigger is defined and therefore associated with. Like in the traditional SQL3 triggers, conditions are SQL queries, and actions are SQL statements to be executed on database instances. Finally, BEFORE triggers and AFTER triggers are defined as in the traditional SQL3 standard. Figure 3 presents the syntax used to define our distributed triggers (See [12] for details on the SQL3 subset of this EBNF).

We extend the SQL3 requirement that the trigger names be unique in a given database schema to a uniqueness requirement in all peer databases. The event, condition and actions are defined on given peer databases. The triggering event remains simple. The trigger action time, i.e. the time when the event is signaled, is BEFORE or AFTER the actual occurrence of the respective SQL statement. The optional DETACHED keyword is added for specialization of the execution semantics of the AFTER triggers and will be discussed in section 5.2. Like the standard SQL3, the extended SQL3 allows the access of both the current and the old state of the database, as well as both the new and old values of tuples that are affected by triggering operation.

Also, the condition remains simple, and, thus, it is to be evaluated in a single database. However, the action is a set of separate sets of transactions; each one of these sets is executed atomically in one database. Note that SQL3 denotes that the action for BEFORE triggers cannot change the state of the database instance. In this paper we do not deal with distributed transactions - we leave this as future work. Nevertheless, requiring the set of transactions in one database to be executed atomically is consistent with the centralized SQL3 definition which requires the action to be executed in the same transaction with the triggering event. It is obvious that in case all parts and subparts of the trigger are declared to be executed in a single database, our extended semantics are reduced to the centralized SQL3 semantics.

As we shall see in the following, even though the extension of the SQL3 trigger syntax is straightforward, the extension of its execution semantics for the distributed and dynamic peer database setting is intriguing.

5 Processing Distributed Triggers

We consider the execution semantics of distributed triggers in the absence of constraints. We consider the case of distributed triggers with simple events and conditions to be executed in a single (possibly remote) peer database. The trigger action part, however, can be a sequence of database actions to be executed on possibly remote peers.

We extend the execution semantics of the centralized SQL3 triggers (without considering constraints) in a way appropriate to deal with the heterogeneous and autonomous nature of peer databases. We deal with heterogeneity by using mapping tables to translate updates destined to remote peers, and we deal with autonomy by considering the transient character of peer databases.

The trigger processing mechanism consists of an execution semantics, a set of termination protocols, and a set of protocols for each one of the following tasks: establishing and abolishing acquaintances, connecting to and disconnecting from a P2P network, and joining and leaving a P2P network. For simplicity, we call the third component acquaintance protocols. The execution semantics specifies how the distributed triggers are executed. Termination protocols are similar to those used for commitment in distributed DBMSs [14]: Whenever a peer involved in the processing of a trigger disconnects, all the other involved peers should be able to gracefully terminate trigger processing. Finally, acquaintance protocols address how the trigger execution process at the disconnected peer recovers when the peer reconnects.

We do not require the existence of distributed transactions involving several peers. Instead, we assume that database operations in separate peers are separate transactions.

5.1 Execution Semantics

Recall that SQL3 has immediate condition and action coupling modes, meaning that condition evaluation immediately follows event occurrence, and action execution immediately follows the condition. The following procedure summarizes the SQL3 execution semantics for a statement S in the absence of constraints [4,12]:

EXECUTE_STATEMENT(S)

1. Save current state change (if any) on a stack.
2. Determine $\mathcal{A}(S)$.
3. Create a state change $\mathcal{C}$ with $\mathcal{C} = (R, E, \mathcal{T})$, where R is the table mentioned in S, E is the update operation mentioned in S, and $\mathcal{T}$ is gained from $\mathcal{A}(S)$ as explained in Section 3.1.
4. PROCESS_BEFORE_TRIGGERS($\mathcal{C}$).
5. Apply $\mathcal{E}(S)$ to the database.
6. PROCESS_AFTER_TRIGGERS($\mathcal{C}$).
7. PROCESS_DETACHED_AFTER_TRIGGERS($\mathcal{C}$).
8. Restore the state change saved in Step 1 (if any).

To process our distributed triggers, we use the same procedure above. However, the procedures called to process the BEFORE and AFTER triggers must accommodate distributed triggers by being able to evaluate conditions and perform database updates

on remote peers. The procedures PROCESS_BEFORE_TRIGGERS($\mathcal{C}$) and PROCESS_AFTER_TRIGGERS($\mathcal{C}$) given in Figures 4 and 5, where $\mathcal{C}$ is a state change, give the execution semantics for the distributed BEFORE and AFTER SQL3 triggers, respectively. The semantics involves one coordinating peer (the coordinator), which controls the outcome of the trigger processing task, and a set of participating peers (the participants), which evaluate conditions and execute trigger actions. Figures 4 and 5 assume that the coordinator is peer P_i. For simplicity, we assume the existence of one single participant for condition evaluation. The procedure EXECUTE_STATEMENT(S), PROCESS_BEFORE_TRIGGERS($\mathcal{C}$), and PROCESS_AFTER_TRIGGERS($\mathcal{C}$) are executed by the coordinator. We use the following notations: $\tau[E]$, $\tau[C]$, and $\tau[A]$ for the event, condition, and action parts of trigger τ, respectively; $\mathcal{R}$, $\mathcal{R}_B$, and $\mathcal{R}_A$ for the sets of triggers, BEFORE triggers, and AFTER triggers, respectively; BTs, and ATs for sets of activated BEFORE and AFTER triggers, respectively. The action part $\tau[A]$ is a sequence $\tau[A_1], \cdots, \tau[A_m]$ of updates to be executed on peer databases; We use the function $select - activated - triggers(\mathcal{R})$ that takes the set $\mathcal{R}$ and returns a set comprising all *enabled* triggers whose events occur in the current state change.

Note that the coordinator of the trigger is always the database on which the event occurs. It is obvious that the coordinator can be different from the creator of the trigger, since peers can define triggers with a remote event part. The actions of the trigger are treated as sets of actions to be executed in different peers. The set of local actions is always executed prior to the execution of the remote actions. This guarantees that in case of a trigger rollback no actions of this trigger instance will have been executed on remote peers. Also, this order of action execution allows the coordinator to finish the trigger execution and proceed with other tasks without any delay added from peer communication. Of course, there can be the case that the local actions of a trigger instance have been executed and some of the remote ones have failed. However, this is acceptable, since we assume that actions on peers take place in separate transactions.

Also, note that, in case of BEFORE row-level triggers with a remote condition, the condition instances of all the rows are precomputed on the remote peer and send back to the coordinator as a set. In this way we avoid the multiple messages with condition instances and reduce the communication load. Moreover, this is compliant with the centralized SQL3 semantics, because the actions of BEFORE triggers are not allowed to change the database. Thus, the precomputed condition instances would be the same as if they would be computed in the correct order with the execution of the respective action instances. However, this does not hold for AFTER triggers: there can be cases were the action changes a table on which the condition is evaluated. Thus, the execution of AFTER row-level triggers should follow the correct order of condition evaluation and action execution. Figure 5 presents this naive execution semantics. Certainly, there are optimizations for the reduction of the number of inter-peer messages in this case. The same set-oriented execution of the condition and action as in Figure 4 can be followed in case that no action influences the tables on which the condition is evaluated. In case that there is action influence on condition evaluation, but locally, i.e. in the coordinator's database, then again we can follow the naive execution of Figure 5. However, in case of a remote condition and action that execute on the same tables the implementation of the distributed trigger execution should decide about the best solution for the

PROCESS_BEFORE_TRIGGERS($\mathcal{C}$)

1. BTs $\leftarrow select-activated-triggers(\mathcal{R}_B)$.
2. If $\mathcal{T} = \emptyset$,remove all row-level triggers from BTs.
3. Compute transition tables OLD_TABLE and NEW_TABLE from $\mathcal{T}$.
4. For each $\tau \in$ BTs in order of trigger creation, do:
   (a) If τ is a statement-level trigger, then
   if $\tau[C]$ is local, then do:
   – send it to the local evaluator
   – WAIT(*Message*)
   if $\tau[C]$ is remote (in peer P_j) then do:
   – Translate the instance of $\tau[C]$ to an instance $\tau'[C]$ of P_j using the set $\mathcal{M}_{i,j}$ of mapping tables
   – Send $\tau'[C]$ to P_j for evaluation
   – WAIT(*Message*)
   if $Message = true$, then first execute the local $\tau[A_i]'s$, and for each $\tau[A_k]$ in remote peer P_k, translate it to $\tau'[A_k]$ using the mapping tables $\mathcal{M}_{i,k}$, and execute $\tau'[A_k]$.
   (b) If τ is a row-level trigger, then:
   if $\tau[C]$ is local, for each $tr \in \mathcal{T}$, do:
   i. Set values for OLD and NEW from tr.
   ii. If $\tau[C]$ evaluates to true, execute $\tau[A]$.
   if $\tau[C]$ is remote (in peer P_j), then do:
   i. OLD_VAL = {}; NEW_VAL = {}; COND_INST = {}
   ii. for each $tr_k \in \mathcal{T}$, do:
   – Set values for OLD_k and NEW_k variables from tr_k.
   – Translate the instance of $\tau[C]$ to an instance $\tau[C_k]$ of P_j using the set $\mathcal{M}_{i,j}$ of mapping tables
   – OLD_VAL=OLD_VAL $\cup\{\text{OLD}_k\}$; NEW_VAL=NEW_VAL $\cup\{\text{NEW}_k\}$; COND_INST=COND_INST $\cup\ \{\tau[C_k]\}$
   iii. Send OLD_VAL, NEW_VAL, and COND_INST to P_j
   iv. WAIT ($\langle msg_1, \cdots, msg_{|\mathcal{T}|}\rangle$)
   for each msg_i do:
   if $msg_i = true$, execute the local $\tau[A_i]'s$, for each remote peer P_k involved in $\tau[A]$ do:
   for each msg_i do:
   if $msg_i = true$, then for each $\tau[A_k]$ translate it into $\tau'[A_k]$ using the mapping tables $\mathcal{M}_{i,k}$, and execute $\tau'[A_k]$

Fig. 4. Execution of BEFORE Triggers

minimization of inter-peer messages. Due to lack of space, we will not go into implementation optimizations.

5.2 Detached AFTER Triggers

The execution semantics described till this point are valid in cases that all peer involved in a trigger are concurrently online. But what happens if one of them is offline when a trigger has to be executed? In order to be compatible with the immediate coupling modes of the centralized execution semantics of SQL3 triggers, we have to restrict the execution of distributed triggers only in periods where all involved peers are online. However, such a tactic would be totally inefficient and contradictory to the spirit of a P2P environment, where the offline state of peers is considered normal and expected. Lets consider again Trigger 1: we would like Dr F's database to be updated with all the tests of his patients performed by Dr S, even though some of these tests may take place when the Dr_F_DB is offline. Reasonably, we expect that when Dr_F_DB comes online and connects to Dr_S_DB the former is updated with tests that were inserted in the latter during the offline period. In order to support such data management appropriate to the P2P database setting, we introduce a variation of AFTER triggers, characterized as DETACHED AFTER triggers. These ones are the triggers that should be considered for execution both when peers are on- and off-line. These triggers are treated as AFTER ones when all involved peers are available. However, as soon as an involved peer

PROCESS_AFTER_TRIGGERS(C)

1–3. Proceed in a way similar to Steps 1–3 of the semantics of BEFORE triggers, with sets ATs and $\mathcal{R}_A$ instead of BTs and $\mathcal{R}_B$.
4. For each $\tau \in$ ATs in order of trigger creation, do:
   (a) If τ is a statement-level trigger, then
   if $\tau[C]$ is local, then do:
   – send it to the local evaluator
   – WAIT(*Message*)
   if $\tau[C]$ is remote (in peer P_j) then do:
   – Translate the instance of $\tau[C]$ to an instance $\tau'[C]$ of P_j using the set $\mathcal{M}_{i,j}$ of mapping tables
   – Send $\tau'[C]$ to P_j for evaluation
   – WAIT(*Message*)
   if *Message* = *true*, then first call EXECUTE_STATEMENT($\tau[A_i]'s$), and for each $\tau[A_k]$ in remote peer P_k, translate it to $\tau'[A_k]$ using the mapping tables $\mathcal{M}_{i,k}$, and call EXECUTE_STATEMENT($\tau'[A_k]$)
   (b) If τ is a row-level trigger, then:
   if $\tau[C]$ is local, for each $tr \in \mathcal{T}$, do:
   i. Set values for OLD and NEW from tr
   ii. If $\tau[C]$ evaluates to true, first call EXECUTE_STATEMENT($\tau[A_i]$), then call EXECUTE_STATEMENT($\tau[A_l]$), for all l such that $l \neq i$
   if $\tau[C]$ is remote (in peer P_j), then, for each $tr \in \mathcal{T}$, do:
   i. Set values for OLD and NEW from tr
   ii. Translate the instance of $\tau[C]$ to an instance $\tau'[C]$ of P_j using the set $\mathcal{M}_{i,j}$ of mapping tables
   iii. Send OLD, NEW, and $\tau'[C]$ to P_j
   iv. WAIT (*Message*)
   v. If *Message* = *true*, then do:
   – Translate $\tau[A_i]$ into $\tau'[A_i]$ using the mapping tables $\mathcal{M}_{i,j}$, and call EXECUTE_STATEMENT($\tau[A_i]$)
   – for each remote $\tau[A_k]$ in $\tau[A]$, translate it into $\tau'[A_k]$ using the mapping tables $\mathcal{M}_{i,k}$, and call EXECUTE_STATEMENT($\tau'[A_k]$)

Fig. 5. Execution of AFTER Triggers

goes offline the coordinator assigns to them different execution semantics, in order for them to continue to perform data coordination. The DETACHED keyword is optional, because not all AFTER triggers should be executed while peers are disconnected.

Rationally, Trigger 2 should be executed in an immediate mode: Dr F would not like his prescription to be 'hanging' if P_DB is not online when the trigger is fired. Moreover, in this case, if the trigger was executed in an asynchronous mode while peers are offline and come online again, it is most possible that the patient would receive the prescribed medicine with a long delay. In the case that P_DB is not available, Dr F would probably prefer to contact another pharmacist for the prescription or give the prescription to the patient. It is obvious that there are cases of AFTER triggers for which the user would like to have control of the outcome of the trigger execution. Thus, the DETACHED execution mode is optional and user-decidable.

Note that we propose this option only for AFTER triggers and not for BEFORE ones. The reason is that, semantically, BEFORE triggers are used by SQL3 in order to implement security and constraint violation control over the database. Thus, offering the option of detached execution of parts of the BEFORE trigger, would not serve this goal. Figure 6 shows the execution semantics of DETACHED AFTER triggers in case that at least one involved peer is offline and later on comes online again. We introduce the set of DETACHED AFTER triggers $\mathcal{R}_{DA} \subseteq \mathcal{R}_A$ and we consider DAT to be the activated enabled subset of them.

Trigger 2. *Suppose that when Dr F is prescribing a medicine to one of his patients, he wants this prescription to be inserted both in his own database and in the database of*

PROCESS_DETACHED_AFTER_TRIGGERS(C)

1–3. Similar to Steps 1–3 in AFTER triggers
4. For each $\tau \in$DATs in order of trigger creation, do:
   (a) If τ is a statement-level trigger, then
   if $\tau[C]$ is local, then do:
   - send it to the local evaluator
   - WAIT(*Message*)
   - if $Message = true$, execute local $\tau[A_i]'s$ and for each $\tau[A_k]$ in remote peer P_k, translate it into $\tau'[A_k]$ using $\mathcal{M}_{i,k}$, and put $\tau'[A_k]$ in the marker $M_{P_i}(P_k)$. Send the markers when peers are online.

   if $\tau[C]$ is in remote peer P_j, translate the instance of $\tau[C]$ to an instance $\tau'[C]$ of P_j using the set $\mathcal{M}_{i,j}$ of mapping tables and:
   - if P_j is online:
     - Send $\tau'[C]$ to P_j for evaluation
     - WAIT(*Message*)
     - if $Message = true$, execute local $\tau[A_i]'s$, and for each $\tau[A_k]$ in P_k, translate it into $\tau'[A_k]$ using $\mathcal{M}_{i,k}$, and put $\tau'[A_k]$ in the marker $M_{P_i}(P_k)$. Send the marker when P_j is online
   - else (P_j is offline)
     - put $\tau'[C]$ in $M_{P_i}(P_j)$
     - WAIT(*Message*)
     - if $Message = true$, execute local $\tau[A_i]'s$ and for each $\tau[A_k]$ in P_k, translate it into $\tau'[A_k]$ using $\mathcal{M}_{i,k}$, and put $\tau'[A_k]$ in $M_{P_i}(P_k)$. Send the markers when peers are online

   (b) If τ is a row-level trigger, then proceed in an analogous way as for statement-level triggers

Fig. 6. DETACHED AFTER execution

the pharmacist, P_DB, so that the latter can sent the medicine to the patients residence. The following trigger declares this task:

CREATE TRIGGER prescriptionInsertion
AFTER INSERT ON Dr_F_Prescription
REFERENCING NEW AS NewPresc IN Dr_F_DB
FOR EACH ROW
BEGIN INSERT INTO Dr_P_Prescription VALUES NewPresc IN Dr_F_DB END

Because of lack of space Figure 6 describes the offline processing of statement-level DETACHED AFTER triggers. Row-level are processed in an analogous way, performing the appropriate processing of NEW and OLD variables and condition instances, in a similar way with Figure 5. As we can see, the coordinator of a DETACHED AFTER trigger executes all the local parts of the latter and saves to the appropriate markers remote condition and action instances. The marker is a data structure used to collect processed data from distributed triggers that are evaluated while peers are offline; the formal definition is in Section 6. In case that the condition is local, the trigger instance is executed (up to the execution of local actions) and is removed from the trigger processor.If the condition is local, then the local actions can and should be executed right away. In this manner the execution semantics of the distributed trigger are reduced to the centralized SQL3 execution semantics for the local part of the trigger - which is actually the part that can be executed while the peer is offline. The remote actions are stored in the appropriate markers which are sent to the respective peers when possible. If the condition is on a remote peer and there is no current connection between the latter and the coordinator, then the condition instance is logged in the appropriate marker and is sent to the remote peer when possible.

The main difference between the online and the offline state of a peer is that the mapping tables and the coordination triggers are enabled and disabled, respectively. While the mapping tables are enabled, the peer uses them in order to translate queries and

triggers to the exported schema of its acquaintees. Accordingly, while triggers are enabled, the peer considers them eligible for firing when local events occur. The DETACHED AFTER triggers are a special case which, while they are disabled, are consider by the peer for 'local' firing. More specifically, the peer considers for firing the local part of a disabled DETACHED AFTER trigger and logs processed data (i.e. remote condition and remote action instances) to send out to the involved acquaintees when both those involved acquaintees and the peer are online again.

Moreover, note that, for each DETACHED AFTER trigger instance, the marker logs the instances of the event, and, if the condition is local, it also logs the condition instance and instances of the remote actions to be executed later. The reason is that if the condition is local, then the local actions can and should be executed right away. In such way the distributed execution semantics of the distributed trigger are reduced to the centralized SQL3 execution semantics for the local part of the trigger - which is actually the part that can be executed while the peer is offline.

5.3 Termination Protocol

This section considers a protocol for terminating an executing a trigger when it is not possible to finish the execution of the trigger according to the semantics presented in section 5.1. We consider timeouts and disconnections during trigger execution by using finite state machines that represent the coordinator and the participants. Figure 7 depicts the finite state machine used for the coordinator and participants in the presentation of our termination protocols. The horizontal bars labeling the transitions have the input message at the top and the output message at the bottom.

For simplicity, assume a trigger with a condition destined to one single remote peer and several actions, some of which are destined to remote peers. The execution of the trigger starts in the EVENT state. When an event is signaled, trigger execution enters the COND state after sending the conditions to the condition evaluator. If the coordinator receives a "false" message from the evaluator, trigger execution aborts; otherwise, the execution enters the LOCAL state after sending the local actions to the local action executor. When the trigger execution receives an acknowledgement message from the local action executor, it then enters the REMOTE state, after sending the remote actions to the appropriate remote peers. After all the acknowledgements have comed back to the execution module, the latter can finally stop.

Finite state machines for participants are very simple: They all have a START state and an END state. A condition evaluator moves from the START state to the END state after receiving and evaluating the condition, and sending the evaluation result back to the trigger execution module. Similarly, the action executor moves from the START state to the END state after receiving and executing the action and sending an acknowledgement message back to the trigger execution module.

Coordinator Termination. If the coordinator is in EVENT state and either the WAIT command timeouts or it receives a disconnection message from a participant which is supposed to evaluate the condition, then it drops the executing trigger instance. If the latter is a DETACHED AFTER trigger, it re-initiates the execution of the same (dropped) trigger instance with DETACHED execution semantics. If the coordinator is

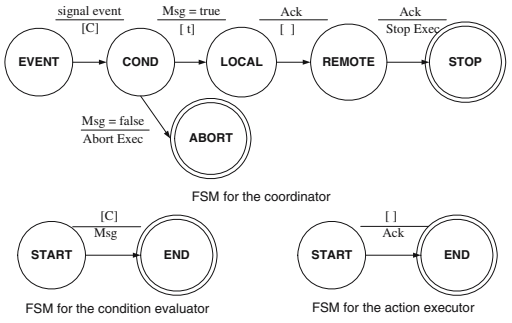


Fig. 7. FSMs for Trigger execution

in CONDITION state, it continues the execution of the current trigger instance, and, in the case of a DETACHED trigger, it saves the actions to be executed in the disconnected participant in the respective marker. Overall, in case of disconnection of any participant, the coordinator switches the execution semantics of AFTER triggers to the DETACHED mode and discards BEFORE or plain AFTER trigger instances.

Participant Termination. Notice that the participants are of two kinds, namely condition evaluators and action executors.

Evaluator. The condition evaluators can timeout in the START state. In this state, the participant expects the condition from the coordinator. Should there be a timeout, this means that the coordinator must have failed in the EVENT state. The participant can stop at this state and exit. Any condition message arriving latter can be ignored, in which case the coordinator would timeout in the state COND. This means that, in case of a timeout, the participant just discards the condition instance and does nothing else, since the coordinator will take care of the termination of the executing trigger instance.

Action Executor. The action executor also can timeout in the START state. Similarly, the participant expects a message containing the action from the coordinator. If there is a timeout, this means that the coordinator must have failed in the LOCAL state in case it is the local action executor; or the coordinator must have failed in the REMOTE state in case the executor is remote. The action executor can stop at this state and exit. Any action message arriving latter can be ignored. In this case, the coordinator would timeout either in the LOCAL state or in the REMOTE state.

6 Acquaintance Protocols

To join a network, a peer P_i must explicitly establish an acquaintance with a known peer, say P_j, that already belongs to the network. At the time it is established, an acquaintance is associated with a set of mapping tables which constrain the exchange of data between peers, and a set of coordination rules (written in triggers language described in this paper) which guide such an exchange.

To leave a network, a peer P_i must drop its set of acquaintances with known peers, say $P_{j_1}, \cdots, P_{j_l}$. The various constraints attached to the acquaintance are abolished when

the latter is dropped. Establishing and abolishing acquaintances have to do with the creation and the drop, respectively, of coordination rules and constraints for data exchange. These procedures have to be performed such that we can guarantee that no local parts of coordination tasks are 'hanging' or delayed because of peer communication problems.

In the following we present acquaintance protocols that are independent, i.e. peers restore their state in the P2P system without the need of consulting other peers.

6.1 Acquaintance Tracking and Peer Markers

To deal with the transient character of peers, we introduce the idea of **acquaintance tracking**, which is similar to the idea of "dependency tracking" mentioned in [19] to single out triggers that mention tables that meanwhile have been dropped. Acquaintance tracking is meant to single out triggers that mention acquainted peers that have left or disconnected from the network.

Definition 3 (Acquaintance Tracking). *Suppose that there is a peer $P_i \in \mathcal{P}$. Let $\mathcal{P}_i \subseteq \mathcal{P}$ be the set such that for all peers $P_j \in \mathcal{P}_i$, $(P_i, P_j) \in \mathcal{L}$ and P_i has at least one trigger that mentions tables of ES_j. Then finding such a set $\mathcal{P}_i \subseteq \mathcal{P}$, the* tracked set*, is called* acquaintance tracking *of peer P_i with respect to its triggers.*

Intuitively, Definition 3 means to find all the acquaintees of P_i for which P_i has triggers mentioning tables of them. In addition to the tracking acquaintances, our protocols use an extension of the idea of **peer marker** mentioned in [9].

Definition 4 (Marker). *Suppose that the acquaintance tracked set of peer P_i is $\mathcal{P}_i$. For each peer $P_j \in \mathcal{P}_i$, P_i may define a marker $M_{P_i}(P_j)$. Suppose that $\tau(P_j) \in P_i$ are the triggers τ defined in P_i that mention tables in ES_j. Let $\tau'(P_j) \subseteq \tau(P_j)$ be the subset of triggers defined in P_i that mention tables from ES_j in the event part. For each $\tau'_m \in \tau'(P_j)$ we define an empty set $I_{\tau'_m} = \{\}$. Then $M_{P_i}(P_j) = \{I_{\tau'_m} | \tau'_m \in \tau'(P_j)\}$.*

A peer can perform the following actions that concern its status in the P2P network:

Establishing Acquaintances. Assume that a peer P_i wants to establish an acquaintance with peer P_j. P_i uses the following algorithm.

1. P_i generates and populates mapping tables $\mathcal{M}_{i,j}$.
2. P_i sends copies of instances of $\mathcal{M}_{i,j}$ to P_j.
3. Both P_i and P_j enable the $\mathcal{M}_{i,j}$.
4. Both P_i and P_j generate a set of initial distributed triggers to coordinate their exchange of data. For each newly defined trigger $\tau \in P$, $(P \in \{P_i, P_j\})$:
   (a) Track acquaintances of P with respect to trigger $\tau \in P$. Let $\mathcal{P}_P$ be the tracked set for P.
   (b) Enable τ if all peers in $\mathcal{P}_P$ are online, and disable τ if at least one peer in $\mathcal{P}_P$ is not online.
   (c) Send τ to the peer that is mentioned in the event if this is not the local database.
5. P_j is added to the list of P_i's acquaintees $\mathcal{A}_{P_i}$, and vice-versa.

Abolishing Acquaintances. To abolish an acquaintance with P_j, P_i executes the following algorithm.

1. Send a message to P_j to delete any mapping tables and coordination triggers over the acquaintance (P_i, P_j) and wait for acknowledgment.
2. Upon receiving the acknowledgment from P_j, delete mapping tables and coordination triggers over the acquaintance (P_i,P_j).
3. P_j is removed from the list of P_i's acquaintees $\mathcal{A}_{P_i}$, and vice-versa.

Connecting to the P2P System. P_i that connects to the P2P system (or, in other words, comes online) executes:

1. Enables all its existing acquaintances, i.e. for each peer $P_j \in \mathcal{A}_{P_i}$, P_i enables the existing respective mapping tables and coordination triggers.
2. For each marker $M_{P_i}(P_j)$, if P_j is connected, P_i sends the marker to the latter.

Disconnecting from the P2P System. A peer P_i that disconnects from the P2P system (or in other words goes offline) disables all its existing acquaintances, i.e. for each peer $P_j \in \mathcal{A}_{P_i}$:

1. P_i and P_j disable the existing respective mapping tables over their acquaintance.
2. if P_i is in the middle of the execution of a trigger τ then terminate it as follows:
   -if P_i is in state EVENT then:
   if the condition is local, continue the execution else:
   (a) for a BEFORE/AFTER trigger discard the instance
   (b) for a DETACHED AFTER trigger log the instance in the respective marker
   -if P_i is in state CONDITION then:
   continue the execution
3. P_i and P_j disable all BEFORE triggers defined in either of them and involve tables in ES_i or ES_j.
4. P_i and P_j disable all AFTER triggers defined in either of them, involve tables in ES_i or ES_j and are not characterized as DETACHED.
5. P_i and P_j exchange markers for DETACHED AFTER triggers defined in either of them and involve tables in ES_i or ES_j.

7 Preliminary Implementation Results

We are implementing a trigger mechanism for the P2P layer of the Hyperion architecture described in Section 1 using the syntax of distributed triggers and the execution semantics described in this work. The mechanism is being implemented using Sun Microsystem's P2P platform JXTA [7] and PostgreSQL DBMS. Yet, in order to test the overall behaviour of the system we implemented a simulation and performed the following experiments.

The experimental setup emulates a master peer that coordinates the execution of distributed triggers among a set of peers. In each test round, the master peer queues triggers to a random number of peers and the peers will execute them. If any of them fail, the master rolls back the transaction. We measure the time by taking the following factors into account: (1) number of peers in the overlay; (2) number of transactions; (3) transmission cost of a message (a.k.a, network cost); (4) percentage of triggered peers for each transaction; (5) cost of processing a trigger; (6) possibility of rolling back a transaction; (7) and the cost of rolling back a trigger execution.

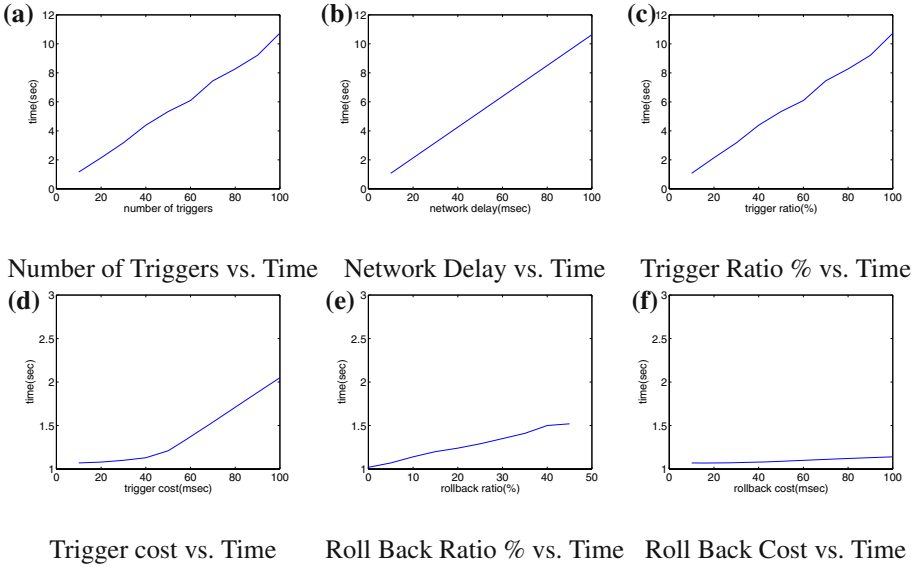


Fig. 8. Experiments on the Execution of Distributed Triggers

In Figure 8 we present some of the major graphs. Figures 8(a), (b) show the amount of time needed for the execution of triggers depending on the total number of signaled triggers and the network delay, respectively. As expected, the amount of time is proportional to these metrics. Figure 8(c) shows that the time is proportional to the trigger ratio executing in each peer separately (i.e. the ratio of local triggers that are signaled).

Furthermore, Figure 8(d) shows the amount of trigger execution time depending on the processing cost of the triggers, i.e. the complexity of the triggers and how costly it is to evaluate them. For low trigger costs the execution time increases with a low rate. This is because trigger evaluation is distributed so their execution cost is distributed too, and, thus, does not influence the total execution time significantly. However, for high trigger cost values, the total amount of execution time increases more; the reason is that for complex triggers, the master peer has to do a lot of processing that cannot be distributed. Figure 8(e) tests how rolling back triggers influences the execution time. Obviously, execution time increases with the increment of the percentage of triggers rolled back. Yet, the increase in time is low, since rolling back triggers is a distributed procedure. Finally, Figure 8(f) shows how the cost of rolling back triggers influences execution time. The results prove that execution time is not really influenced. The reason is that the cost of rolling back one trigger does not hold back significantly the overall execution of triggers, since it concerns very few peers.

8 Related Work

In [18], a component-based architecture for an active mechanism is described for federated database systems. The ECA rules executed by the system are global to all the

databases involved in the federation, and their conditions and actions may involve subparts destined to different databases. Oppositely, the globality of our triggers is limited to acquainted peers. We do not permit a peer to define triggers that involve further peers that are not acquainted wit the former. Unlike [18], we rather assume that peers have autonomous schemas and instances than a common universe of discourse.

In [15], an approach for managing consistency of interdependent data in a multidatabase environment is presented. Among others, the approach proposes a mechanism for maintaining mutual consistency of these data. User specified requirements allow consistency to be violated up to a specified level. Whenever it reaches the limits, transactions are executed at the target data to restore an acceptable degree of consistency.

The work in [3] gives a method for specifying inter-database dependencies in a federated database system and an execution model. The authors mix the execution quasi-transaction model [6], during which events are generated, with the execution of ECA rules. Unlike in [3],we do not assume (for now) an advanced transaction model that spans the peer databases involved in a P2P network. We rather assume that each peer executes its updates as a separate transaction. The only coordination we enforce among the peers is to first execute updates destined to the local peer in order to avoid sending remote updates when the local transaction aborts.

Our triggers are a first step toward an SQL3-based extension of a rule language for multidatabase coordination presented in [10]. Unlike the work in [10], however, the work reported in this paper deals with the transient character of peers, and extends the standard execution semantics of SQL3.

9 Conclusions

We have proposed an SQL3-based language for distributed triggers to be used for coordinating data exchange among peer databases. We provided a processing mechanism for these triggers by extending the standard execution semantics of SQL3 in two ways. We first presented distributed procedures for processing the different classes of triggers identified for the P2P setting by taking into account the heterogeneity of peers. Then we described protocols for managing the acquaintances of peers to accommodate peer transiency during rule processing. Also, we are implementing the proposed trigger language for the P2P layer of the Hyperion architecture, and we showed some preliminary simulation results.

References

[1] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison Wesley, Readind, MA, 1995.

[2] Marcelo Arenas, Vasiliki Kantere, Anastasios Kementsietsidis, Iluju Kiringa, Rene J. Miller, and John Mylopoulos. The hyperion project: from data integration to data coordination. *SIGMOD Record*, 32(3):53–58, 2003.

[3] R. Arizio, B. Bomitali, M.L. Demarie, A. Limongiello, and Mussa. P.L. Managing interdatabase dependencies with rules + quasi-transactions. In *Third International Workshop on Research Issues in Data Engineering: Interoperability in Multidatabase Systems*, pages 34–41, Vienna, April 1993.

[4] R. Cochrane, H. Pirahesh, and N. Mattos. Integrating triggers and declarative constraints in sql database systems. In *VLDB*, pages 567–578, 1996.

[5] E.F. Codd. A relational model of data for large shared data banks. *Communications of the ACM*, 13(6):377–387, 1970.

[6] A. Elmagarmid, M. Rusinkiewicz, and A. Sheth. *Management of Heterogeneous and Autonomous Database Systems*. Morgan Kaufmann Publishers, 1999.

[7] J. Gradecki and J. Gradecki. *Mastering JXTA: Building Java Peer-to-Peer Applications*. Wiley, 2002.

[8] V. Kantere. A rule mechanism for p2p data management. Technical report, University of Toronto, 2003. CSRG-469.

[9] Vasiliki Kantere, Iluju Kiringa, John Mylopoulos, Anastasios Kementsietsidis, and Marcelo Arenas. coordinating peer databases using ECA rules. In *DBISP2P*, 2003.

[10] Vasiliki Kantere, John Mylopoulos, and Iluju Kiringa. A Distributed Rule Mechanism for Multidatabase Systems. In *Proceedings of COOPIS*, 2003.

[11] Anastasios Kementsietsidis, Marcelo Arenas, and Renée. J. Miller. Data mapping in peer-to-peer systems: Semantics and algorithmic issues. In *sigmod*, 2003.

[12] K. Kulkarni, N. Mattos, and R. Cochrane. Active database features in sql-3. In N. Paton, editor, *Active Rules in Database Systems*, pages 197–219. Springer Verlag, 1999.

[13] Beng Chin Ooi, Yanfeng Shu, and Kian-Lee Tan. Relational data sharing in peer-based data management systems. *SIGMOD Record*, 32(3):59–64, 2003.

[14] M. T. Özsu and P. Valduriez. *Principles of Distributed Database Systems*. Prentice Hall, Upper Saddle River, 2 edition, 1999.

[15] A. Sheth and M. Rusinkiewicz. Management of Interdependent Data: Specifying Dependency and Consistency Requirements. In *Proc. of the Workshop on the Management of Replicated Data*, Houston, TX, November 1990.

[16] A. P. Sheth and J. A. Larson. Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases. *ACM Computing Surveys*, 22(3):183–236, 1990.

[17] Igor Tatarinov, Zachary G. Ives, Jayant Madhavan, Alon Y. Halevy, Dan Suciu, Nilesh N. Dalvi, Xin Dong, Yana Kadiyska, Gerome Miklau, and Peter Mork. The Piazza peer data management project. *SIGMOD Record*, 32(3):47–52, 2003.

[18] G. Vargas-Solar, C. Collet, and H.G. Ribeiro. Active Services for Federated Databases. In *ACM Symposium on Applied computing*, pages 356–360, Como, Italy, 2000.

[19] J. Widom and F. Finkelstein. Set-oriented production rules in relational database systems. In H. Garcia-Molina and H.V. Jagadish, editors, *Proceedings of the 1990 ACM SIGMOD International Conference on Management of Data*, pages 259–270, 1990.

Satisfaction-Based Query Load Balancing*

Jorge-Arnulfo Quiané-Ruiz**, Philippe Lamarre, and Patrick Valduriez

INRIA and LINA
Université de Nantes
2 rue de la houssinière, 44322 Nantes Cedex 3, France
{Jorge.Quiane, Philippe.Lamarre}@univ-nantes.fr,
Patrick.Valduriez@inria.fr

Abstract. We consider the query allocation problem in open and large distributed information systems. Provider sources are heterogeneous, autonomous, and have finite capacity to perform queries. A main objective in query allocation is to obtain good response time. Most of the work towards this objective has dealt with finding the most efficient providers. But little attention has been paid to satisfy the providers interest in performing certain queries. In this paper, we address both sides of the problem. We propose a query allocation approach which allows providers to express their intention to perform queries based on their preference and satisfaction. We compare our approach to both query load balancing and economic approaches. The experimentation results show that our approach yields high efficiency while supporting the providers' preferences in adequacy with the query load. Also, we show that our approach guarantees interesting queries to providers even under low arrival query rates. In the context of open distributed systems, our approach outperforms traditional query load balancing approaches as it encourages providers to stay in the system, thus preserving the full system capacity.

1 Introduction

We consider dynamic distributed systems, providing access to large number of heterogeneous and autonomous information sources. We assume that information sources play basically two roles: consumers that generate requests (we will indifferently use the terms request and query throughout this paper) and providers which perform requests and generate informational answers.

Providers can be heterogeneous in terms of capacity, competence and data. Heterogeneous capacity means that some providers are more powerful than others and can treat more requests per unit time. Heterogeneous competence means that some providers may treat some query types that others cannot, and vice versa. Data heterogeneity means that requests are performed differently by different providers, i.e. the same request performed by different providers may

* Work partially funded by ARA "Massive Data" of the French ministry of research (projects MDP2P and Respire) and the European Strep Grid4All project.

** This author is supported by the Mexican National Council for Science and Technology (CONACyT).

R. Meersman, Z. Tari et al. (Eds.): OTM 2006, LNCS 4275, pp. 36–53, 2006.

return different results. We also consider that requests are heterogeneous, i.e., some requests consume more providers' resources than others.

Providers, on the other hand, are autonomous over their resources and data management. Thus, they can express their preferences to perform queries through an *intention* value. Such preferences may represent, for example, their strategies, their topics of interest, or the combination of both of them. The providers' *intentions* might be the result of merging the *preferences* with others factors, such as the query load or reputation/quality of the consumers. Providers' *preferences* are rather static (i.e. long-term) and do not change much while their *intentions* are more dynamic (i.e. short-term).

One of the problems that has been thoroughly investigated in the area of query allocation is *query load balancing* (*QLB*). The main objective of *QLB* is to maximize overall system performance (throughput and response times) by balancing the query load among providers. However, even if performance is very good, providers may be not satisfied with the system and may leave it. Thus, the system should fulfill providers' expectations in order to preserve full system capacity.

This problem is quite important in dynamic information systems where providers can leave the system at will. When a provider is no longer satisfied with the system, the only way to express *unsatisfaction* it is to leave. In order to achieve stability, our goal is to maximize performance while ensuring over time that providers are satisfied enough to stay in the system. Taking into account load balancing, providers' preferences and providers' satisfaction in the allocation of requests in distributed systems composed of autonomous component systems is a very important idea. It is particularly timely with the potential profusion of software based on web services in particular and on a services oriented to architecture in general.

In this paper, in order to address this problem, we propose a *QLB* approach which allows providers to express their intention and take care of their *satisfaction*. We also develop a model that allows providers to know whether the system is fulfilling their expectations. We provide an experimental validation which compares our approach with both query load balancing and economical approaches. The experimental results show that our approach yields high efficiency while supporting the providers' preferences in adequacy with the query load.

The rest of the paper is organized as follows. Section 2 gives a motivating scenario. Section 3 defines precisely the problem we address. Section 4 presents our *QLB* approach. Section 5 defines the metrics used for validating our approach. Section 6 presents our experimental validation. Section 7 discusses related work. Finally, Section 8 concludes.

2 Motivating Scenario

We, here, illustrate the problem that we consider in this paper by means of a healthcare application example. Consider a large distributed information system

Table 1. Providers set that are able to deal with the request

Providers	Utilization	Preference
p_1	0.15	No
p_2	0.43	No
p_3	0.78	Yes
p_4	0.85	No
p_5	1.1	Yes

gathering thousands of medical doctors with the goal of sharing their patients information. By hospital policies or simply information privacy, providers (i.e. the medical doctors) have preferences to share information. Thus, each provider locally stores its preferences and may change them at any time. In fact, any system with competitive companies meets this schema.

Now, consider a simple scenario where a medical doctor (the consumer), who has received a patient with a new skin disease, requests the system for similar symptoms of the illness. Clearly, she specifies some specific parameters, such as the hospital where she works and the illness symptoms. Suppose that in order to better understand the skin disease she asks for two results. That is, the system have to allocate the request to two different providers.

As first job, the system needs to identify the providers that are capable to deal with the request. This can be done using a matchmaking technique (for example [17]). As second job, the system must obtain their availability and preference for dealing with such a request. This can be done following the architecture in [8] for example.

Assume, then, that the resulting list contains 5 providers with their utilization and preferences (see Table 1). Assume that the respective medical doctors (column 1) of this list in ascending utilization order (column 2) are: p_1, p_2, p_3, p_4, and p_5. Assume that p_1, p_2, and p_4 are not interested on serving the request (column 3) for their own reasons.

As final job, the system needs to allocate the request to the two most capable providers, such that the providers' preference is respected. Current *QLB* approaches[1] would fail in such a scenario since neither p_1 nor p_2 want to deal with the request. The only two options that satisfy the providers' preference are p_3 and p_5, but allocating the query to them may hurt response time.

This example illustrates the conflict between providers' preference and utilization in query allocation. However, considering allocation alone is not very meaningful. What is more important is that providers be globally satisfied with the allocation process, even though they are sometimes overloaded or do not get queries they want. This can be checked by making regular assessment over some k last queries.

The entire treatment of this scenario encompasses different aspects. First, query planning processes may be required. This problem is addressed in different ways in the literature [19]. We do not consider it in this paper, and we can

[1] Whose aim is to allocate queries to the less utilized providers.

indifferently assume that it is done by the consumer or any other site. Second, the system must support matchmaking techniques in order to find the relevant providers for performing requests. Such matchmaking techniques have been proposed by several groups [9,12,17]. So, we simply assume there exists one in our system. In addition, we do not consider either the way in which providers obtain their preferences values since it is out of the scope of this paper and orthogonal to the query allocation problem.

3 Satisfactory Query Load Balancing Problem

In this section, we make precise the problem we consider. The system consists of a set of consumers $\boldsymbol{C}$ and of a set of providers $\boldsymbol{P}$ which can join and leave the system at will. It is possible to have $\boldsymbol{C} \cap \boldsymbol{P} \neq \emptyset$. Consumers issue queries in a *tuple* format $q = < c, t, d, n >$ such that, $q.c \in \boldsymbol{C}$ is the identifier of the consumer that has issued the query, $q.t \in \boldsymbol{T}$ is the query type and $\boldsymbol{T}$ the set of query types that the system can support, $q.d$ is the description of the task to be realized, and $q.n \in \mathbb{N}^*$ is the number of providers to which the consumer wishes to allocate its query.

Definition 1. ***Feasible Query***
Let $T_p \subseteq \boldsymbol{T}$ be the query types that the provider $p \in \boldsymbol{P}$ can treat. Given a query q with $q.t \in \boldsymbol{T}$, issued by the consumer $q.c \in \boldsymbol{C}$, let P_q denote the set of providers which can deal with q, where $P_q = \{p : (p \in \boldsymbol{P} \backslash \{q.c\}) \wedge (q.t \in T_p)\}$. Then, a query q is said to be feasible if and only if $P_q \neq \emptyset$.

We only consider feasible queries which are defined in Definition 1 and assume that providers do not request the system for information that they have[2]. Each feasible query q has a ***cost***, $cost_p(q) > 0$, that represents the treatment units that q consumes at p. Query allocation of some feasible query q among the providers which are able to deal with q is defined as a vector (see Definition 2).

Definition 2. ***Query Allocation***
Allocation of query q amongst the providers in P_q is a vector[3] $All\overrightarrow{oc}$ of length $||P_q||$ such that: $\forall p \in P_q$,

$$All\overrightarrow{oc}[p] = \begin{cases} 1 & \text{if provider } p \text{ gets the query} \\ 0 & \text{otherwise} \end{cases}$$

$$\text{with } \sum_{p \in P_q} All\overrightarrow{oc}[p] \leq q.n$$

In the following, the set of provider such that $All\overrightarrow{oc}[p] = 1$ will also be noted $\widehat{P_q}$. Each provider $p \in \boldsymbol{P}$ has a finite *capacity*[4], $cap_p > 0$, for performing feasible queries. We then define the provider's utilization as in Definition 3.

[2] Notice that without lost of generality one cannot assume this.
[3] $All\overrightarrow{oc}_q$ when there is an ambiguity on q.
[4] Capacity means the number of treatment units that a provider can have per time unit.

Definition 3. ***Provider Utilization***
$U_t(p)$ denotes the utilization of a given provider $p \in \boldsymbol{P}$ at time t, which is the capacity portion utilized by the queries that p is treating

$$U_t(p) = \frac{\sum_{q \in Q_p} cost_p(q)}{cap_p}$$

where Q_p denotes the set of queries that have been allocated to p but have not already been treated at time t (i.e. the pending queries).

Providers are free to express their *preferences*[5] for performing each arrival feasible query q. However, since such providers' *preferences* are considered as private information, providers hide their *preferences* by means of an *intention* value denoted by $I_p(q) \in]-\infty, 1]$. Expressing *intention* may be the result of merging, for example, their preferences, utilization, and satisfaction. We formalize this notion in Section 4.2. If the *intention* value is positive, the greater it is, the greater the desire for performing queries. If the *intention* value is negative, the smaller it is, the greater the refusal for performing queries. Providers' refusal can go down to $-\infty$ because their utilization can grow up, theoretically, to $+\infty$.

Given this, the overall aim of the system is to allocate each feasible query q of a set of incoming feasible queries $\boldsymbol{Q}$ to providers in P_q by taking into account the providers' *utilization* and *preferences* in a equitable way for providers and the system in general.

4 Satisfaction-Based Query Allocation

In this section, we present our query allocation approach based on satisfaction, called *Satisfaction-based Query Load Balancing* S_bQLB. We first present the providers characteristics on which S_bQLB is based and then present the S_bQLB approach.

4.1 Providers Characterization

We present, in this section, what the providers can perceive from the system. We focus on three principal characteristics: the satisfaction with the system (*Satisfaction*), their adequation to the system (*Adequation*), and their satisfaction with the query allocation method (*Allocation Satisfaction*).

Let us first introduce some notations to describe how a provider can perceive the system. We assume that each provider maintains a vector $\overrightarrow{PP}^k_p$ of its k last preferences for performing the queries that have been proposed to it by the system[6]. This set of proposed queries is noted PQ^k_p. By convention, $\forall q \in$

[5] Whose values are between -1 and 1, where 1 means the greatest interest and -1 the greatest refusal.

[6] Note that k may be different for each provider depending on it's storage capacity, or strategy. For the sake of simplicity, we have assumed here that they all use the same k.

PQ_p^k, $\overrightarrow{PP}_p^k[q] \in [-1..1]$. Finally, SQ_p^k denotes the set of queries that have been served by p among PQ_p^k ($SQ_p^k \subseteq PQ_p^k$).

Satisfaction. The *satisfaction.* of providers denotes the satisfaction degree that the providers have from their obtained queries. That is, if they serve queries that they want in general. This notion allows providers to know if they are accomplishing their objectives in the system. We define the provider's *satisfaction* as the average of the preferences that a provider $p \in P$ had to their received queries among the k last incoming feasible queries. In other words,

$$\delta_s(p) = \begin{cases} \left(\frac{1}{2||SQ_p^k||} \sum_{q \in SQ_p^k} \overrightarrow{PP}_p^k[q]\right) + \frac{1}{2} & if\ ||SQ_p^k|| > 0 \\ 0 & otherwise \end{cases} \quad (1)$$

Notice that for a better representation of the *satisfaction* notion we translate the *satisfaction* function's values to [0..1]. This is why $||SQ_p^k||$ is multiplied by 2 and 1/2 is added to the average result.

Adequation. The *Adequation* characteristic denotes the adequation degree of a provider to the system. It is based on the preferences that the provider had towards all the feasible queries proposed by the system. In other words, a provider is adequate to the system whether it receives interesting queries from the system. This notion can only be used by providers in those information systems where they can see feasible queries to pass even if they do not finally serve them. This is the case of our utilized system architecture [8]. We define the *adequation* of a given provider p as the average of its k last preferences for performing queries,

$$\delta_a(p) = \left(\frac{1}{2k} \sum_{q \in PQ_p^k} \overrightarrow{PP}_p^k[q]\right) + \frac{1}{2} \quad (2)$$

Similar to the *satisfaction* function, we translate the *adequation* function's values to [0..1] for a better representation of this notion.

Allocation Satisfaction. The *allocation satisfaction* denotes the satisfaction degree that a provider has from the query allocation method. It allows a provider to know if the allocation method is doing a good job for it. That is, if the system tries to give them, in average, interesting feasible queries. We, thus, define the *allocation satisfaction* of a given provider p, $\delta_{as}(p)$, as the quotient of its *satisfaction* divided by its *adequation*,

$$\delta_{as}(p) = \frac{\delta_s(p)}{\delta_a(p)} \quad (3)$$

Thus, the greater the provider satisfaction value with respect to the provider adequation value is, the more satisfied the provider is from the mediation process.

Algorithm 1. *providersSelection*$(q,\ k,\ P_q)$

1: **begin**
2: **foreach** p **in** P_q **do**
3: ask for the provider's intention $I_p(q)$
4: put the $I_p(q)$ value into $\overrightarrow{I}^q$
5: **done**
6: compute the providers' intention vector ranking $\overrightarrow{R}^q$
7: $providersSelection \leftarrow \overrightarrow{I}^q[\overrightarrow{R}^q[1..min(n, ||P_q||)]]$
8: **end**

4.2 Satisfaction-Based Query Load Balancing

This section details the S_bQLB approach itself for allocating a query q to the $q.n$ providers based on the providers' satisfaction. We focus on the case where requests can be viewed as single units of work called *tasks*. Requests arrive to the system to be allocated to n providers. We assume that a previous matchmaking step has found the set of providers P_q to deal with an incoming query.

Algorithm 1 shows the main steps of the S_bQLB approach for selecting the providers that will treat the queries. These steps are detailed below.

Given an incoming feasible query q and the providers' set P_q, as first step, the S_bQLB asks to each $p \in P_q$ for its intention[7] to deal with q and build a vector $\overrightarrow{I}^q$ containing such intentions values. Providers obtain their intentions as in Definition 4 where, as said so far, such details are considered private information.

Definition 4. *Providers' Intention*
Given a query q the intention of a provider $p \in P_q$ to deal with it is defined as follows,

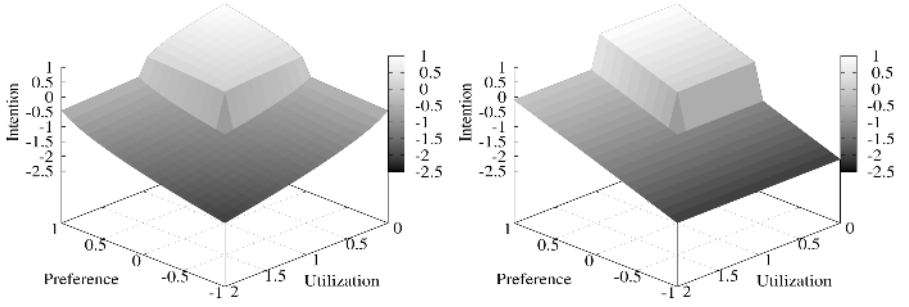
$$I_p(q) = \begin{cases} (pref^{1-\delta_s(p)})(1-U_t(p))^{\delta_s(p)} & if(pref >= 0) \wedge (U_t(p) <= 1) \\ -(((1-pref)+\epsilon)^{1-\delta_s(p)}(U_t(p)+\epsilon)^{\delta_s(p)}) & otherwise \end{cases}$$

with $q.t \in T_p$, $\epsilon = 0.1$ and $pref$ denotes the p's preference for dealing with q.

The way in which providers work out their intentions allows providers to base their intentions principally on their preferences when are not satisfied and on their utilization conversely. For example, assume a simply scenario where two providers $p1$ and $p2$ have the same *preference* 0.1 to perform a request $q1$ and the same *utilization* 0.2. Consider that $p1$ and $p2$ have a *satisfaction* of 0.8 and 0.5 respectively. Thus, in such an example scenario, it is $p1$ which receive $q1$ since its intention is principally based on its utilization while the $p2$'s intention is equally based on its *preference* and *utilization* and its preference towards $q1$ is low.

Figures 1 illustrate the behaviour of the $I_p(q)$ function when providers are satisfied of 0.5 and 0. We can observe that providers show positive intention for

[7] Notice that, considering our system architecture, this operation is realized in local which considerably reduce the network traffic



(a) for a provider's *satisfaction* of 0.5 (b) for a provider's *satisfaction* of 0

Fig. 1. Tradeoff between preference and utilization for getting intention

dealing with queries only when their preferences and utilization are between 0 and 1. This means, that S_bQLB strives to allocate queries to those providers that desire to perform them and that are not overutilized. Note that this allows providers to preserve their preferences while good response times are also ensured to consumers.

On the one hand, we observe that when providers are satisfied of 0.5 (see Figure 1(a)) the providers' *preferences* and *utilization* have the same importance in the way that the providers' *intentions* are obtained. On the other hand, when providers are not satisfied at all (see Figure 1(b)) the providers' utilization has not importance for providers and are their preferences that define their intentions. Conversely, when providers are completely satisfied the utilization defines their intentions.

Going back to the S_bQLB steps, as second step, the S_bQLB approach computes the providers' ranking $\overrightarrow{R}^q$ based on their shown intentions. Such a ranking is introduced for an easiest use of the $\overrightarrow{I}^q$ vector and to enable the selection of the $q.n$ providers to deal with q. Intuitively, $\overrightarrow{R}^q[1]$ is the most interested provider to deal with q, $\overrightarrow{R}^q[2]$ the second, and so on up to $\overrightarrow{R}^q[||P_q||]$ which is the least.

As a result, the S_bQLB returns the $q.n$ best ranked providers in the $\overrightarrow{I}^q$ vector if $q.n <= ||P_q||$, else it returns the $||P_q||$ providers. Note that the providers' intention is the criterion by which S_bQLB chooses the providers.

5 Metrics

We now propose a solution to analyze the *QLB* approaches against the *Satisfactory QLB* problem. To do so, we use two metrics, one for evaluating the ensured *satisfaction-balance* in the system and the other one for evaluating the ensured *query-balance*.

5.1 Satisfaction Balance Metrics

We define the *dissatisfaction* ratio as the ratio between the most and the least satisfied providers in the system. Since we look for providers' satisfaction, we will use the *satisfaction* term instead the *dissatisfaction* one. Obviously, the *satisfaction* ratio is defined as the *dissatisfaction* ratio's inverse.

Before going further, let us say that we have had to define how providers are satisfied by the system, $\delta_s(p)$ (see Equation 1). The following can also be developed for the *adequation* and *allocation satisfaction* functions (Equations 2 and 3), but for space reasons we just develop it for the *satisfaction* function.

A distributed information system ensures a provider *satisfaction-balance* ratio, α, if after each query allocation,

$$\min_{p \in \boldsymbol{P}} \big(\delta_s(p)\big) + c_s \geq \alpha \left(\max_{p \in \boldsymbol{P}} \big(\delta_s(p)\big) + c_s \right)$$

for some fixed constant[8] c_s and where α denotes the desired *satisfaction-balance* ratio in the system. Having said this, we measure the *satisfaction-balance* at a given time t as follows,

$$\gamma = \frac{\min\limits_{p \in P_x} \big(\delta_s(p)\big) + c_s}{\max\limits_{p \in P_x} \big(\delta_s(p)\big) + c_s}$$

γ denotes the factor under which the system is said to be *satisfaction-balanced* at a given time t.

5.2 Query Balance Metrics

As is conventional, we define the imbalance ratio at a given time t as the ratio between the most and the least utilized provider in the system at that time. For better representation and explanation of this property, we use the term *balance* instead of *imbalance*, which is defined as the imbalance ratio's inverse,

$$\min_{p \in \boldsymbol{P}} \big(U_t(p)\big) + c_u \geq \sigma \left(\max_{p \in \boldsymbol{P}} \big(U_t(p)\big) + c_u \right)$$

for some fixed constant[9] c_u and where σ denotes the desired *query-balance* ratio in the system. In order to measure the *query-balance* λ in the system, we first must measure the providers' utilization in the system, as in Definition 3, and then, we proceed to measure λ as follows,

$$\lambda = \frac{\min\limits_{p \in P_x} \big(U_t(p)\big) + c_u}{\max\limits_{p \in P_x} \big(U_t(p)\big) + c_u}$$

[8] Which is set to the minimal satisfaction value that a provider could have with a query, 0.001 for example.

[9] Which is set to the minimal utilization value that a provider may have with a query, 0.001 for example.

We believe that it is quite important that the system also be able to balance all the incoming feasible queries among the relevant providers in average. That is, it must strive to give queries to all the providers in the system if possible, in order to avoid having providers leave the system because of *starvation* and *unsatisfaction*. This is why we introduce the *average query-balance*, λ', metric. We do not need to define this metric since it is symmetrical to the *query-balance* metric. We only define what an *average provider utilization* means in a discrete time interval.

Definition 5. *Provider Average Utilization*
$U_{[t_1, t_2]}(p)$ is defined to be the average of the p's utilization (with $p \in \boldsymbol{P}$) at the time interval $[t_1, t_2]$,

$$U_{[t_1, t_2]}(p) = \frac{\sum_{t_i \in [t_1..t_2]} U_{t_i}(p)}{(t_2 - t_1) + 1}$$

6 Experimental Validation

In this section, we present our experimental evaluation and discuss the results. We have simulated a *mono-mediator* distributed information system with heterogeneous and autonomous provider sources and carried out a series of tests with the objective of assessing *how well the S_bQLB approach operates in autonomous environments*. We first describe the *Capacity Based* and *Economic* approaches against which we compare the S_bQLB algorithm. Next, we describe the simulation setup.

We have conducted three types of evaluations. In the first series of evaluations, we analyze and compare the performance of S_bQLB against *Capacity Based* and *Economic* approaches in considering the *unsatisfaction* departures of providers. The second series focuses on providers' utilization. More precisely, we measure the *QLB* achieved by current *QLB* approaches. We also study, in these experiments, if such approaches really strive to give requests to all providers in the system, i.e. we measure how well they realize the *QLB* in average. Finally, the third series of tests focuses on the providers' satisfactions. We study, here, how well S_bQLB and current *QLB* approaches satisfy the providers, and how well they do it with different query rates.

6.1 Baseline *QLB* Approaches

Capacity Based. In distributed information systems, the *capacity based* [11,15, 18] approach is a well known approach for balancing requests across providers. In such an approach, one common way to allocate queries is choosing the providers that have more available capacities amongst the providers' set that can deal with them. In other words, it sends queries to those relevant providers that are the less utilized. This criterion is defined below.

$$criterion : 1/(1 + U_t(p))$$

Another way is to discard those providers that fail a small constant *utilization* threshold, and then, randomly select the n demanded providers among the remaining ones. However, in practice is not easy to set the utilization threshold. By this fact, we analyze in our experiments the first way for allocating queries.

Economic. Economical models have been introduced in distributed systems with the goal of decreasing the data management complexity by decentralizing the resources' access and the allocation mechanisms [4,5,2,16]. We implemented an not pure economical[10] *QLB* approach based on the *Sealed Bid Auction*, where providers pay for acquiring requests. The criterion to select providers is in considering the providers' available capacity and the providers' bid. That is, the highest below criterion for a provider is given access to the request.

$$criterion : bid \times (1/(1 + U_t(p)))$$

In the economical approaches presented in [5,16], the *bid* is set using a bulletin board containing the preference for bidding which, conversely to our providers' *preferences* notion (see Section 4.1), inherently limits the provider's *preferences* to the queries' type.

Moreover, let us say that for our experiment simulations, we use virtual money (*jetons*) which is just seen as a means of regulation. In the course of the time the *jetons* are spent by providers in order to acquire requests. The process itself does not provide them anyway to earn money. Nevertheless, a source of financing is necessary to them, because otherwise, after some time, providers would not have more *jetons* to bid positively. Different solutions are possible. We have chosen to associate a bank with the mediator (if there were several mediators, there would be as many banks as mediators). The mediator's bank gives an specific amount of *jetons* to providers at the registration step and in the course of the time it equally redistributes the *jetons* which it gained to the providers after some given time.

6.2 Setup

In all experiments, the number of consumers and providers is 200 and 400 respectively, with only one mediator allocating all the incoming feasible queries[11]. Feasible queries arrive to the system in a Poisson distribution, which has been found in dynamic autonomous environments [10]. Consumers always ask for 1 provider to solve their requests (i.e. $q.n = 1$). Providers are initialized with a satisfaction value of 0, and a satisfaction size[12] of 500. Since our principal focus in this paper is to study the way in which requests are allocated in the system, we do not take into account the bandwidth problem and consider it as a future work.

[10] Since not only the bids are considered to select providers.

[11] We assigned sufficient resources to the mediator so that it does not cause bottlenecks in the system.

[12] Which denotes the k last arrival feasible queries.

Table 2. Simulation parameters

Parameter	Definition	Value
nbConsumers	Number of consumers	200
nbProviders	Number of providers	400
nbMediators	Number of mediators	1
qDistribution	Query arrival distribution	Poisson
iniSatisfaction	Initial providers' satisfaction	0
qTypes	Supported query types	10
nbSimulations	Number of realized simulations for each experience test	10

We set the providers' capacity heterogeneity, in our experiments, in accordance to the results in [14]. This work measures the nodes capacities in the Napster and Gnutella systems which are a clear example of large distributed systems. Based on these results, we generate around 10% of low-capable, 60% of medium-capable, and 30% of high-capable providers. The high-capable providers are 3 times more capable than medium-capable providers and still 7 times more capable than low-capable ones. We generate 10 types of requests which can be of two classes that consume, respectively, 130 and 150 treatment units at the high-capable providers[13].

In order to simulate high autonomy in our experiments, providers' *preferences* are randomly obtained between. More sophisticated mechanisms for obtaining such preferences can be applied ([13] for example).

6.3 Performance Results

We now investigate the impact on performance of the *Capacity Based*, *Economic*, and S_bQLB approaches against the providers' departures by *unsatisfaction*.

To this end, we have to set the *unsatisfaction* threshold in which providers decide to leave the system whether they fail it. In our experiments, providers decide to leave the system if they receive more than the 50% of uninteresting queries. We evaluate response times[14] with different request arrival rates in order to study the possible impact that the *unsatisfaction* departures could have. Also, we measured the number of *uninteresting* queries allocated by all these three approaches. Results are shown in Figure 2. We observe that S_bQLB and the *Economic* approaches allocate only interesting queries to providers, while the *Capacity Based* approach allocates in average a 50% of uninteresting queries!

Now, in order to see the impact of provider departures, we show in Figure 3(a) the ensured response times when providers are not allowed to leave the system by *unsatisfaction* (i.e. with the full system capacity). In Figure 3(b) the same parameters are drawn when providers are authorized to quite the system by unsatisfaction. As expected, the *Capacity Based* approach suffers significantly from *unsatisfaction* departures. We can observe that, in such a case, the *Capacity*

[13] Such a treatment takes almost 1.3 and 1.5 seconds, respectively.

[14] As is conventional, the response time is defined as the elapsed time from the time that a query q is issued to the time that $q.c$ receives the response.

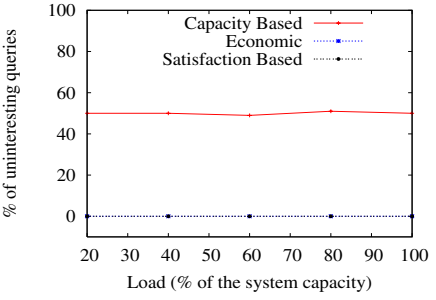
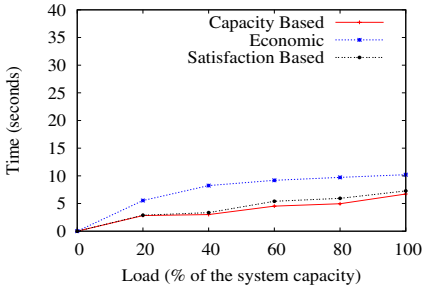
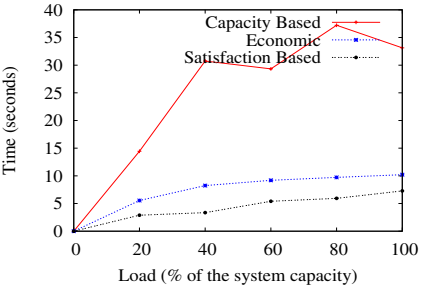


Fig. 2. Uninteresting received queries



(a) when providers do not leave the system by *unsatisfaction*



(b) when providers leave the system by *unsatisfaction*

Fig. 3. Response times

Based degrades in average the response times by a factor of 4.5! This is because even if it gives requests to all providers, it usually gives them uninteresting ones. On the other hand, since the S_bQLB and *Economic* approaches strive to give interesting requests to providers, they deals better with this problem in preserving their full system capacity.

The choice of an *unsatisfaction* threshold under which the providers would leave the system is very subjective and may depend on several external factors. As shown previously, it has a deep impact on response time, but also on adequation and satisfaction of remaining providers. By this fact, to avoid any suspicion on the choice of the *unsatisfaction* threshold, in the following experiments, we have preferred to consider captive providers: they are not allowed to leave the system whatever their degree of dissatisfaction is.

6.4 Providers' Utilization Results

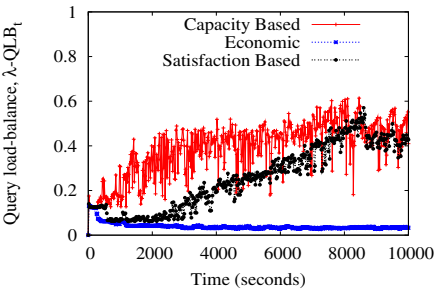
We measure the *query-balance* factor (λ-QLB_t) for different Poisson arrival rates[15]. λ-QLB_t (Y-axis) at different times (X-axis) of the experiment.

[15] For example, given our simulation setup, the 40% and 80% of the total system capacity correspond respectively to 84 and 164 requests per second.

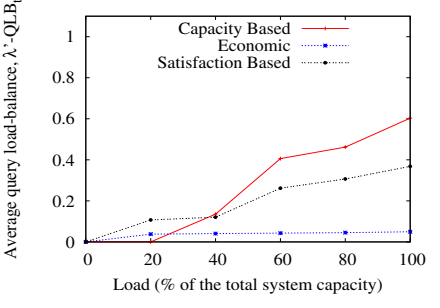
Contrary to the expected, the results show that the *Economic* approach has serious problems to ensure good λ-QLB_t in the system, because for providers, preference is more important than utilization in order to express their intentions.

The results show that the *Capacity Based* and S_bQLB approaches have serious problems to ensure good λ-QLB_t with request arrival rates under 40% of the total system capacity. In contrast, when the query arrival rate increases, both approaches improve the λ-QLB_t in the system. This is due to the fact that when most providers have no queries (i.e. have all their capacity available), queries may be allocated to those providers that spend more treatment units to perform them. Hence, each time a query is allocated to the less capable providers, the distance between the more and less utilized providers increases significantly.

To prove this intuition, we studied the ensured λ-QLB_t by varying the query arrival rate from 30 (at the beginning of the simulation) to 120% (at the end of the simulation) of the total system capacity. The result is shown in Figure 4(a). We observe that both *Capacity Based* and S_bQLB approaches improve the λ-QLB_t as the query arrival rate increases.



(a) for a range request load of 30-120%



(b) in average for different query rates

Fig. 4. Query-balance

In all cases, the *Capacity Based* is better than S_bQLB. This is because it takes into account providers' intentions. However, *Capacity Based* suffers from greater variations which means that, at times, some providers are much more utilized than others (when all of them have the same chances to get queries). S_bQLB better deals with such variations by considering providers' satisfaction. Furthermore, S_bQLB is much better than the *Economic* approach.

We also measured the *average query-balance* factor (λ-$QLB_{[t_1,\,t_2]}$) ensured by these approaches. That is, we analyzed how well these approaches avoid the *starvation* problems. To this end, we ran several tests with different request arrival rates and measure the λ-$QLB_{[t_1,\,t_2]}$ in such a time interval (from the beginning to the end of the simulations). The results of such experiments are shown by Figure 4(b).

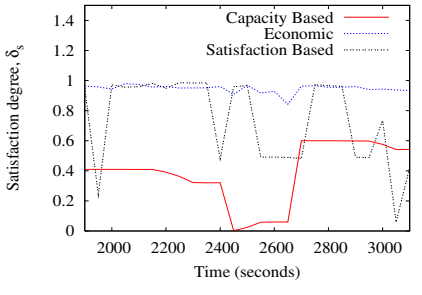
Unlike the λ-QLB_t experiments, we observe that for request arrival rates under 40% of the total system capacity the S_bQLB guarantees a better

λ'-$QLB_{[t_1, t_2]}$ than the *Capacity Based* and *Economic* approaches. This means that S_bQLB strives to give queries, in the course of the time, to all the providers in the system even if the arrival query rate is not sufficient to do it. In contrast, when the query arrival rate is greater than the 40% of the total system capacity, *Capacity Based* does it better. This is because the arrival query rate is sufficient for giving queries to all providers in the system. But, we observe that S_bQLB is still better than the *Economic* approach in which providers significantly suffers from *starvation* problems.

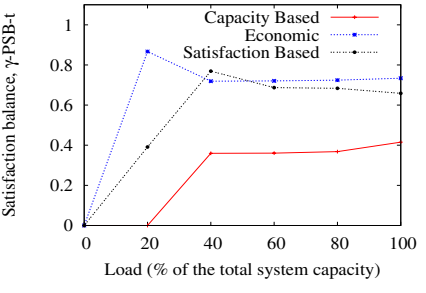
6.5 Providers' Satisfaction Results

As expected, the *adequation* of providers depends only on query arrival and their preferences for performing queries. Adequation is completely independent of the query arrival rates. In our experiments, the providers' *adequation* is 0.5 in average for all the query arrival rates. So, graphics are not presented here since they do not say anything else.

In our experiments the *allocation satisfaction* and *satisfaction* of providers are very similar but at different scale. This is because the results that providers get, depend roughly on the query allocation process. Then, because of space, we just present the providers' *satisfaction* results for an arrival query rate of 80% of the total system capacity (see Figure 5(a)).



(a) providers' satisfaction for a request load of 80%



(b) provider satisfaction balance

Fig. 5. Provider satisfactions

The results show that conversely to the QLB aspect (Section 6.4) the *Economic* approach better satisfies the providers than the S_bQLB and *Capacity Based* approaches. This is because, the providers' intentions in the *Economic* approach are principally based on the providers' preferences while S_bQLB takes equally into account the providers' utilization as shown in Section 4.2. We can also observe, in our experiments, that the *satisfaction* of providers with S_bQLB are almost all the time over the providers' *adequation* (i.e. over 0.5). In contrast, the *satisfaction* of providers with *Capacity Based* are almost all the time under the providers' *adequation*.

Furthermore, we observe that while the ***less satisfied*** provider with the *Economic* and S_bQLB approaches has, respectively, a *satisfaction* of 0.69 and 0.61 with an arrival query rate of 80% the total system capacity, the ***more satisfied*** provider with *Capacity Based* has a satisfaction of 0.61! This confirms that *Capacity Based* significantly hurts the providers' preferences for performing queries. The impact of this phenomenon was shown in Section 6.3.

Finally, the experiments show that the *Economic* and S_bQLB guarantee better γ-PSB$_t$ factors than the *Capacity Based* (see Figure Figure 5(b)). In other words, both *Economic* and S_bQLB strive to satisfy equally all the providers while *Capacity Based* does not.

7 Related Work

The problem of balancing queries while respecting the providers' autonomy for performing queries has not received much attention and is still an open problem.

Much work on query load balancing has been done in distributed systems [1,6,11,15,18]. We can classify load balancing algorithms into two approaches: *load based* and *capacity based*. *Load based* approach decide to allocate requests to those providers with the highest inverse probability of their reported load. Generally, load is defined as the number of request that the providers has in its arrival queries' queue. Thus, they inherently assume that providers and requests are homogeneous. *Capacity based* approach already take into account such heterogeneity by allocating requests to those providers with the greatest available capacity. The provider's capacity is defined as the maximum query rate that the provider can treat. All these works mainly model and address the problem of minimizing the providers' *load* or *utilization* for ensuring good response times in the system. However, unlike the S_bQLB approach, they do not consider the providers' *preferences* for performing requests.

Many solutions [2,5,16] strive to deal with providers' autonomy by means of economical models. Providers denote their preferences for performing queries via a bidding mechanism. A survey of economic models for various aspects of distributed system is presented in [4]. The motivation of economical models is to decentralize the access to the system's resources. Nevertheless, such models need robust market mechanisms for avoiding a degradation of the system's performance, and rationalized pricing schemes for giving *price* guarantees to consumers (see [4]).

Mariposa [16] is one of the first systems which deals with the query processing and data migration problem, in distributed information systems, based on a bidding process. In this approach, providers bid to acquire parts of a query and consumers pay for their queries' execution. In order to ensure a crude form of query load balancing, the providers' bid is multiplied by their load. Then, the mediator broker selects a set of bids that corresponds to a set of relevant queries and has an aggregate price and delay under a bid curve provided by the consumer. But, it is unclear how this approach really ensures the *QLB* in

the system. Furthermore, some queries may not get processed although relevant providers exist, just because they do not intend to treat them.

In addition, our approach also differs from the above works since it also strives to balance queries in the course of the time reducing, by consequent, the request starvation in the system. In contrast, we assume that providers say the truth about their *utilization* and *satisfaction*, but also, *Capacity Based* and *Economic* approaches do (for the providers' *utilization* and *credit balance* respectively). If this is not the case, works about providers' reputation can be applied to tackle this issue, for instance [3,7].

8 Conclusion

In this paper, we addressed the problem of balancing queries in dynamic and distributed systems with autonomous providers. We considered not only providers' *utilization* but also providers' *preferences* and *satisfaction*. This paper has several contributions.

First, we defined a model that enables providers to evaluate if they are meeting their objectives. The model relies on three definitions: *satisfaction* which defines what providers are really getting from the system; *adequation* which defines how much interesting are the proposed queries; and *allocation satisfaction* which allows providers to know whether the query allocation process is doing a good job for them.

Second, we proposed a *QLB* approach which takes care of the providers' *satisfaction*, called *Satisfaction-based QLB* (S_bQLB). S_bQLB allows providers to express their intention to perform queries by taking into account their preferences and utilization in accordance to their satisfaction.

Third, we evaluated and compared the S_bQLB approach against the *Capacity Based* and *Economic* approaches. The experimental results show that, in such dynamic and autonomous environments, the S_bQLB approach outperforms the *Capacity Based* and *Economic* approaches. We could observe that *Capacity Based* drastically hurts providers' autonomy in allocating them, in average, 50% of uninteresting queries. By consequent, the system's response time is degraded by a factor of 4.5! We also observed that, conversely to the expected, the *Economic* approach does not perform well the *QLB* task since it roughly depends on the providers' *preferences* for performing requests.

Finally, while S_bQLB does not rely on special mechanisms, *Economic* approach does [5,4]. This makes the S_bQLB approach more suitable for open and very large distributed systems, such as *peer-to-peer* systems.

As future work, we plan to design a S_bQLB approach that also takes into account the consumers' preferences for allocating requests. In order to guarantee equity at all levels, we must be able to decide which is the most important criterion, among consumers' satisfaction, providers' satisfaction or providers' utilization, at a given time in the system.

References

1. Y. Azar, A. Z. Broder, A. R. Karlin, and E. Upfal: Balanced Allocations. In SIAM Journal on Computing. 1999.
2. R. Buyya, H. Stockinger, J. Giddy, and D. Abramson: Economic Models for Management of Resources in Grid Computing. In CoRR Journal. 2001.
3. M. Feldman, K. Lai, I. Stoica, and J. Chuang: Robust Incentive Techniques for Peer-to-Peer Networks. In Procs. of the EC ACM Conference. 2004.
4. D. Ferguson and C. Nikolaou and J. Sairamesh and Y. Yemini: Economic Models for Allocating Resources in Computer Systems. Market-based control: a paradigm for distributed resource allocation. World Scientific Publishing Co., Inc. 1996.
5. D. Ferguson, Y. Yemini, and C. Nikolaou: Microeconomic Algorithms for Load Balancing in Distributed Computer Systems. In Procs. of the ICDCS Conference. 1988.
6. Z. Genova and K. Christensen: Challenges in URL Switching for Implementing Globally Distributed Web Sites. In Procs. of the ICPP Workshops. 2000.
7. S. Kamvar, M. Schlosser, and H. Garcia-Molina: The Eigentrust Algorithm for Reputation Management in P2P Networks. In Procs. of the WWW Conference. 2003.
8. P. Lamarre, S. Cazalens, S. Lemp, and P. Valduriez: A Flexible Mediation Process for Large Distributed Information Systems. In Procs. of the CoopIS Conference. 2004.
9. L. Li and I. Horrocks: A Software Framework for Matchmaking Based on Semantic Web Technology. In Procs. of the WWW Conference. 2003.
10. E. P. Markatos: Tracing a Large-Scale Peer to Peer System: An Hour in the Life of Gnutella. In CCGRID Symposium. 2002.
11. R. Mirchandaney, D. Towsley, and J. Stankovic: Adaptive Load Sharing in Heterogeneous Distributed Systems. In Parallel and Distributed Computing Journal. 1990
12. M. Nodine, W. Bohrer, and A. Ngu: Semantic Brokering over Dynamic Heterogeneous Data Sources in InfoSleuth. In Procs. of the ICDE Conference. 1999.
13. A. Sah, J. Blow, and B. Dennis: An introduction to the Rush language. In Procs. of the TCL Workshop. 1994.
14. S. Saroiu, P. Krishna Gummadi, and S. Gribble: A Measurement Study of Peer-to-Peer File Sharing Systems. In Procs. of the MCN Conference. 2002.
15. N. Shivaratri, P. Krueger, and M. Singhal: Load Distributing for Locally Distributed Systems. In Computer IEEE Journal. 1992
16. M. Stonebraker, P. Aoki, W. Litwin, A. Pfeffer, A. Sah, J. Sidell, C. Staelin, and A. Yu: Mariposa: A Wide-Area Distributed Database System. In VLDB Journal. 1996.
17. K. Sycara, M. Klusch, S. Widoff, and J. Lu: Dynamic Service Matchmaking Among Agents in Open Information Environments. In SIGMOD Record. 1999.
18. H. Zhu, T. Yang, Q. Zheng, D. Watson, O. Ibarra, and T. Smith: Adaptive Load Sharing for Clustered Digital Library Servers. In HPDC Symposium. 1998.
19. T. Özsu and P. Valduriez: Principles of Distributed Database Systems (2nd ed.). Prentice-Hall, Inc. 1999.

Efficient Dynamic Operator Placement in a Locally Distributed Continuous Query System

Yongluan Zhou, Beng Chin Ooi, Kian-Lee Tan, and Ji Wu

National University of Singapore

Abstract. In a distributed processing environment, the static placement of query operators may result in unsatisfactory system performance due to unpredictable factors such as changes of servers' load, data arrival rates, etc. The problem is exacerbated for continuous (and long running) monitoring queries over data streams as any suboptimal placement will affect the system for a very long time. In this paper, we formalize and analyze the operator placement problem in the context of a locally distributed continuous query system. We also propose a solution, that is asynchronous and local, to dynamically manage the load across the system nodes. Essentially, during runtime, we migrate query operators/fragments from overloaded nodes to lightly loaded ones to achieve better performance. Heuristics are also proposed to maintain good data flow locality. Results of a performance study shows the effectiveness of our technique.

1 Introduction

In many emerging monitoring applications (e.g. network management, sensor networks, financial monitoring etc.), data occurs naturally in the form of active continuous data streams. These applications typically require the processing of large volumes of data in a responsive manner. In order to scale up the volumes of streams and queries that can be processed, a distributed stream processing system is inevitable. However, as the properties of data streams (e.g., arrival rates) and the processing servers' load are hard to predict, the initial placement of query operators may result in unsatisfactory system performance. The problem is exacerbated by multiple continuous queries that run long enough to experience the changes in the environment parameters. As such, any suboptimal performance will persist for a long time.

Clearly, a distributed stream processing system must adapt to changes in environment parameters and servers' load. We believe a dynamic load management scheme is indispensable for the system to be scalable. In particular, we expect aggressive methods such as query operator migration during runtime to bring long term benefit (especially for long running continuous queries) even though they may incur some short term overhead. The necessity of dynamic load management for a scalable distributed stream processing system has also been identified in previous work [7,11]. However, to date few complete and practical solutions have been proposed for this problem. In this paper, we offer our solution to the problem. More specifically we make the following contributions:

R. Meersman, Z. Tari et al. (Eds.): OTM 2006, LNCS 4275, pp. 54–71, 2006.

– We formally define the metric *Performance Ratio (PR)* to measure the relative performance of each query and the objective for the whole system (informally, we want to minimize the worst relative performance among all queries).

– By building a new cost model, we identify the heuristics that can be used to approach the objective. More specifically, the heuristics (1) balance the load among all the processing nodes; (2) restrict the number of nodes that the operators of a query can be distributed to; (3) and minimize the total communication cost under conditions (1) and (2).

– The design objective of a platform independent (independent on the underlying stream processing engines) and non-intrusive load management scheme distinguishes our approach from existing ones (e.g. [14]). The proposed techniques are meant to allow the leveraging of existing well developed single-site stream processing engines without much modifications. This is reflected throughout the design of the whole system, especially the load selection strategy.

– To support heuristic (1), we focus on new architectural design that allows us to tap on existing well studied load balancing algorithms instead of proposing new ones. The architectural design includes constructing the load migration unit, load management partner selection, online collection of load statistics, selection of operators to be migrated, operator migration mechanisms.

– To reduce the overhead of employing heuristic (2), unlike existing proposals [7,11,14] where load (re)distribution is done at the operator level, we adopt the notion of *query fragments* (a subset of operators) as the finest migration unit. It also helps reduce the overhead of making load balancing decisions.

– To employ heuristic (3), we propose the data flow aware load selection strategy to select the query fragments to be migrated. It effectively maintains data flow locality so that the communication cost is minimized.

– We conducted an extensive simulation study to evaluate the proposed strategy. Results show that the proposed strategy can effectively adapt to the runtime changes of the system to approach our objective.

The rest of this paper is organized as follows. Section 2 formulates the problem and presents our analysis. We present the details of our system design in Section 3. Experiment results are presented in Section 4. Finally Section 5 concludes the paper.

2 Problem Formulation and Analysis

In this section, we present the system model and define the metric to measure the system performance, followed by a formal presentation of the problem statement. Finally, we analyze the problem by building a new cost model and present the proposed heuristics.

2.1 Problem Formulation

Our system consists of a set of geographically distributed data stream sources $S = \{s_1, s_2, \cdots, s_{|S|}\}$ and a set of distributed processing nodes $N = \{n_1, n_2, \cdots, n_{|N|}\}$ interconnected by a local network. As transfer cost from the sources

to the processing nodes is much higher than the one among the processing nodes, each source stream is routed to multiple processing nodes through a delegation node. We denote the delegation scheme as Ω. Users impose a set of continuous queries $Q = \{q_1, q_2, \cdots, q_{|Q|}\}$ over the system. The set of operations $O_k = \{o_1, o_2, \cdots, o_{|O_k|}\}$ of query q_k might be distributed to a set of nodes $N_k \subseteq N$ for processing. The operators we consider include filters, window joins and window aggregations. In addition, we denote the set of streams that a query q_k operates on as S_k.

Like previous work on continuous processing of streams [5,11], we are concerned about the delay of resulting data items, which is also one of the main concerns of end users in terms of system performance. More formally, if the evaluation of query q_k on a source tuple $tuple_l$ from stream s_l generates one or more result tuples, then the delay of $tuple_l$ for q_k is defined as $d_k^l = t_{out} - t_{in}$, where t_{in} is the time that $tuple_l$ arrived at the system and t_{out} is the time that the result tuple is generated. If there are more than one result tuples, then t_{out} is the time that the last one is generated. A similar metric was used in [11]. We focus on this metric because users in a continuous query system typically make decisions based on the results arrived so far. Shorter delay of result tuples would enable a user to make more timely decisions.

At a closer look, d_k^l includes the time used in evaluating the query (denoted as p_k^l), the time waiting for processing as well as the time it is transferred over the network connections. For a specific processing model and a particular query q_k, we regard the evaluation time p_k^l as the inherent complexity of q_k. Since different queries may have different inherent complexities, the value of d_k^l cannot reflect correctly the relative performance of different queries. For example, a query may experience a long delay because its evaluation time is long. We cannot conclude that the relative performance of this query is worse than another one which has a shorter evaluation time. However, in a multi-query and multi-user environment, we wish to tell the relative performance of different queries. Hence we propose a new metric *Performance Ratio (PR)* to incorporate the inherent complexity of a query. Formally, the PR_k^l of the processing of $tuple_l$ for q_k is defined as $PR_k^l = \frac{d_k^l}{p_k^l}$. And the performance ratio of q_k is defined as $PR_k = \max_{s_l \in S_k} PR_k^l$. PR_k reflects the relative performance of q_k. Our objective is to minimize the worst relative performance among all the queries.

The formal problem statement is as follows: *Given a set of queries Q, a set of processing nodes N, a set of data stream sources S and a delegation scheme Ω, according to the change of system state, dynamically distribute the operators of each query to the $|N|$ processing nodes so that the maximum performance ratio $PR_{max} = \max_{1 \le k \le |Q|} PR_k$ is minimized.*

2.2 Problem Analysis

In this section we develop a cost model to estimate the values of d_k^l and p_k^l. Note that our cost model is meant to be simple for us to figure out the main

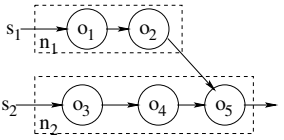


Fig. 1. An example query plan

factors that affect these values and to allow us to analyze the problem complexity. Finding that the problem is NP-hard, we design some heuristics to help solve the problem.

Cost Model. In our cost model we adopt the following simplifications and assumptions:

1. Operators of each query compose a separate processing tree. They are grouped into query fragments and distributed to the processing nodes. Figure 1 shows an example processing tree for a query whose operators are grouped into two query fragments and distributed to two nodes: n_1 and n_2. Tuples arrived at each node are processed in a FIFO manner. Only when an input tuple[1] is fully processed would a new input tuple be processed. The cost of delivering the final results to the users is not considered.
2. For an operator o_j, we assume its per-tuple evaluation time t'_j is independent of its location. And we define its average per-tuple selectivity sel_j as the average number of tuples that would be generated for a given input tuple.
3. Workload ρ_i of a node n_i is defined as the fraction of time that the node is busy.

Given these assumptions, we now look at how to estimate p^l_k and d^l_k. In a particular execution plan of a query, for source tuples from each querying source, there is a path composed by some operators and possibly some network connections. For example, in Figure 1, the path for source tuples from s_1 consists of o_1, o_2, o_5 and the connection between n_1 and n_2, while the path for those from s_2 comprises o_3, o_4 and o_5. Hence, roughly speaking, the p^l_k and d^l_k of a source tuple are respectively equal to the total processing time of the operators in its path and the total time that the tuple stays in its path. In the following paragraphs we will compute them one by one.

For query q_k, assume the path for source tuples from s_l comprises a set O^l_k of operators and some network connections. Furthermore, let O^l_k be distributed to a set N^l_k of nodes and $O^l_{k,i} \subseteq O^l_k$ be the subset of operators of O^l_k assigned to node n_i (where $n_i \in N^l_k$). Let the average per-tuple evaluation time of operator $o_{l_j} \in O^l_k$ be t'_j and its average per-tuple selectivity be sel_j. Without loss of generality, assume o_{l_j} is processed before $o_{l_{j+1}}$. Note that only those source tuples that would be output as result tuple(s) are counted in our metric (hence, each operator's selectivity on these particular tuples is at least 1). Assume $tuple_l$

[1] A tuple here could be a batch of individual tuples in a batch processing mode.

from s_l is such a tuple, then the average processing time of o_{l_j} incurred by $tuple_l$ is $t_j = t'_j \prod_{h=1}^{j-1} \max(sel_h, 1)$. Hence we have

$$p_k^l = \sum_{o_{l_j} \in O_k^l} t_j. \tag{1}$$

In our model every processing node is a queueing system. From queueing theories, in all solvable single task queueing systems, the time that a data item spends in a system can be calculated as $t = g(\rho) * t_s$, where t_s is the processing time of a data item and $g(\rho) \geq 1$ is a monotonically increasing concave function of the system's workload ρ. The exact form of $g(\rho)$ depends on the type of system, e.g. $g(\rho) = \frac{1}{1-\rho}$ in an M/M/1 system.

This inspires us to model the delay of $tuple_l$ as

$$d_k^l = (\sum_{n_i \in N_k^l} (f(\rho_i) \times \sum_{o_{l_j} \in O_{k,i}^l} t_j)) + t_c \times m, \tag{2}$$

where t_c is the communication delay of a tuple and m is the number of times that a tuple is transferred over the network. $f(\rho_i)$ is a monotonically increasing concave function. Note that $f(\rho_i)$ is different from $g(\rho)$ mentioned above and may have a much higher value than $g(\rho)$. That is because there are multiple tasks running on each node. We assume $f(\rho_i)$ is identical for all nodes. Hence the first term of the right-hand side of Equation (2) summarizes the delay in the processing nodes while the second term summarizes the delay caused by the communications.

Based on Equations (2.1), (1) and (2), we have

$$PR_k^l = PPR_k^l + CPR_k^l, \tag{3}$$

where

$$PPR_k^l = \frac{\sum_{n_i \in N_k^l} (f(\rho_i) \times \sum_{o_{l_j} \in O_{k,i}^l} t_j)}{\sum_{o_{l_j} \in O_k^l} t_j}, \tag{4}$$

and

$$CPR_k^l = \frac{t_c \times m}{\sum_{o_{l_j} \in O_k^l} t_j}. \tag{5}$$

We call PPR_k^l the processing performance ratio (PPR) and CPR_k^l the communication performance ratio (CPR). Analogously, $PPR_k = \max_{s_l \in S_k} PPR_k^l$ and $CPR_k = \max_{s_l \in S_k} CPR_k^l$.

Problem Complexity. Given the cost model, let us examine the complexity of the problem. We can observe that the total number of possible allocation schemes is $|N|^{|O|}$ where $O = \bigcup_{1 \leq k \leq |Q|} O_k$. Even worse, we can derive that the problem is actually NP-hard. To see this let us first ignore the communication

cost and only consider minimizing $PPR_{max} = \max_{1 \leq k \leq |Q|} PPR_k$. It is easy to see from Equation (4) that PPR_k^l is a weighted sum of the $f(\rho_i)$ values, where the weight for $f(\rho_i)$ is the fraction of evaluation time p_k^l allocated to node n_i. Assume we can migrate the load between nodes in the finest granularity. Then we have the following observation.

Observation 1. *To minimize PPR_{max}, PPR_k is equal for all queries and ρ_i is equal for all nodes.* □

The intuition behind it is when PPR_k of a query q_k is higher than the others, we can always allocate more resources to q_k (i.e. reducing the workload of some of the processing nodes for q_k by load migration to the other nodes) so that PPR_k is still the largest but is reduced. When the load is balanced then PPR_k equal to $f(\overline{\rho})$ for all queries, where $\overline{\rho}$ is the uniform workload of all nodes. However, we cannot migrate the load in the finest granularity in practice and hence the best plan is to minimize the difference of loads among all the nodes. By restricting our problem to ignore the communication cost, it is equivalent to a MULTIPROCESSOR SCHEDULING problem which is NP-hard. Hence our problem is NP-hard.

Heuristics. In view of the complexity of the problem, we opt to designing heuristics instead of finding an optimal algorithm. From the estimation equation d_k^l, we know that the extra delay is caused by the communication and the workload of the system. Hence, we adopt the following heuristics. (1) Dynamically balance the workload of the processing nodes. This heuristic is inspired by Observation 1. (2) Distribute operators of a query to a restricted number of nodes so that communication overhead of a query is limited. We call the maximum of this number as the distribution limit of that query. Note that always distributing all the operators of every query to a single node is impractical, because it would incur excessive data flow over the network.(3) Minimize the communication cost under conditions (1) and (2). In short, we have to design a dynamic load balancing scheme where the operations of each query should not be distributed to too many nodes and the total communication traffic is minimized.

Besides employing the heuristics stated above, the scheme should also satisfy the following objectives in the perspective of system design:

1. *It is fast and scalable.* Because dynamic re-balancing could happen frequently at runtime, the overhead of making re-balancing decisions should be kept low. Furthermore, a distributed scheme is preferred to enhance scalability and avoid bottleneck.

2. *It does not rely on any specific processing model.* There are different single-node processing models that are currently under development such as TelegraphCQ [6], Aurora [4] and STREAM [1]. Our system is not restricted to any processing model because it separates the stream processing engine in each node from the distributed processing details. Queries are compiled into logical query plans which consist of *logical operators*. The logical operators are distributed to the processing nodes by our placement scheme. Then the logical operators

would be mapped into *physical operators* by the stream processing engine for processing. Different engines under different processing models could map a logical operator into a different physical operator.

2.3 Related Work

Distributed continuous query systems attracted much research attention in the recent years. The necessity of dynamic load management in distributed stream processing has been identified in several published references e.g. [7,11]. However, the authors did not propose any complete and practical strategies. In Flux [9], a dynamic load balancing strategy for the *horizontal* (or intra-operator) parallelism was employed. While the mechanism was developed in the context of continuous queries, a centralized synchronous controller is used to collect workload information and to make load balancing decisions. Our work, on the other hand, takes a complimentary approach by focusing on allocation of operators in the context of *vertical* (or pipelined) parallelism. Furthermore, our approach is decentralized and asynchronous. More recently, Borealis system [14] also adopted a centralized load distribution technique in the context of vertical parallel processing. An innovative load balancing approach is presented which considers the time correlations between the operators. On the contrary, our work does not focus on proposing new load balancing algorithms.

Furthermore, the problems of the above two pieces of work bear a few important differences from ours. First, in their approach, stream delegation is not employed. Hence it is possible that some sources have to communicate with multiple processing nodes or some processing nodes may have to collect streams from a lot of sources. Second, the network resources are assumed to be abundant and hence communication cost is ignored in their techniques. How to take the communication cost in account is still unclear. Third, without considering the delegation scheme and the communication cost, the problem is only a partitioning problem which partitions the operators into balanced partitions. The processing nodes are identical in terms of partition allocation. However, our problem is essentially an assignment problem as assigning query operators to different nodes would have different communication cost for a given delegation scheme.

On the other hand, [3,8] focused on minimizing communication cost but ignores load balancing. [10] studied the static operator placement in a hierarchical stream acquisition architecture, which is much different from our system architecture. Load balancing is also ignored in this piece of work. [15] proposed an adaptive scheme disseminate stream data to the distributed stream processors without considering operator placement.

3 System Design

In our dynamic operator placement scheme, we adopt a local load balancing strategy. Each node would select its load management partners and dynamically

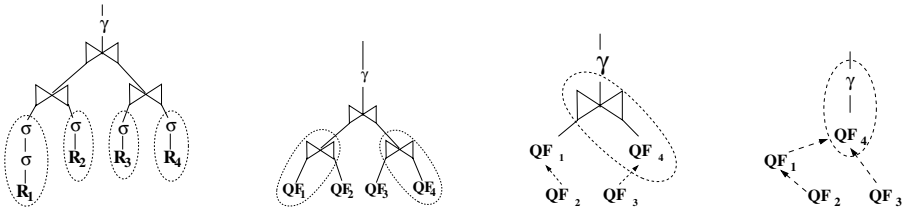
balances the load between its partners. To implement this, there are several issues to be addressed: (1) initial placement of operators; (2) load management partner selection; (3) workload information collection; (4) load balance decision-making; (5) selection of operators for migration. We address these issues in the following subsections.

3.1 Initial Placement of Operators

In our initial placement scheme, we only consider minimizing the communication cost and leave the load balancing task to our dynamic scheme. The scheme generates one query fragment for each participating stream and then distributes the query fragments to the delegation nodes of their corresponding streams. More specifically, the scheme comprises the following steps:

1. When a query is submitted to the system, it is compiled and optimized into a logical query plan without considering the distribution of the data streams. The logical query plan, which is represented as a traditional query plan tree, determines the required logical operators such as filters, joins, aggregation operators and their processing orders. Existing optimization techniques [12,2] can be applied at this step. Figure 2(a) is an example of the resulting query tree of this step.
2. For each stream involved in the query, generate one query fragment which is initially set to empty. Add each leaf node (i.e. the stream access operators) to its corresponding query fragment QF_i and then replace it with QF_i.
3. For each query fragment, if the parent operator is a unary operator, the operator would be added to the query fragment and removed from the query tree. The step is repeated until all the operators are removed or the parent operator for every query fragment is a binary operator. Figure 2(b) is an example of the resulting query tree of this step. The intuition is to place each stream's filters at its delegation node to reduce the amount of data to be transferred.
4. Now we have a query tree in which all the next-to-leaf nodes are binary operators. Add each next-to-leaf binary operator to one of its two child query fragments, say QF_i, whose estimated resulting stream rate is higher than the other one. Then remove the other query fragment from the tree and push QF_i up a level to replace that binary operator. A binary operator is added to the query fragment of higher (estimated) resulting stream rate to reduce the volume of data that needs to be transmitted through the network if the two fragments of the two involved streams are to be evaluated at two different nodes. This process continues until all operators are removed or the parents of one or more of the remaining query fragments are unary operators. For the latter case, the algorithm goes back to step (3). Figures 2(c) and (d) illustrate the procedure of this step.
5. Distribute the query fragments to the delegation nodes of their corresponding streams.

Based on the operator ordering, there is a downstream and upstream relationship between some of the query fragments. For example, in Figure 2, results



(a) Phase 1 (b) Phase 2 (c) Phase 3 (d) Phase 4

Fig. 2. Query Fragments Generation

Algorithm 1. PARTNERSELECT

```
sort neighbors in descending order of neighboring factor;
for (i ← 0; |g1| < max1 AND i <|neighbors|+ MaximumTry; i++) do
    if i < |neighbors| then n ← neighbors[i];
    else n ← a random node ∉ neighbors ∪ g1;
    if n ∈ g2 then
        move it from g2 to g1;
    else if n ∉ g1 then
        send a request to n;
        if the request is accepted then
            add n to g1;
        endif
    endif
endfor
```

of QF_2 should be further processed by the binary operator of QF_1 and hence we call QF_2 the upstream query fragment of QF_1. Similarly, QF_1 is the upstream query fragment of QF_4. Symmetrically, we call QF_1 (or QF_4) the downstream query fragment of QF_2 (or QF_1). We call a query fragment's downstream or upstream query fragments its neighbors. For instance, QF_2 and QF_4 are neighbors of QF_1. Furthermore, if a query fragment QF_i's corresponding data stream is delegated to a node n_j then QF_i is called a *native query fragment* of n_j and n_j is a *native node* of QF_i. Otherwise, QF_i is called a *foreign query fragment* of n_j and n_j is a *foreign node* of QF_i.

Furthermore, the native nodes of two neighboring query fragments are called *neighbors* to each other. And the number of neighboring query fragments between two nodes is called the *neighboring factor*.

3.2 Partner Selection Strategy

As stated above, our dynamic load balancing scheme is a local strategy. Each node n_i has a number of load management partners (abbreviated as partners). The partner relationship is symmetric, i.e. if n_i is a partner of n_j, then n_j is also a partner of n_i. In this section, we discuss the partner selection strategy for each node.

In our scheme, each node sends out requests to some other nodes to initiate the partner relationships and receives such requests from its peers. We separate the partners of each node into two groups : (1) g_1, the relationship is created by the (explicit) request of this node; (2) g_2, the rest. There is a maximum bound for each group of partners denoted as max_1 and max_2 respectively. Each node would use Algorithm 1 to send out requests. Neighbors with higher neighboring factors with the current node have higher priority to be selected. That is to enhance the opportunity of reducing communication cost during load redistribution, which is can easily be seen in Section 3.5. Algorithm 1 is implemented in asynchronous mode in our system. It does not wait for a remote response but instead returns once all requests have been sent out. After a node receives a response message, the algorithm is called to resume the processing. Furthermore, a node n_i which receives a request will check whether the sender n_j is also being requested by n_i or is already in g_1. If so, n_i accepts the request and adds n_j into g_1 if necessary. Otherwise it adds n_j into g_2 if $|g_2| < max_2$ or sends back a reject message otherwise. A node will update its partners periodically.

3.3 Information Collection Strategy

The information collection strategy determines when and how workload information of nodes in the system is collected and also what information is to be collected.

We adopt a window based and asynchronous workload collection approach. Time is divided into windows which have static lengths τ. Each node accumulates the total processing time t of all its physical operators within each window and the workload with respect to a window is computed by dividing t by τ. Each node asynchronously collects its workload within each window and updates its workload once the current time window elapsed. It broadcasts the workload information to all its partners if its workload increases to κ or decreases to $1/\kappa$ times of the last broadcast value.

The above strategy performs well only if the input rate and the processing time are constants. But in practice they are random variables. The resulting workload may fluctuate over time, which renders the system unstable. As stated before, we only focus on adaptation to long term system changes which would bring long term benefits and alleviate the short term adaptation overhead. To prevent the system from reacting to short term fluctuations, we use a low pass filter to remove the high frequency noises (caused by the short term changes of stream rates, tuple processing time, etc.) in workload collection. In particular, workload is computed as $\rho_{i+1} = \alpha \times \rho_i + (1-\alpha) \times \rho_c$, where ρ_{i+1} and ρ_i are the workload information used for load balancing after $i+1$ and i time windows, and ρ_c is the collected workload within the $(i+1)$th time window. α is a parameter to determine the responsiveness of the estimated value to the workload changes. The purpose of using this formula in previous work is to give more weight to recent collected statistics. Here we analytically show that it can also smooth out short term fluctuations. A validation experiment can be found in [16].

We now consider how α would be set in a system. Without loss of generality, we assume the workload is increasing. Given the initial workload ρ_0 and that we want to filter out transient workload fluctuation where the workload is changed to $l\rho_0$ ($l > 1$) within $m_1\tau$ time and last for $m_2\tau$ time, we should choose α such that the estimated workload after $(m_1 + m_2)\tau$ time $\rho_{m_1+m_2}$ should satisfy $\rho_{m_1+m_2} \leq \kappa\rho_0$. In practice, the values of m_1, m_2 and l reflect the typical range and time span of short term fluctuations. They can be adaptively tuned by collecting the characteristics of the system. In our calculation, we assume the workload increases $(l-1)\rho_0/m_1$ within each τ time during the $m_1\tau$ period. After $(m_1 + m_2)\tau$ time, the estimated workload $\rho_{m_1+m_2}$ can be calculated as [16]:

$$\alpha^{m_2}\rho_0 + \frac{\alpha^{m_2}(l-1)\rho_0}{m_1}(m_1 + 1 + \frac{\alpha^{m_1+1} - 1}{1-\alpha}) + (1 - \alpha^{m_2})l\rho_0.$$

Substitute the above equation into the inequality $\rho_{m_1+m_2} \leq \kappa\rho_0$, we have

$$\alpha^{m_2} + \frac{\alpha^{m_2}(l-1)}{m_1}(m_1 + 1 + \frac{\alpha^{m_1+1} - 1}{1-\alpha}) + (1 - \alpha^{m_2})l \leq \kappa.$$

Hence we can calculate the lower bound of α by solving the above inequation given the values of m_1, m_2 and l. For example, given $m_1 = m_2 = 1$, $l = 2$ and $\kappa = 1.2$, we can get $\alpha \geq 0.9$. The case for short term workload decrease can be analyzed similarly. On the other hand, α also cannot be too close to 1, otherwise the current workload will not be reflected. A similar upper bound analysis can be performed [16].

3.4 Load Balance Decision Strategy

The load balance decision strategy determines whether it is beneficial to initiate a load balance attempt and how much workload should be transmitted between the nodes. Our strategy is adapted from the local diffusive load balancing strategy introduced in [13]. It is a receiver-initiated strategy, which is found to be more efficient in [13]. It works in rounds. The length of each round is denoted as Δ. Each node maintains its own value of Δ. At the start of each round, Algorithm 2 is run to generate one workload request if necessary. In this algorithm, the load request is generated by the potential load receiver (i.e., the node with smaller load initiates load balancing). Since we focus on continuous queries, load migration can bring long term benefits. As such our decision strategy does not consider the short term migration overhead. Once a node receives a workload request, it satisfies the request as much as possible, provided the workload to send out within each Δ time window is no more than half of its total workload at the beginning of the current window.

It is possible that the nodes in the system are separated into several non-overlapping groups and the workloads are not balanced between groups. Hence once a node in our system detects that itself and all of its partners are overloaded, it will randomly probe the other nodes until it finds an underloaded node to add it as a partner or the probe limit is reached.

Algorithm 2. GENERATEREQUEST

compute the average workload $\overline{\rho}$ within itself and its partners;
if *the local workload* $\kappa\rho_l < \overline{\rho}$ **then**
  find the partner n_i whose workload ρ_i is the largest;
  compute the load request $\rho_r = (\rho_i - \rho_l)/2$;
  request ρ_r amount of workload from n_i;
endif

3.5 Load Selection Strategy

As stated above, once a potential load sender receives a load request, it will select the *victim* query operators to satisfy the request as far as possible. When multiple such requests are received, the sender processes them in descending order of the workload amount requested. The sender will estimate its resulting workload after each migration, and if it detects that half of the workload has been exported within the current Δ interval, it will stop processing any request until the start of the next round. In this subsection, we explore how to select the victim operators for migration and discuss how to migrate them in the next subsection.

Migration Unit. The first question to be answered is what is the smallest task unit used for load migration. We consider the following choices:

1. Using *the whole query* as the migration unit is easy to implement. However, a good evaluation plan often distributes the operations across multiple nodes in order to minimize the communication overheads. So migrating in the unit of query is inappropriate.

2. *Operator* as another candidate is a fine-grained unit. Migrating at this level may result in better balance state. However, it is hard to implement our second heuristic which imposes a distribution limit on the query operators (see Section 2.2). When we are trying to move an operator, we have to know the location of the other operators belonging to the same query. Otherwise, we do not know if the distribution limit is violated. This results in high update overhead and is not compatible to our local strategy as a node cannot make decisions based on local information.

3. *Query fragments.* Based on the above analysis, a good candidate for migration unit should render the maintenance of good query plans easy and allow the separation of load balancing strategy from the underlying stream processing engine and hence introduce less complexity to the existing processing techniques. Furthermore, this unit should not be too coarse to restrict the adaptive ability of the load management module. For the above purposes, we would like to find a subset of operators that is of appropriate size and would be processed in the same site in most cases for a good query plan. Furthermore, we consider only candidates in the logical level. We adopt the notion of query fragment - a subset of logical operators of a query. We set the number of query fragments of a query as its distribution limit. This exempts the task of keeping track of the distribution of all the operators of a query while we are implementing heuristic

(2). The distribution limit would always be met no matter where we allocate the query fragments.

While a query can be fragmented in a lot of ways, we simply use the query fragments generated in our initial placement scheme as the migration units. Operators in each of such query fragments would be allocated to the same processing node in a good query plan generated by applying traditional optimization heuristics. Furthermore, by doing so, the distribution limit of a query is set to the number of streams involved by the query. Here, we assume that queries involving more streams are more complicated and hence can afford a higher distribution limit.

Data Flow Aware Load Selection. The choice of query fragments to be migrated is critical in maintaining data flow locality. A poor choice may cause streams to be scattered across too many nodes and result in network congestion. In this subsection, we propose a lightweight query fragment selection strategy which makes decisions only based on local information.

In our strategy, for each request, the sender chooses the query fragments in the following order until the request is satisfied or half of the workload of this node has been exported within the current Δ interval.

1. *Query fragments that are foreign to the sender but native to the receiver.* This kind of query fragments is considered to be of highest priority to migrate because migrating them has the potential to reduce the data flow.
2. *Other query fragments that are foreign to the sender.*
3. *Query fragments that are native to the sender.* This kind of query fragments is considered of lowest priority for migration because migrating them tends to scatter the streams delegated to this node.

The above heuristics are reasonable in maintaining data flow locality. However, its categorization is too coarse. The migrations of the query fragments within each category may still have different effects on the data flow locality and the delay of the queries. For example, migrating a query fragment QF_i to a node that is evaluating a neighbor of QF_i may bring less increase of data flow than migrating it to other nodes. This is because it avoids the transfer of the data flow between QF_i and its neighbor. Hence, within each of the above categories, we further classify the query fragments into one of the following categories and we list them in the order of descending migration priorities.

1. *Query fragments that have neighbors being evaluated at the receiver but none at the sender.* The migration of this class of query fragments eliminates the transmission of the data flow between the sender and the receiver caused by the migrated query fragment. Figure 3(a) shows a possible situation in this case. The situations before and after migration are plotted on the left and the right respectively. Solid arrows in the figure indicate the data flows between the query fragments. For brevity, the other query fragments being evaluated in the two nodes are not shown. In this example QF_1 is a neighbor of QF_2. n_i is the sender while n_j is the receiver. After migration, the data flow introduced by QF_1 and QF_2 between n_i and n_j is eliminated.

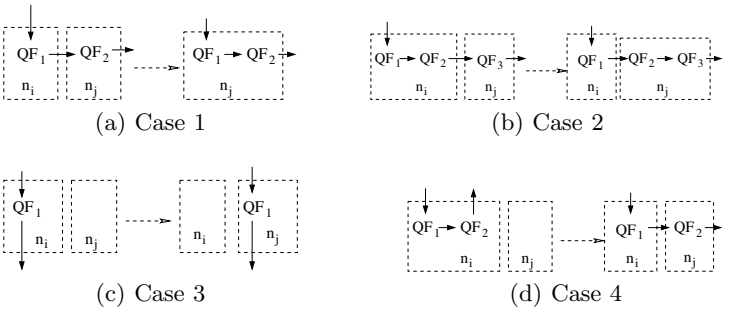


Fig. 3. Query fragments migration cases

2. *Query fragments that have neighbors at both nodes.* This class of query fragments has lower migration priority than the above-mentioned one because the migration eliminates one data flow but also creates another one between the sender and the receiver. For example, in Figure 3(b), the transmission of the data flow introduced by QF_2 and QF_3 is eliminated while the one incurred by QF_1 and QF_2 is created by the migration.

3. *Query fragments have neighbors at neither node.* Figure 3(c) is an example situation.

4. *Query fragments that have neighbors at the sender but none at the receiver.* This class has lower priority than the third one because the migration may introduce extra data flow between the sender and the receiver. An example of this case can be found in Figure 3(d). The migration in this example creates the data flow between n_i and n_j caused by QF_1 and QF_2.

If there is more than one query fragment in the above subcategories, we will compute the migration priority for each of them and will migrate those with higher priorities first. The migration priority of a query fragment is computed as $\frac{\rho}{\max(size,1)}$, where ρ is the workload it incurs, and $size$ is its state size in bytes. We call this value the *load density* of the query fragment as it means the amount of workload will be migrated for each byte of state transmission. Furthermore, ρ is estimated by summing up the estimated workload incurred by each of its logical operator, which is estimated as $1/n$ of the workload caused by its corresponding physical operator. n is the number of logical operators sharing that physical operator.

4 A Performance Study

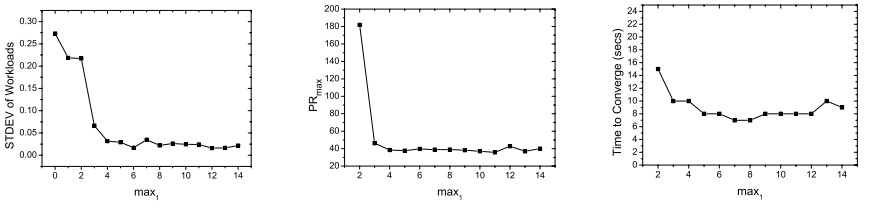
In our experiments, the stream processing engine in each node is an emulation of the TelegraphCQ system. We use a simulator to simulate the communication among the processing nodes. The simulator is implemented in JAVA using the JavaSim discrete event simulation package. We use 32 simulation nodes and an additional sink node as our basic configuration. Each processing node is

delegated 3 streams. Tuples from every stream are of 100 bytes and consist of 10 attributes. The bandwidth of the network connecting the nodes is modeled as 100Mbps.

We use 500 randomly generated queries and a total of 5750 logical operators, to measure our system performance. The sliding window size for window joins is randomly selected from 5000 to 20000. The selectivities of the operators are from 0.5 to 0.8. We set the average data inter-arrival time to be $4ms$ and the mean processing time for each filter and join operation to be $20\mu s$ and $80\mu s$ respectively. Besides, we use the following algorithm parameters: the workload collection window $\tau = 100ms$, the length of load management round $\Delta = 1s$, and the threshold to broadcast workload $\kappa = 1.2$. The real values of p_k^l and d_k^l were collected online and the PR_k values were computed by the sink node when it received a result tuple.

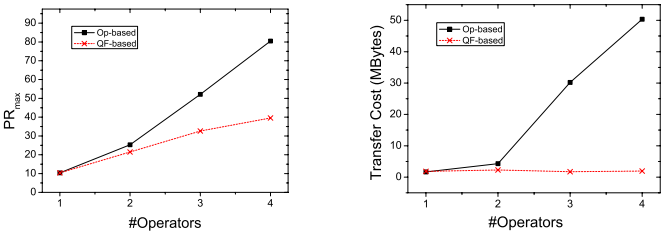
4.1 Partner Selections

We have two parameters for our partner selection strategy: max_1 and max_2. In this experiment, we set $max_2 = \lceil \frac{1}{2} max_1 \rceil$ and vary the value of max_1. To generate an imbalanced workload, the streams that a query operates on are chosen according to a Zipfian distribution ($\theta = 0.95$). We use the standard deviation (STDEV) of the ρ_i for all processing nodes to measure the load imbalance, i.e. $\sqrt{\frac{\sum_i (\rho_i - \overline{\rho})^2}{|N|-1}}$. Figure 4(a) shows the final load distribution for different values of max_1. $max_1 = 0$ means that dynamic load balancing is disabled. We can see when $max_1 >= 4$ the load is well balanced. No significant improvement can be made by using a larger max_1 value. Figure 4(b) illustrates the PR_{max} after the system is stable. It is computed by averaging on the values within 10 seconds. It is clear that the PR_{max} values are also similar when $max_1 >= 4$. Figure 4(c) shows the time it takes to converge to the final load distribution. There is not much difference between small and large number of partners. The above comparisons show that our system works well with a small max_1 value. As a larger number of partners would increase the runtime cost (such as transferring workload update messages, making load balancing decisions), we could keep the number to a small value and hence keep the cost low. In the subsequent experiments, we set $max_1 = 5$ and $max_2 = \lceil \frac{1}{2} max_1 \rceil$.



(a) Load imbalance (b) PR_{max} (c) Time to Converge

Fig. 4. Effect of various partner selection parameter



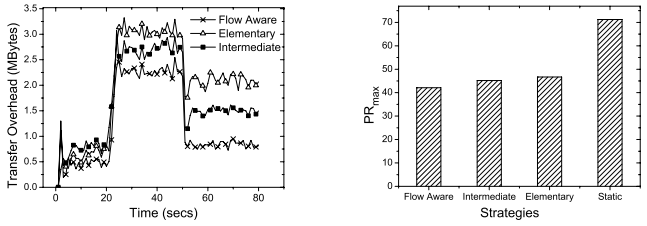
(a) PR_{max} (b) Transfer cost

Fig. 5. QF-based vs. OP-based

4.2 Load Selection Heuristics

The first experiment examines the necessity of imposing a distribution limit. This is done by comparing the QF-based (query fragment based) load balancing strategy with the OP-based (operator based) strategy proposed by reference [14]. The latter approach does not impose any distribution limit. We varied the number of operators per query fragment in our experiment. We ran the experiment under each case for 60 seconds simulation time and report the average values. We can see from Figure 5 that with more number of query operators, the PR_{max} value of the QF-based approach performs much better. Figure 5(b) may explain this phenomenon. The data transfer volume of the OP-based scheme increases quickly with more operators, because operators of a single query are migrated to too many site. Therefore the data streams are scattered over the network and leads to network congestion. On the other hand, the QF-based strategy still maintains small transfer overhead and hence it still performs well in data delay. Note that, by employing a distribution limit, an OP-based strategy can achieve better performance. However, as analyzed before, the cost to maintain such a limit would be higher than a QF strategy and such a scheme does not fit into a local load management strategy.

The second experiment examines the effectiveness of our flow-aware load selection strategy in maintaining good data flow locality. We impose an initially balanced load distribution over the processing nodes and use a uniform distribution to choose the querying streams S_k for every query q_k. At time $t = 20s$ we randomly select 4 nodes and then increase the input rates of the streams delegated to those nodes by 3 times. At $t = 50s$, the increased input rates drop back to their initial values. To show the effect of load selection strategy, we design another two approaches for comparison: (1) Elementary: the query fragments are selected in descending order of their load density. (2) Intermediate: the same as Elementary except foreign query fragments are given higher migration priorities than the native query fragments. In previous work, such as [9,14], data flow relationship is not considered. Hence their effects on the communication cost can be well represented by the Elementary algorithm. We compare the transfer overhead introduced by the three strategies against the static query fragment



(a) Transfer Overhead (b) PR_{max}

Fig. 6. On load selection strategies

allocation strategy, i.e. the initial placement scheme. The static strategy allocates the query fragments to their native nodes, hence its data flow transfer cost is minimum though it may incur very high data delay due to the unbalanced load allocation. We subtract the amount of transfer cost of the static strategy from those of the other three and then compare the extra transfer overheads of the three dynamic strategies over the static one.

From figure 6(a), we can see that the data flow aware strategy outperforms the other two at all stages of the experiment. Both Intermediate and Elementary, unlike the data flow aware strategy, fail to identify the neighborhood relationship of the query fragments. Intermediate is better than Elementary because it can differentiate between foreign query fragments and native query fragments and to some degree can help maintain data flow locality. At $t = 50s$ when the perturbed stream rates dropped back to the original value, all three strategies' transfer overheads are reduced. However, both Intermediate and Elementary cannot restore back to the state prior to the change. This is because both strategies are unable to identify their native nodes when migrating foreign query fragments. That means they would become worse and worse with the evolution of the system state while the data flow aware strategy is able to maintain a more stable state over time. Figure 6(b) shows the PR_{max} for all the four strategies. The values are calculated by averaging over the whole simulation time. The static strategy performed the worst simply because of the absence of load balancing strategy. Furthermore, the three dynamic strategies performed similarly. This is attributed to our heuristic to maintain a distribution limit for every query. Since processing load are similar for the three dynamic strategies due to the balanced load distribution, PR_{max} was similar for the three strategies. However, in the case when network traffic is so high that it approaches the bandwidth limit, the data flow aware strategy will do much better to avoid network congestion situation.

5 Conclusion

Distributed processing of continuous queries over data streams suffers from run time changes of system resource availability and data characteristics. Dynamic operator placement techniques are indispensable for a distributed stream processing system. In this paper, we formalized the problem and analyzed it by

building a cost model. As shown in our experiments, load imbalance can cause severe performance degradation and our proposed techniques can alleviate such degradation by dynamic load balancing. Our data flow aware load selection strategy can help restrict the scattering of data flows and lead to lower communication cost.

References

1. A. Arasu, et al. Stream: The stanford stream data manager. *IEEE Data Eng. Bull*, 26(1):19–26, 2003.
2. A. Ayad and J. F. Naughton. Static optimization of conjunctive queries with sliding windows over infinite streams. In *SIGMOD*, pages 419–430, 2004.
3. Y. Ahmad and U. Çetintemel. Networked query processing for distributed stream-based applications. In *VLDB*, pages 456–467, 2004.
4. D. Carney, et al. Monitoring streams - a new class of data management applications. In *VLDB*, pages 215–226, 2002.
5. D. Carney, et al. Operator scheduling in a data stream manager. In *VLDB*, pages 838–849, 2003.
6. S. Chandrasekaran, et al. Telegraphcq: Continuous dataflow processing for an uncertain world. In *CIDR*, 2003.
7. M. Cherniack, et al. Scalable distributed stream processing. In *CIDR*, 2003.
8. P. Pietzuch et al. Network-aware operator placement for stream-processing systems. In *ICDE*, pages 49, 2006.
9. M. A. Shah, et al. Flux: An adaptive partitioning operator for continuous query systems. In *ICDE*, pages 25–36, 2003.
10. U. Srivastava, et al. Operator Placement for In-Network Stream Query Processing In *PODS*, pages 250–258, 2005.
11. F. Tian and D. J. DeWitt. Tuple routing strategies for distributed eddies. In *VLDB*, pages 333–344, 2003.
12. S. Viglas and J. F. Naughton. Rate-based query optimization for streaming information sources. In *SIGMOD*, pages 37–48, 2002.
13. M. Willebeek-LeMair and A. P. Reeves. Strategies for dynamic load balancing on highly parallel computers. *IEEE Trans. Parallel Distrib. Syst*, 4(9):979–993, 1993.
14. Y. Xing, et al. Dynamic load distribution in the Borealis stream processor. In *ICDE*, pages 791-802, 2005.
15. Y. Zhou, et al. Adaptive reorganization of coherency-preserving dissemination tree for streaming data. In *ICDE*, pages 55, 2006.
16. Y. Zhou, et al. Dynamic load management for distributed continuous query systems. Unpublished manuscript, 2005. `http://www.comp.nus.edu.sg/~zhouyong/papers/op.html`.

Views for Simplifying Access to Heterogeneous XML Data

Dan Vodislav[1], Sophie Cluet[2], Grégory Corona[3], and Imen Sebei[1]

[1] CNAM/CEDRIC, Paris, France
[2] INRIA, Rocquencourt, France
[3] Xyleme, Paris, France

Abstract. We present *XyView*, a practical solution for fast development of user- (web forms) and machine-oriented applications (web services) over a repository of heterogeneous schema-free XML documents. XyView provides the means to view such a repository as an array, queried using a QBE-like interface or through simple selection/projection queries. Close to the concept of universal relation, it extends it in mainly two ways: (i) the input is not a relational schema but a potentially large set of XML data guides; (ii) the view is not defined explicitly by a query but implicitly by various mappings so as to avoid data loss and duplicates generated by joins. Developed on top of the Xyleme content management system, XyView can easily be adapted to any system supporting XQuery.

1 Introduction

For decades, companies have produced digital data such as notes, contracts, emails, progress reports, minutes, etc. This data constitute a mine of useful information that is largely unexploited. The advent of XML provides the opportunity to change that. Many enterprises are now considering storing their home data in XML repositories so as to be able to query them in a significant way, i.e., with tools more sophisticated than full text search engines. In this paper, we are addressing the problem of querying such repositories. More precisely, we are interested in developing, easily and quickly, a simple query API (web services) or user interfaces (web forms) over these repositories.

An important characteristic of the applications we are considering is that they deal with legacy data that have been mostly produced by human beings using standard text editors. As a result, the data is (i) poorly typed (well formed rather than valid XML) and (ii) highly heterogeneous (although documents have strong semantic connections). These features are particularly challenging since they call for sophisticated tools to ease the application programmer task while at the same time disabling most existing approaches.

The solution we propose borrows from the universal relation paradigm of the seventies [18]: *XyView* provides the means to easily view a set of heterogeneous XML documents as a single array that can be queried through simple selections and projections. Obviously, the context being XML, the array contains XML subtrees and is built using XQuery. But the fundamental differences with classical universal relations are the following:

R. Meersman, Z. Tari et al. (Eds.): OTM 2006, LNCS 4275, pp. 72–90, 2006.

– The array is not defined by one query but by a specification of how a simple selection-projection user query is to be translated into an XQuery.

  This difference is important. The problem with universal relations is that, unless the database schema has particularly nice properties which is rarely the case, projection operations generate many duplicates that are not always easy to remove. This is due to the join operations entering the definition of the universal relation. Alternatively, the join operations can also be the cause of missing information. This is usually solved by introducing outer-joins but at the cost of having to deal with null values.

  Note that these problems of data loss and duplicates may occur any time a view is defined as a structured query (SQL or XQuery).

  Our approach is not to define the view as a query but rather as a virtual set of queries that are generated on the fly to fit the user current requirements. In this way, we avoid incomplete or verbose answers.

– To deal with the complexity of the input data, we define views in two steps. The first deals with data heterogeneity and maps heterogeneous, but semantically connected documents into a target structure. At run time, this step generates unions. The second step corresponds to a standard view definition where data is aggregated. At run time, this leads to joins.

  Borrowing from a general wrapper-mediator architecture, our view model adds an intermediary level that (i) strongly structures the view by separating unions from joins, and (ii) provides homogeneous XML typing for the universal relation elements.

We implemented XyView as a set of tools on top of the Xyleme [19] XML repository, but it can easily be adapted to any system supporting XQuery. The XyView tools cover the view definition process but also automatic generation of web form applications and web services. Although its expressive power is limited as will be explained in this paper, XyView has proved its worth with several industrial applications.

The rest of the paper is organized as follows. The next section presents an example application scenario that illustrates the problem we are addressing. Section 3 describes the XyView model. Section 4 explores the expressiveness and some more subtle features of the model, then Section 5 describes the XyView system that is built on top of an XML repository. The final sections present related work and explore some future improvements.

2 Example Application Scenario and Motivation

The example that we present here is a drastic simplification of a real life application. A sports news company handles several types of news wires. The wires are well formed XML documents, with no global schema, that have been extracted from text files. These files have been edited by various local correspondents over the years, according to the company (mostly verbal) editing recommendations. The wires have different structures, depending on the sport and the kind of information they contain.

```
<!-- Document 1: National league result -->
<GameResult>
   <WireHeading> ... </WireHeading>
   <Description> Real Madrid 1 - Valencia 0
   </Description>
   <Date> 2004-05-22 </Date>
   <Team>
      <Name> Real Madrid </Name>
      <Scored> 1 </Scored>
      <Scorer><PlayerName> Zidane
              </PlayerName>
              <Count> 1 </Count>
      </Scorer>
   </Team>
   <Team>
      <Name> Valencia </Name>
      <Scored> 0 </Scored>
   </Team>
</GameResult>
```

```
<!-- Document 2: Inter-countries game -->
<Result Date="2004-03-15">
    <Summary> France 1 - Spain 1 </Summary>
    <Scorers>
        <Player Goals="1">
            <Name> Zidane </Name>
            <Country> France </Country>
        </Player>
        <Player Goals="1">
            <Name> Raul </Name>
            <Country> Spain </Country>
        </Player>
    </Scorers>
</Result>
```

```
<!-- Document 3: Sports encyclopedia -->
<Encyclopedia>
  <Football>
    <Player><Name> Zidane </Name>
            <Biography>...</Biography>
    </Player>
    ...
  </Football>
  ...
</Encyclopedia>
```

Sample queries on football documents

Q_1: "Games in which Zidane scored more than once"

Q_2: "The biography of Zidane"

Q_3: "Biographies of scorers from games on 2004-09-08"

Fig. 1. Examples of documents and queries

For ease of understanding, we show in Figure 1 only two such wires about football (soccer) in a simplified form. The first considers results from national leagues (e.g., Document 1), and the second results from international games (e.g., Document 2). The news company wants to build an application that queries through simple web forms the various football results wires and a sports encyclopedia with detailed information about football players (Document 3).

The application manipulates documents whose structures are similar, but not necessarily identical, to Documents 1, 2 and 3. Notably, other documents may have more or less information. These three kinds of documents are stored in a single XML content management system in collections whose respective identifiers are NationalURI, InternationalURI and EncyclopediaURI.

The application queries, as those in Figure 1, may concern football results (Q_1), player biographies (Q_2), or both (Q_3).

These apparently simple queries are in fact rather hard to program in XQuery as illustrated by Figure 2 for Query Q_3 (issues regarding the typing of results are discussed in Section 4, we assume here that queries return simple strings).

Our objective with XyView is to optimize the productivity of graphical user interface programmers, who are not database experts, by allowing them to view the database as something as simple as a query form consisting of fields that can be used to filter or extract data. In the meantime, we want to simplify as much as possible the task of creating and maintaining such views.

The main contributions of this paper are the following:

```
union(
   For $doc1 in collection(NationalURI),
       $var1 in $doc1/GameResult,
       $doc2 in collection(EncyclopediaURI),
       $var2 in $doc2/Encyclopedia/Football/Player,
       $var3 in $var2/Biography
   Where $var1/Date = xs:date('2004-09-08') and
         $var1/Team/Scorer/PlayerName = $var2/Name
   Return string($var3),
   For $doc1 in collection(InternationalURI),
       $var1 in $doc1/Result,
       $doc2 in collection(EncyclopediaURI),
       $var2 in $doc2/Encyclopedia/Football/Player,
       $var3 in $var2/Biography
   Where $var1/@Date = xs:date('2004-09-08') and
         $var1//Player/Name = $var2/Name
   Return string($var3) )
```

Fig. 2. Query Q_3 expressed in XQuery

- A view model (XyView) that provides a universal relation-like access to heterogeneous, schema-free XML data, freeing the user from manipulating complex schemas and query languages.
- A method for avoiding the drawbacks of query-based views: joins in the view definition produce duplicates or data loss; expressing the view query for heterogeneous data is difficult; queries on the view produce nested queries, that are harder to optimize. Instead, views are defined by simple mappings and join conditions, easy to create and to maintain with graphical tools. A simple, effective and scalable query translation algorithm produces equivalent XQuery, with no useless duplicates, no data loss and no nesting.
- A view structure model, organized on several levels, adapted to heterogeneous, schema-free XML data.
- A set of tools for creating views, for rapid development and for automatic generation of web forms and web services.

3 The XyView Model

In XyView, views are defined by a set of mappings and join conditions that specify how a simple selection-projection user query is translated into an XQuery. This approach overcomes the problems of query-based views, as explained later. The view definition is equivalent to a virtual set of flat and easily optimizable queries that are generated on the fly to fit the user current requirements. Notably, given the appropriate view specification, the query of Figure 2 would be generated at run time by XyView to answer Query Q_3.

This results in a simpler definition and maintenance of views, using intuitive graphical editors. Given the complexity of view queries caused by data heterogeneity, this is a crucial advantage for the view designer.

Also, in order to cope with heterogeneous data, XyView adds an intermediary level in the view definition process. To the physical and view schemas, we add

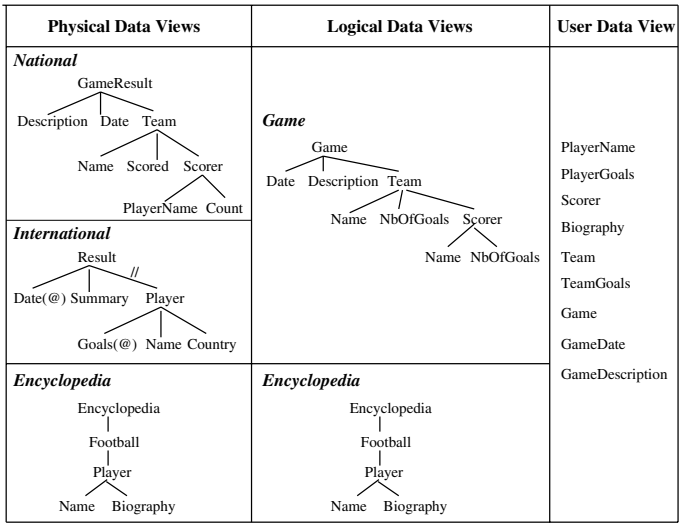


Fig. 3. From Trees to Table

logical schemas whose purpose is to provide homogeneity to semantically related data. More precisely:

1. The first level deals with schema-free data, by defining *physical data views* that summarize XML access paths to useful information in documents.
2. The second level deals with heterogeneity, by defining integrated *logical data views* over *unions* of physical data views with similar contents.
3. The third level defines the *user data view* as *joins* between logical data views.

Figure 3 illustrates this three level definition. It is built using the sample data introduced in Section 2. On the right handside is the **user data view**. It consists of a set of **"concepts"** that the user wants to query. Concepts are *typed* by the view designer. For instance *PlayerGoals* and *TeamGoals* are integers, *GameDate* is of type date, the other concepts are considered as XML strings (or elements, as will be explained in the next section).

As is the case with universal relations, the query language supported at this level consists of selections and projections. For instance, Query Q_3, that returns biographies of scorers from games on 2004-09-08 consists of a selection on *GameDate* = *2004-09-08* and a projection on *Biography*.

On the left handside of the figure, are the **physical data views (PDV)**. They represent the data as it is stored in the repository. In the example, there are three physical data views (*National*, *International* and *Encyclopedia*), the first two representing respectively local and international soccer games results, the other a sport encyclopedia. The trees are **data summaries**, i.e. trees gathering useful access paths to data elements in the XML documents. Similar to Lore data guides [8], they are generated by the system to cope with the fact that many documents are simply well-formed and do not come with a schema.

In Xyleme, these summaries are generated at loading time, there is one summary per distinct root element. XyView also provides a tool to extract these summaries from a set of documents. In both cases, we use an incremental algorithm that takes all the XML paths in documents and extends the existing data summary paths with new subpaths. Note that the algorithm does not care about data types. It is the view designer who associates types to view concepts. More will be said on this topic in the sequel.

When designing the view, one can edit data summaries to remove branches that are useless for the application or to create shortcuts in long branches by using a descendant (//) connection between two nodes. E.g., the subtree *Wire-Heading* has been removed from the structure of Document 1, while the structure of Document 3 only features the *Football* element, other sports having been discarded. Also, the *Biography* element is not detailed in the PDV, because its internal structure is not useful for the application. PDV *International* contains an example of shortcut: element *Scorers* has been discarded from the path to *Player*, because it is useless and removing it introduces no ambiguity; the edge leading to *Player* is marked with //. This simplification eases the view design process, by keeping only useful access paths from possibly cumbersome document structures. Also, // shortcuts significantly improve query processing of the final XQuery, by reducing the number of structural conditions to check.

In the center of the figure, we have gotten rid of the soccer games results heterogeneity by introducing so-called **logical data views (LDV)**. Logical data view *Game* unifies in a single structure game results from documents described by PDVs *National* and *International*. Note that the second LDV (*Encyclopedia*) is a duplication of the corresponding PDV. In real life, we do not duplicate data views, we did it here for the sake of clarity.

We now illustrate how one goes first from physical to logical then to user data views and the queries that are associated to each level. Next, we further detail how user queries are translated into queries against the repository.

3.1 From Physical to Logical Data Views

A physical data view consists of a data summary tree and a set of so-called *clusters* in which we find documents conforming to the summary (there may be documents conforming to other summaries as well). A cluster is the unit in which we store documents and provides an entry point in the repository. It is queried in XQuery as a collection of documents, by using the *fn:collection* function on the cluster URI. For the sake of clarity, in the following examples we consider a single cluster for each PDV.

Semantics: a PDV P is a view over a collection of documents $Coll(P)$ (the cluster), whose schema is a data summary tree $Tree(P)$. For each p node of $Tree(P)$, let $path(p)$ be the path from the root of $Tree(p)$ to p. *The interpretation* of p is the set of XML elements that match $path(p)$ in some document of $Coll(P)$, i.e. the result of the XQuery expression $Coll(P)/path(p)$.

$Eval(p) = XQuery(Coll(P)/path(p))$

Notations: for x and y nodes in the same tree, $LCA(x, y)$ denotes their lowest common ancestor and $ancestor(x, y)$ is true if x is ancestor of y.

The interpretation of a tuple $(p_1, ..., p_k)$, $p_i \in Tree(P)$ is:

$Eval(p_1, ..., p_k) = \{(e_1, ..., e_k) \mid \exists doc \in Coll(P), \forall i \in \{1, ..., k\},$
$e_i \in Eval(p_i) \bigcap Elem(doc), \forall j, l \in \{1, ..., k\},$
$\exists e_{jl} \in Eval(LCA(p_j, p_l)), ancestor(e_{jl}, LCA(e_j, e_l)) \bigcap Elem(doc)\},$

where $Elem(doc)$ is the set of XML elements of document doc.

The meaning is that a tuple's elements must belong to the same document and must be the "closest" possible. Closeness is expressed wrt the PDV schema: any two tuple elements e_j, e_l must be at least as close as the corresponding PDV nodes in the schema p_j, p_l, i.e. e_j and e_l must have a common ancestor at the level of $LCA(p_j, p_l)$.

We will show in Section 3.3 how the query translation algorithm guarantees closeness by introducing XQuery variables for the LCA nodes.

A logical data view is an annotated data summary. The annotations represent the correspondence (mappings) between physical and logical data views. This is illustrated on the left side of Figure 4 for LDV *Game* and PDVs *National* and *International*. Note that to each node in the LDV data summary is associated the set of corresponding nodes in the physical data views. To keep the figure readable, only mappings for LDV nodes *Game* and *Date* are illustrated.

Mappings between LDVs and PDVs are based on correspondences between LDV and PDV tree nodes. By identifying a node in a tree with its path from the root, one can note that this approach to representing correspondence between trees is close to the *path-to-path mappings* used in [4]. The *restriction* we add - an LDV node can be mapped to *at most one node in the same PDV* - makes sure that translation from LDV to PDV in query processing is unique. A PDV not respecting the restriction can always be split into several "correct" PDVs.

Compared to classical query-based methods to define correspondences between schemas, the simplicity of our node-to-node mappings approach provides several advantages.

- In many cases, mappings can be semi-automatically generated by relying on the semantics carried by a sequence of tags (see [15,17]).
- The process of creating these mappings can easily be supported by a graphical interface and they are easier to maintain than query-based mappings.
- Such node-to-node mappings are easy to reverse, therefore the view model can be seen both as *global-as-view* (providing easy query translation) and *local-as-view* (providing easy update).
- In Section 4, we will see that such mappings can easily be extended in order to support a richer semantics.

Semantics: a LDV L is a view over a set of PDVs $PDV(L)$, whose schema is a tree $Tree(L)$. The interpretation of a tuple $(l_1, ..., l_k)$, $l_i \in Tree(L)$ is:

$Eval(l_1, ..., l_k) = \bigcup_{P \in PDV(L)} Eval(p_1, ..., p_k)$, $p_i \in P$ is mapped to l_i

This union semantics is defined in two variants: *strict matching*, where only PDVs containing mappings to all the l_i are considered, and *relaxed matching*,

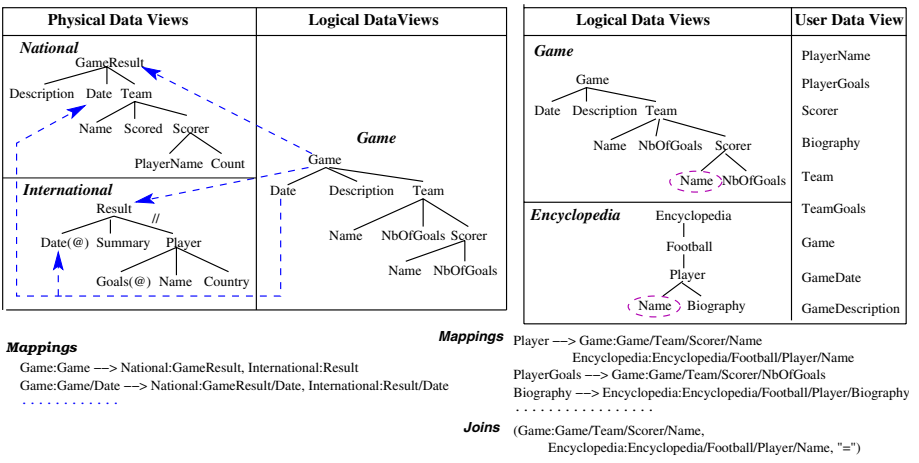


Fig. 4. From physical to logical data view, from logical to user data view

where all the PDVs are considered, but only incomplete tuples, based on existing mappings, are built in each PDV.

The algorithm in Section 3.3 translates straightforwardly a query against a LDV into a union of queries against its corresponding PDVs by transforming paths from the LDV query into the corresponding paths in each PDV.

3.2 From Logical to User Data Views

A user data view consists of a set of typed concepts, their correspondence with nodes in the logical data views and a set of predicates that are used to join the logical data views (in the example, a single join predicate is defined). This case is illustrated on the right side of Figure 4. Each concept has at least a mapping to some LDV node, concept *Player* being the only one mapped to both LDVs. The join predicate specifies the joined LDV nodes and the join operator ('=' in our example). If several join predicates connect two LDVs, the global join condition is the conjunction of the individual predicates.

The semantics of a user data view is described by the query translation algorithm below. Roughly speaking, the interpretation of a tuple of concepts is a n-ary join of partial tuples of LDV nodes, found in LDVs through mappings.

We now explain in details the translation algorithm from user queries to physical queries, via logical queries.

3.3 Translating User Queries

Let us now consider Query Q_3 as an example to illustrate the translation algorithms. It involves a join between the two LDVs in order to return the biographies of scorers from games played on 2004-09-08.

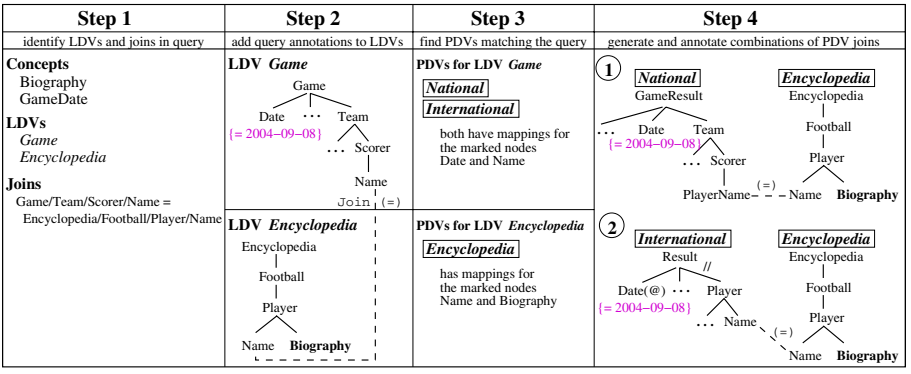


Fig. 5. First steps for translating Query Q_3: `Select` *Biography* `Where` *GameDate*=2004-09-08

Definition 1. *A user query in XyView has the form*
`Q: Select` c_1, ..., c_n
`Where` $cond_1(c'_1)$ `and` ... `and` $cond_m(c'_m)$
where c_i *and* c'_j, $i = 1, ..., n$, $j = 1, ..., m$, *are user view concepts and* $cond_j$ *are predicates over a single concept.*

Figure 5 illustrates the translation of Query Q_3 into XQuery. The translation algorithm for a user query `Q` consists of *five steps*:

1. Identify LDVs and joins involved in user query `Q`;
2. Produce a tree representation of `Q` based on the LDV trees;
3. For each LDV annotated tree, find the subset of PDV trees that match `Q`;
4. Generate all combinations of joins between PDVs;
5. Generate the final XQuery by unioning the combinations of step 4.

Step 1. One identifies the sets of concepts (C_Q), LDVs (L_Q) and joins (J_Q) involved in `Q`. They are the following:

$\mathbf{C}_Q = \{c_1, ..., c_n\} \bigcup \{c'_1, ..., c'_m\}$

$\mathbf{L}_Q$ basically contains only LDVs involved in the query, i.e. having nodes mapped to some concept in C_Q, **in order to discard useless joins**. Several semantics are implemented in XyView for L_Q. One possibility is *to remove redundant LDVs from* L_Q, i.e. those contributing with data already provided by other LDVs. Another possibility is *to add LDVs that appear along some join path between LDVs in the initial* L_Q.

$\mathbf{J}_Q = \{j=((l_1, path_1), (l_2, path_2), op) \mid j$ is a join, $l_i \in L_Q$, $path_i$ is a path in l_i, i=1,2, *op* is the join predicate$\}$, joins between members of L_Q.

In the example (Figure 5, Step 1),
C_{Q3}={`Biography`, `GameDate`}
L_{Q3}={*Game*, *Encyclopedia*}, because *Game* has a node mapped to concept

`GameDate` and *Encyclopedia* has a node mapped to concept `Biography`
J_{Q3}={((*Game*, Game/Team/Scorer/Name), (*Encyclopedia*, Encyclopedia/Football/Player/Name), '=')} includes the only existing join, because L_{Q3} contains both joined LDVs.

Step 2. One adds query annotations to nodes of LDV trees from L_Q.

Definition 2. *The following query node annotations are defined for LDV nodes:*
- isProjected, *a boolean, true iff the node is mapped to a projected concept;*
- condSet, *a set of condition predicates composed of predicates* cond_j *of* `Q` *over a concept mapped to the node;*
- isJoined, *a boolean, true iff the node occurs in* J_Q.

Definition 3. *A LDV tree node is called* a marked node *if* isProjected = *true or* condSet$\neq \emptyset$ *or* isJoined = *true.*

Figure 5, Step 2 shows query annotations added to LDV trees for Q_3. The projected node, **Biography** (*isProjected*=true), is in bold font; join nodes (*isJoined* =true) are connected through a dashed line; and the selection node (*condSet* $\neq \emptyset$) is annotated with the set of condition predicates. We removed nodes that are not involved in the query.

Step 3. For each $l \in L_Q$, the set of PDVs matching `Q` is

$$\mathbf{P}_{Q,l} = \{p \mid p \text{ is a PDV}, \forall\, n \text{ marked node of } l \Rightarrow \exists\, n' \text{ of } p \text{ mapped to } n\}$$

This corresponds to the *strict matching* semantics presented in Section 3.1. In the example, we obtain $P_{Q3,Game}$={*National*, *International*} and $P_{Q3,Encyclopedia}$={*Encyclopedia*}, because all the marked nodes in Step 2 are mapped into all the corresponding PDVs. This is not always the case; suppose that game dates are lacking from PDV *National*, in that case *National* must be removed from $P_{Q3,Game}$, because *Date* is a marked node in LDV *Game*.

Step 4. One computes all the combinations Comb_Q of joins between PDVs found at Step 3. Note that joins may be n-ary (in opposition to binary), this may occur when the schema has more than two LDVs.

For each *comb* $\in \text{Comb}_Q$, the PDV nodes get the same query annotation as the LDV nodes to which they are mapped. A PDV node mapped to no LDV node gets *isProjected*=false, *condSet*=$\emptyset$ and *isJoined*=false.

Also, for each *comb* $\in \text{Comb}_Q$, the set J_{comb} of join conditions is obtained from J_Q by replacing the LDV nodes by the corresponding PDV nodes.

In our example, there are two such combinations, shown in Figure 5, Step 4.

Step 5. For each *comb* $\in \text{Comb}_Q$, representing a join between PDVs, a *For-Where-Return* query is generated. The final XQuery is obtained by unioning all these join queries. For our example, the final result is presented in Figure 2.

The algorithm for generating a *For-Where-Return* query creates *For/Where/Return* clauses as concatenations of the clauses generated for each individual PDV. To the *Where* clause, one must also concatenate the join conditions from J_{comb}.

```
ForClause(pdv) → String
    pdv.variable = GenerateVar()
    pdv.varNodeList =
          VariableGen(pdv.root)
    forClause = concat(pdv.variable,
          ' in collection(',
          pdv.collectionURI, ')' )
    for each n_i ∈ pdv.varNodeList repeat
       forClause = concat(forClause,
          ',', n_i.variable, ' in ',
          AncestorVarAndPath(n_i, pdv))
    end for
    return forClause
end ForClause
```

```
VariableGen(n) → NodeList
    for each n_i ∈ n.children repeat
       varNodeList_i = VariableGen(n_i)
    end for
    childrenVarNodeList = concat(varNodeList_1, ...)
    if n.isProjected then
       n.variable = GenerateVar()
       n.markedAncestor = true
    else maChildren = NbMarkedAncestor(n.children)
       n.markedAncestor = n.condSet ≠ ∅ or
          n.isJoined or maChildren > 0
       if maChildren > 1 then
          n.variable = GenerateVar()
       else n.variable = null
       end if
    end if
    if n.variable ≠ null then
       return concatList([n], childrenVarNodeList)
    else return childrenVarNodeList
    end if
end VariableGen
```

Fig. 6. "For" clause generation for a PDV

Let us describe now the algorithms **For**, **Where** and **Return** that generate the corresponding clauses for a single annotated PDV.

The *For* clause defines variables and access paths to queried data in the PDV. Variable generation respects the following rules:

Rule 1. *A variable is defined for each projected node in the PDV.*
Rule 2. *For any two marked nodes of a PDV, there is a variable definition for their lowest common ancestor in the PDV tree.*

Rule 2 ensures the closeness semantics for the XML elements addressed by the query, i.e. those corresponding to marked nodes. For instance, it ensures that in query Q_3 the date and the scorer name belong to the same game.

Considering the annotated PDV *National* in the first combination in Figure 5, Step 4, there are only two marked nodes *Date* and *Name*, none of them projected. Then, the only variable element defined on national games is that of *GameResult*, their lowest common ancestor.

The algorithm for generating the *For* clause for a single PDV is presented in Figure 6. It adds some new query annotations to tree nodes:

- *variable*, the name of the variable generated for the node, if any;
- *markedAncestor*, a boolean, true iff the node's subtree contains at least a marked node.

It also adds the following annotation for each PDV:

- *variable*, the variable name for documents that match the PDV;
- *varNodeList*, the list of variable nodes in the PDV, following the order of variables in the *For* clause.

The *ForClause* function uses the *VariableGen* function to obtain the ordered list of variable nodes in the PDV. The *For* clause starts by defining the document variable iterating in the collection associated to the PDV. All variable names are generated by calls to function *GenerateVar*, that returns unique variable

names. The rest of the *For* clause defines variables for each variable node. The *AncestorVarAndPath* function searches for the first variable ancestor of the node, then returns this variable concatenated with the path from this ancestor to the node. If no variable ancestor exists, one uses the PDV variable and the path from the root to the node.

The *VariableGen* function builds the list of variable nodes in the subtree of the parameter node n, but also annotates with *variable* and *markedAncestor* each node in the subtree. First, it recursively builds the variable node lists for each child of n, then concatenates these lists. Then, it must decide if n is a variable node or not; if not, the result is the concatenated list from children, else n is added in front of this list. This step produces a consistent order for the *For* clause, because a node is always placed before its descendants.

Rules 1 and 2 are used to decide if n is a variable node. This is true either if n is projected, or if it has at least 2 children being *markedAncestor*. In the latter case, it is easy to demonstrate that n is the LCA of marked nodes from the subtrees of these children. Function *NbMarkedAncestor* returns the number of nodes being *markedAncestor* in the parameter list. Also, n is itself *markedAncestor* if it is projected or if it has at least one child being *markedAncestor*.

Note that **the algorithm does not generate useless variables**, only marked nodes (i.e. needed in the user query) are connected through variables on the LCA.

The algorithm for the *Where* clause produces a *conjunctive* condition. For each PDV node with $condSet \neq \emptyset$, it generates a condition predicate for each element of *condSet*. The node is identified by the path from its first variable ancestor. Note that well-typed constants are generated, by using the types of the view concepts. A similar algorithm is used to generate join conditions.

The *Return* clause describes the query result, using the variables of projected PDV nodes. Several choices for the XML type of the result are possible in XyView; they are discussed in the next section.

4 Deeper Inside XyView

4.1 Duplicates and Data Loss

So far, we have presented views as providing a flat, relational-like, representation of arbitrary XML trees. The flattening is performed by accessing nodes through path expressions (preserving closeness through variables on the lowest common ancestor) and applying the XQuery **string()** operation on the projected nodes. The transformation from logical to user data views then corresponds to a simple sequence of join operations between results of path expressions followed by projection/map operations. In the transformation from physical to logical data views, joins are replaced by unions. The main difference with a standard view mechanism is that the view query is not defined *a priori* but rather in an opportunistic way, depending on the user query, so as to avoid duplicates and information loss that would be generated by unnecessary joins and variables.

Therefore, XyView differs from any standard view mechanism relying on query composition: **a XyView view is not defined by a query and is not equivalent to a query**. To see why query-based views may be problematic, let us have a closer look to this possibility.

If we consider that a XyView view is defined by a query, this query has to provide a full view over the various documents structures for all the view concepts. Thus, it would naturally feature (i) all the possible join and union operations in the view, as well as (ii) variables for internal nodes so as to preserve closeness of all concept elements belonging to the same subtree. A good candidate for the view query is *the user query that projects all the view concepts*, i.e. its translation through the previous algorithm.

A query on this view will face the following problems:

- **Data loss:** the join operations would make it impossible to return e.g., the biography of players who are not part of some games. The problem is that the join with LDV *Game*, that is part of the view query definition, is useless when querying player biographies. Data loss can be solved by introducing outer-joins, but they generate null values and the need to deal with them.
  In a similar way, but with no apparent reasonable XQuery solution, useless variables may be responsible for data loss. E.g., the view query contains a variable on the internal node representing scorers (required to connect scorer name and goals). If it cannot be instantiated, the corresponding parent element would be discarded, i.e. games with no scorer would be discarded from all results.
- **Duplicates:** useless joins would also lead to unnecessary duplicates, e.g. by returning several biography occurrences for most players (one for each occurrence as a scorer). The same is true for useless variables, coming from view concepts not addressed in the query. A result being produced for each new binding to the tuple of all the variables, useless variables produce duplicates for the useful variables in the result.
  Duplicates can be eliminated through **distinct** operations, but (i) it is sometimes very difficult to distinguish between good (existing in data) and bad duplicates, and (ii) distinct operations have a cost (notably when the desired order is not that required by the distinct operation).

4.2 View Customization

More expressive power is added to XyView through *LDV node annotations*, which allow ***customizing*** query translation, notably by (i) typing results using tree structures and transformation functions rather than returning flat results and (ii) adding selections to the view.

Tree results can be obtained in XyView in several ways. The simplest one consists in typing results according to the PDVs, i.e., returning trees as they are stored in the repository. Note that this solution leads to heterogeneous results.

For instance, consider a new query, close to Query Q_3, modified to ask *scorers* (name and number of goals) from games on 2004-09-08.

Q_3': "Scorers from games on 2004-09-08"

This query would be translated as follows:

```
union(
For $doc1 in collection(NationalURI),
    $var1 in $doc1/GameResult, $var2 in $var1/Team/Scorer
Where $var1/Date = xs:date('2004-09-08')
Return $var2,
For $doc1 in collection(InternationalURI),
    $var1 in $doc1/Result, $var2 in $var1//Player
Where $var1/@Date = xs:date('2004-09-08')
Return $var2)
```

Note that, in that case, the result of the query is heterogeneous, featuring scorers as they are stored in *National* and *International* PDVs. This solution is well adapted for *ad hoc* queries expressed by users who want to see the data as it has been produced. Also, it is an interesting semantics for the view designer who, in the preliminary phase, wants to get some information about data types. However, if the results are to be fed to an application or if the end users are not aware of the data as it is stored, we need to provide an alternative.

The second solution provides the means to type results according to the LDVs. This can be performed in a simple way by associating to each leaf node the full text (or typed atomic value) corresponding to the physical nodes to which they are mapped. It is then simple to re-construct the elements as they are defined in the logical data view. Query Q_3' becomes:

```
union(
For $doc1 in collection(NationalURI),
    $var1 in $doc1/GameResult, $var2 in $var1/Team/Scorer,
    $var3 in $var2/PlayerName, $var4 in $var2/Count
Where $var1/Date = xs:date('2004-09-08')
Return <Scorer>
         <Name>string($var3)</Name>
         <NbOfGoals>xs:integer(string($var4))</NbOfGoals>
       </Scorer>,
For $doc1 in collection(InternationalURI),
    $var1 in $doc1/Result, $var2 in $var1//Player,
    $var3 in $var2/Name
Where $var1/@Date = xs:date('2004-09-08')
Return <Scorer>
         <Name>string($var3)</Name>
         <NbOfGoals>xs:integer(string($var2/@Goals))</NbOfGoals>
       </Scorer>)
```

Note that new variable definitions are generated in the *For* clause, in order to access PDV nodes necessary to build the LDV subtree for the scorer. Results of both unioned queries have the same type, given by the LDV. Note also that the LDV subtree may not have all the subelements if some of them are not mapped into the PDV.

Both typing solutions above can be performed automatically by activating the appropriate translation option. Still, there are some cases where we want to achieve more sophisticated typing. For instance, we may want to add some PCDATA or attributes to the internal nodes. To do this, the view designer further annotates the nodes of the data summaries with ***transformation functions***.

Consider for instance the previous example, but in which we want to add an attribute called *source* that gives the URI of the source document. Suppose that

the document URI can be obtained by applying the **element2URI(*element*)** function to some element of that document. Node *Scorer* in the LDV must be annotated as follows:

```
Return: <Scorer source=element2URI($$)> $1 $2 </Scorer>
```

As in Yacc, we use $$ to signify the current node (*Scorer*), $1 and $2 to represent its first (*Name*) and second (*NbOfGoals*) children. Typing of children $1 and $2 is recursively done following the same method. For instance, this solution allows selecting in the result only part of the node's subelements. Also, we use # to represent user input, i.e. the list of constant values in the user query coming from conditions on the current node, and may be used as an argument in transformation functions.

Note that producing flat string results, or PDV subtrees, or LDV subtrees is equivalent, respectively, to the following annotations for the *Scorer* LDV node:

```
Return: string($$)                //flat
Return: $$                        //PDV subtree
Return: <Scorer> $1 $2 </Scorer>  //LDV subtree
```

Selections to the view may also be specified through annotations. Suppose that we want to discard from our view all the games before 2000. This can be simply done through a new type of node annotation: selection predicates. In the example, the following annotation must be added to the *Date* node in the LDV:

```
Where: $$ >= xs:date('2000-01-01')
```

If the user query concerns some LDV, all the selection predicates of that LDV are added to the conjunctive *Where* clause of the generated XQuery.

These modifications are easily added to the algorithm detailed in the previous section. However, note that, even with these additions, the view mechanism is far from supporting all the features of XQuery. Notably, XyView does not provide grouping/nesting, sorting or disjunctive join predicates. Some of the missing features can be supported by the client program, using e.g., stylesheets. In any case, there is a necessary tradeoff between ease of use and expressive power. So far, the tool has proven useful for most applications.

5 The XyView System

XyView has been implemented as a set of tools for rapid development of web applications over the Xyleme XML repository. Yet, XyView is not dependent on Xyleme and can be easily adapted to any content management system that supports XQuery. The XyView system is composed of the following modules:

- *A view editor* that enables visual creation and modification of XyView views.
- *A run-time environment* that provides a simple API for using XyView views in user- (web forms) or machine-oriented (web services) web application.
- *A web-form application generator* that provides a graphical environment for creating simple web-form applications over the Xyleme repository.

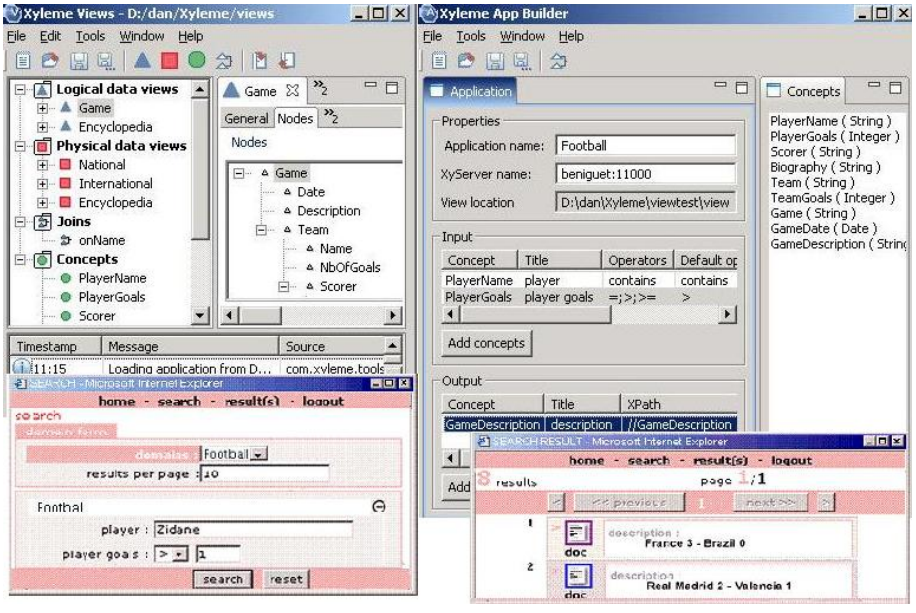


Fig. 7. XyView editor and web-form generator

The view editor (upper-left window in Figure 7) is a graphical tool enabling simple and intuitive creation of each view component: PDVs (using data summary extractors), LDVs, concepts, mappings, joins, etc. Views are saved in a persistent form, as a set of XML files.

The run-time environment provides a simple Java API for using XyView views in programs. The main functionalities provided by the XyView API are: (i) creating/modifying a view, (ii) loading/saving a view from/in its persistent form, (iii) building user queries against the view, (iv) translating a user query into an equivalent XQuery.

Note that XyView simply translates the user query into XQuery and does not interfere afterwards in the communication between the application and the XML repository. This architecture has the advantage of minimizing the dependency between XyView and the underlying XML content management system, allowing easy adaptation of XyView to any system supporting XQuery.

The web-form application generator (upper-right window in Figure 7) enables complete development of simple applications for end-users that query the Xyleme repository through a web-form interface. It provides a graphical interface that helps the application programmer to choose a XyView view, then to formulate queries on the view concepts. The web-form built in Figure 7 is based on the Football example view; it asks user input for conditions on concepts *PlayerName* and *PlayerGoals* and displays concept *GameDescription*. The system automatically generates the HTML query form (bottom-left window) and the application servlets producing the query report (bottom-right window).

Several applications were developed with XyView on top of Xyleme, to integrate more or less heterogeneous, semi-structured data sources, covering domains such as news publishing, financial reports, press archives, etc. Examples used in this paper are summaries of one such application involving about 50 PDVs and 11 LDVs (an encyclopedia, 10 different sports and an average of 5 kinds of wires for each). The original documents were well annotated ASCII files transformed into XML documents using a dedicated tool.

6 Related Work and Conclusion

Various approaches for simplifying query formulation over XML data were proposed. Systems like XQBE [1] and Xing [6] use visual specification of XML queries based on tree patterns. But even if it is simpler to express queries graphically than in XQuery, the user must handle XML structures, express joins, etc. Other systems allow writing queries with minimal knowledge about the structure of documents: keyword search in XML data [5,12,9] or tag and keyword search [14]. Such systems are not adapted for application development over heterogeneous XML documents, because of their limited expressive power (e.g. no joins) and lack of precision and/or meaningfulness.

XyView's approach of adapting the universal relation paradigm [18] to simplify query formulation fits well the needs of both end user and application development. Querying XyView views is very simple, it guarantees precision, meaningfulness of results and minor processing overhead. The price to pay is the view designer's effort to create and maintain the view. But the XyView model is not query-based and rather borrows from mediator-like [2,13,4,7] or P2P [10] XML data integration systems, to define views through basic one-to-one mappings, like those used in [4,7]. This allows the use of graphical tools, which greatly simplifies the view designer's task.

An alternative approach is to shred XML in relations, physically (like many RDBMS today) or virtually ([11]), then to create a relational view on top. This solution may work efficiently for homogeneous XML documents, with no structural variation and when XML is really stored in tables. Our application context is more general; we build views over heterogeneous and schema-free XML, stored in any system supporting XQuery.

Among the tools for rapid development of web applications over XML data, Qursed [16] is close to our application development context. Qursed enables rapid development of user-oriented applications over XML data, based on web query forms and reports. Its main module is a visual editor, which roughly takes an HTML query form (input for the user), a report template (output for the user) and an XML Schema describing the data. The programmer defines mappings between input query fields and XML data, then between XML data and report output. Qursed is similar to our XyGen web-form application generator, but can produce more sophisticated output reports. Yet, Qursed is not appropriate for heterogeneous, schema-free XML data. It needs XML Schema for data and

can handle a single document schema in the same application. Also, Qursed is designed for user-oriented applications, but not to program web services.

In the same category of tools, BEA Liquid Data [3] provides an advanced environment for data integration and web application development. It overcomes the limitations of Qursed by defining data views over several schemas connected through joins. Unions are also possible, but the method to define them is unnatural, based on a cloning of data view elements. Beyond the fact that this complex tool focuses on specialized programmers, its support for heterogeneous schema-free XML documents has several limitations: (i) data sources must provide a schema, (ii) views are defined by queries, with all the problems of useless joins and variables, (iii) one cannot reasonably mix in the same data view several joins and unions, etc. Even if the latter problem can be bypassed by chaining several data views, this results in bad query processing performance.

Through its simple programming interface that removes the need to work with XQuery and XML schemas, XyView increases the productivity of programmers who implement query interfaces on top of a heterogeneous, schema-free XML repository. The view designer's task is highly simplified by the intuitive representation of views (set of mappings), manipulated through graphical editors. The query translation algorithm is simple, effective and scalable, it avoids useless duplicates, data loss and unnecessary nesting. Tools for Java programming and for automatic generation of web-form applications complete the XyView environment.

Although the tool does not provide the full expressive power of XQuery, it has proven sufficient for many industrial applications. Furthermore, the possibility to customize the query generation algorithm by adding functions to the view specification opens interesting perspectives in terms of expressive power. We illustrated this by considering typing and selections, we plan to further explore this mechanism to add some needed functionalities such as aggregation.

References

1. E. Augurusa, D. Braga, A. Campi, and S. Ceri. Design and Implementation of a Graphical Interface to XQuery. *Proceedings ACM Symposium on Applied Computing*, pages 1163 – 1167, 2003.
2. C. K. Baru, A. Gupta, B. Ludäscher, R. Marciano, Y. Papakonstantinou, P. Velikhov, and V. Chu. XML-Based Information Mediation with MIX. *Proceedings SIGMOD*, 1999.
3. BEA Liquid Data. *http://www.bea.com*.
4. S. Cluet, P. Veltri, and D. Vodislav. Views in a large scale XML repository. *Proceedings of the 27th VLDB Conference*, pages 271–280, 2001.
5. S. Cohen, J. Mamou, Y. Kanza, and Y. Sagiv. XSEarch: A Semantic Search Engine for XML. *Proceedings VLDB*, 2003.
6. M. Erwig. Xing: A Visual XML Query Language. *Journal of Visual Languages and Computing*, pages 5–45, Februray 2003.
7. I. Fundulaki, B. Amann, C. Beeri, M. Scholl, and A.-M. Vercoustre. STYX: Connecting the XML Web to the World of Semantics. *Proceedings EDBT*, pages 759–761, 2002.

8. R. Goldman and J. Widom. DataGuides: Enabling Query Formulation and Optimization in Semistructured Databases. *Proceedings of the 23rd VLDB Conference*, pages 436–445, 1997.
9. L. Guo, F. Shao, J. Shanmugasundaram, and C. Botev. XRANK : Ranked keyword search over XML documents. *Proceedings SIGMOD*, 2003.
10. A. Halevy, Z. Ives, P. Mork, and I. Tatarinov. Piazza: Data management infrastructure for semantic web applications. *Proceedings WWW*, 2003.
11. A. Halverson, V. Josifovski, G. Lohman, H. Pirahesh, and M. Mörschel. ROX: Relational over XML. *Proceedings VLDB*, 2004.
12. V. Hristidis, Y. Papakonstantinou, and A. Balmin. Keyword proximity search on XML graphs. *Proceedings ICDE*, 2003.
13. Z. G. Ives, A. Y. Halevy, and D. S. Weld. An XML query engine for network-bound data. *The VLDB Journal*, 2:380–402, December 2002.
14. Y. Li, C. Yu, and H. Jagadish. Schema-Free XQuery. *Proceedings VLDB*, 2004.
15. J. Madhavan, P. A. Bernstein, and E. Rahm. Generic Schema Matching with Cupid. *Proceedings VLDB*, pages 49–58, 2001.
16. Y. Papakonstantinou, M. Petropoulos, and V. Vassalos. QURSED: Querying and Reporting Semistructured Data. *Proc. SIGMOD*, 2002.
17. C. Reynaud, J.-P. Sirot, and D. Vodislav. Semantic Integration of XML Heterogeneous Data Sources. *Proceedings IDEAS*, pages 199–208, 2001.
18. J. D. Ullman. Universal Relation Interfaces for Database Systems. *Proceedings IFIP*, 1983.
19. Xyleme. *http://www.xyleme.com*.

SASMINT System for Database Interoperability in Collaborative Networks

Ozgul Unal and Hamideh Afsarmanesh

Informatics Institute, University of Amsterdam
{ozgul, hamideh}@science.uva.nl

Abstract. In most suggested systems aiming to enable interoperability and collaboration among heterogeneous databases, *schema matching* and *integration* is performed manually. The SASMINT system introduced in this paper proposes a (semi-) automated approach to tackle the following: 1) identification of the syntactic/semantic/structural similarities between the donor and recipient schemas to resolve their heterogeneities, 2) suggestion of corresponding mappings among the pairs of matched components, 3) facilitation of user-interaction with the system, necessary for validation/enhancement of results, and 4) generation of a proposed integrated schema, and a set of derivation rules for each of its components to support query processing against integrated sources. Unlike other systems that typically apply one specific algorithm, SASMINT applies a hybrid approach for schema matching that combines a selection of algorithms from NLP and graph theory. Furthermore, SASMINT exploits the user-validated schema matching results in its semi-automatic generation of the integrated schema and its necessary derivations.

1 Introduction

In order to remain competitive in a highly aggressive global market, organizations need to achieve their goals better and faster. At this point, the need for collaboration has become apparent, resulting in a rise in the number of collaborating organizations. A collaborative network (CN) is formed by variety of autonomous, geographically distributed, and heterogeneous organizations that collaborate to better achieve common or compatible goals [1]. There are a number of benefits of CNs, including increased access to market opportunities, sharing risks, reducing costs, achieving business goals not achievable by a single organization, etc.

In order to support rapid formation of collaborative networks, it is required to have a common interoperable infrastructure, operating rules, agreements, etc. [2]. Due to the lack of common standards, each organization uses its own format in database schema definitions, which makes the interoperation among nodes very difficult. Especially if the number of organizations in a CN is large, an important challenge is how to share data represented by heterogeneous schemas. Automatic resolution of schema heterogeneity still remains a main bottleneck for provision of integrated data access/sharing among autonomous, distributed databases.

Integrating data from a given set of databases requires making choices about the structure as well as the semantics of elements in the database schemas. A variety of

R. Meersman, Z. Tari et al. (Eds.): OTM 2006, LNCS 4275, pp. 91 – 108, 2006.

approaches to interoperability have been proposed, aiming at different levels of integration [3]. For example, multidatabases and federated databases [3] have introduced approaches to deal with collaboration among autonomous and heterogeneous databases. With the exception of the approach in pure Federated Databases – where every database node creates its own individual integrated schema (by inter-linking its local schema to the schema imported from others [4]) – all other suggested approaches deem to create one common integrated schema to be shared by all nodes within a collaboration. Therefore, the interoperability infrastructure supporting collaborations, requires effective mechanisms not only to integrate/inter-link database schemas, but also to provide homogeneous access and integrated interface to the heterogeneous and distributed databases. Whenever a new organization joins, its schema (donor) needs to be matched and integrated into the common integrated schema (recipient) of the network, which results in the new extended common integrated schema.

Schema matching and integration is challenging for many reasons. Most importantly, even schemas for identical concepts may have structural and naming differences. They may use similar words to have different meanings. In most previous approaches reported in research literature, there is a great amount of manual work involved in schema matching and integration. Although there is some research focused on schema matching (as later addressed in the related research section), they do not interlink it with the automation of schema integration. Furthermore, schema matching approaches often need manual tuning, such as setting thresholds, providing a thesaurus, etc. There is still a need for a clever and flexible user interface to display match results. Another limitation of the previous approaches is that they typically do not combine different match algorithms in a flexible way. Taking these limitations into account, we propose the SASMINT (Semi-Automatic Schema Matching and INTegration) system and approach [5] introduced in this paper. SASMINT (semi-) automatically matches the schemas and then after user validation/enhancement of matching results, it produces a new extended integrated schema. It combines a variety of linguistic and structural similarity algorithms from Natural Language Processing and Graph Theory research areas, producing more accurate results for schema matching.

The rest of this paper is organized as follows: Section 2 provides an overview of related work. Section 3 motivates the research through several example cases of structural and linguistic (syntactic and semantic) conflicts that need to be considered by the schema matching approach. Section 4 describes the approach of SASMINT, addresses its steps, and introduces the derivation language used for defining the integrated schemas. Finally, Section 5 summarizes the main conclusions of the paper.

2 Related Work

In this section, the main research works related to SASMINT are addressed. First, a number of database interoperability efforts are briefly cited. Then, the main approaches to schema matching, corresponding to the first step of SASMINT, are addressed in more detail, focusing on their limitations compared to SASMINT.

In the literature, there is a great deal of work addressing the challenge of database interoperability. However, these efforts involve a large amount of manual work at different stages of interoperability. For example, in [6] and [7] it is required that schemas are represented in terms of common data model, by the domain expert. Similarly, in several other projects, such as [8-10], local schemas need to be either defined using the terms from a common ontology / schema or mapped to this common ontology manually. These approaches to database interoperability typically ignore the step of schema matching, which could be automated to a great extend.

On the other hand, the schema matching has been usually considered as a separate problem and the related challenges have been addressed by a number of research and development projects. However, these projects still require substantial amounts of manual work and are limited in the solutions that they provide. Furthermore, they mostly do not use linguistic techniques, which are needed in order to increase the overall accuracy of the schema matching system. Another limitation of these projects is that semi-automatic schema matching is not combined with other parts of interoperability, such as schema integration. The main schema matching approaches addressed below.

Cupid [11] involves a normalization process, but it is not as comprehensive as the pre-processing step in SASMINT. Moreover, the name matching in Cupid involves a syntactic matching, which employs only one string similarity metric. Although the COMA system [12] provides a library of matchers that utilize element and structural properties of schemas, it does not support the pre-processing of elements' names. Similarity Flooding [13] identifies the initial maps between the elements of two schemas using only a simple string matcher. These initial maps are then used by a structure matcher. However, Similarity Flooding has no knowledge of edge and node semantics. The ONION [14] system identifies the likely matches between concepts of two ontologies using the linguistic, structure, and inference-based matching. However, it does not employ any combination of string similarity metrics. Furthermore, it is assumed that the relationships among concepts are defined using a set of relationships with pre-defined semantics. GLUE [15] provides a name matcher and several instance-level matchers based on machine- learning techniques. However, ontologies need to be first mapped manually in order to train the learners and a set of domain synonyms and constraints need to be defined before any matching occurs. Hence, a large amount of manual effort is required. Clio [16] generates alternative mappings as SQL view definitions based on the value correspondences defined by the user and thus, no linguistic matching techniques are used as well as much manual work is required.

The results of several other schema matching efforts have been published in [17-19]. The focus of these efforts is on matching large schemas or extensibility of the developed system. However, they share similar problems with the previous efforts mentioned above. They either require much manual work, or if they use linguistic techniques, it is typically a limited usage.

It is clear from the references above that schema matching has been the subject of many efforts. However, none of these efforts considers how to use the result of schema matching for semi-automatic schema integration.

3 Examples of Structural and Linguistic Conflicts

Autonomous organizations define their schemas differently. These different definitions are frequently conflicting and thus their matching and integration is challenging as exemplified below. Different types of conflicts exist among the representation of the same concept in different databases. Since the focus of this paper is only on the structural and linguistic (semantic and syntactic) schema conflicts, the examples that are given in this section belong to this category and are considered as important obstacles to the interoperability among database schemas.

Each of the following examples represents a different kind of schema conflict that belongs to one of three categories: structural, linguistic, and a combination of the two. Structural conflicts are more difficult to resolve than linguistic conflicts.

Clearly, the varieties of types of conflicts that exist among databases are not limited to the ones given here. Furthermore, these examples are from relational databases, but they can be easily extended to the ones from other types. Note that, due to space constraints, we show only partial schemas in a simple format. If relevant to the example, some primary and foreign keys are also shown.

1) Examples of Structural Conflicts: Structural conflicts exist due to the fact that different organizations use different constructs and integrity constraints to represent concepts.

- **Atribute-Entity Conflict:** This conflict arises when a concept is represented as an attribute in one schema and as a separate entity in the second schema. For example, the "section" information is represented by the column 'section' in the schema S1, while the schema S2 represents it as a separate table "Section".

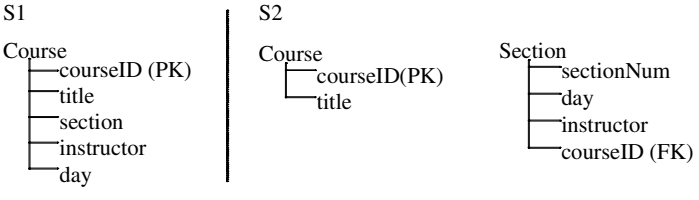


Fig. 1. Attribute-Entity conflict

Another example for Attribute-Entity conflict is shown in Figure 2. In this example, "address" data is spread over three columns in S1, whereas in S2, it is stored in a separate table.

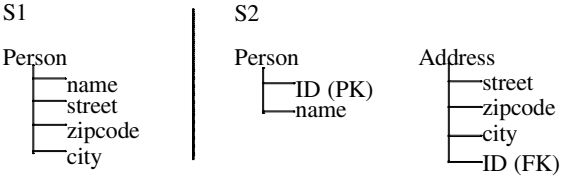


Fig. 2. Attribute-Entity conflict

- **Key Conflict:** This conflict arises when different keys are assigned for the same entity in different schemas. In the example shown in Figure 3, similar to the first case, there is a separate table in S2 for "section" information. Furthermore, although in S1 the "Time" table has a foreign key to the table "Course", in S2 this relationship is through the "Section" table.

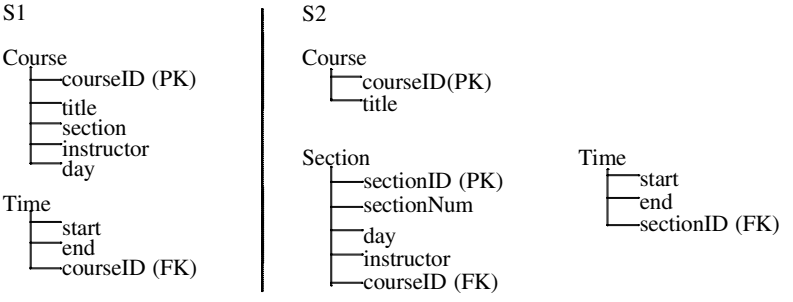


Fig. 3. Key conflict

- **Relationship Conflict:** This conflict is related to the case, in which relationships between entities in different schemas are defined differently, as exemplified in Figure 4: In S1, there is a one-to-many relationship between the "Student" and "Campus", while in S2, the relationship is many-to-many, thus introducing a third table "Apply", to represent this relationship.

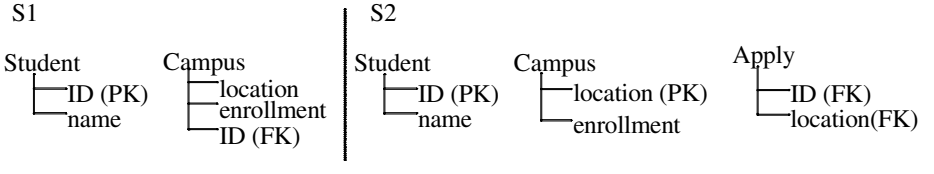


Fig. 4. Relationship conflict

- **Attribute-Attribute Conflict:** This conflict arises when the same concept is represented by using different number of attributes in different schemas. For the example shown in Figure 5, "address" data is stored in one column of the "Person" table in S1, while in S2, it is spread over three columns.

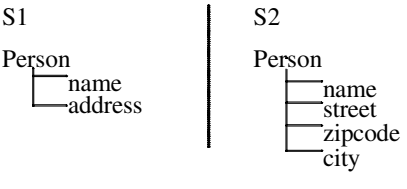


Fig. 5. Attribute-Attribute conflict

2) Examples of Linguistic Conflicts: Linguistic conflicts arise because of different terminology and names that different organizations use to refer to the same data. Linguistic conflicts can be of two types: syntactic and semantic. Below are the examples for syntactic and semantic conflicts. Since this type of schema conflict is related to the differences in representation of "names" of columns or tables, we have not provided example schemas but only some related but different names in the following examples.

- Syntactic Conflicts
  - Names with more than 1 token (word) with different order of tokens, e.g. *year_of_birth* vs. *birth_year*.
  - Abbreviated vs. Extended Names, e.g. *GPA* vs. *GradePointAverage.*
  - Existence of Stop Words, e.g. *departmentName* vs. *name_of_department.*
  - Different forms of a string (plural, singular, etc.), e.g. *shelves* vs. *shelf.*
  - Use of short forms of strings, e.g. *crdts* vs. *credits.*
- Semantic Conflicts
  - Use of synonyms, e.g. *gender* vs. *sex.*
  - Applying some linguistic semantics (like IS-A hierarchy), e.g. *person* vs. *facultyMember.*

3) Example of combined Structural and Linguistic Conflicts: The two types of conflicts addressed above may occur in a combined form in some parts of the schema. Following is an example for this case, where "contact" information in S2 can be matched with the concatenation of "firstName, lastName" in S1, if we apply both the linguistic "IS-A hierarchy" and the structural "one attribute in the first schema matching two attributes in the second" (attribute-attribute conflict).

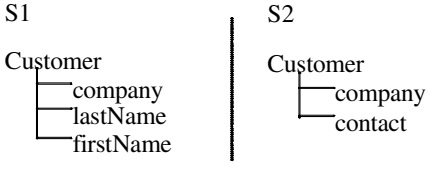


Fig. 6. Combined Structural and Linguistic conflicts case

4 The Proposed Approach of SASMINT

In this section, the approach of the SASMINT system for semi-automatic schema matching and integration is described. As shown in Figure 7, the SASMINT has two main steps: Schema Matching and Schema Integration. Section 4.1 focuses on the sub-steps comprising schema matching, with detailed explanations about the Comparison sub-step. How to use the results of schema matching to generate the integrated schema as well as the derivation language used for defining this schema are the subjects of Section 4.2. As addressed earlier, unlike other approaches, the main contribution of SASMINT is to perform schema matching, as well as schema integration

automatically, to the extent possible, which is achieved by means of the steps explained below.

4.1 Schema Matching

The Schema Matching consists of three main steps, as explained below: 1) Preparation, for bringing schemas into a common representation, 2) Comparison, for identifying the likely matches between elements of two schemas, 3) Result Generation and Validation, for obtaining the user approval /modification/ rejection.

1) Preparation
Since SASMINT aims at supporting schemas represented in different schema languages, such as relational and object-oriented, they need to be brought into a common format before being compared. The Directed Acyclic Graph (DAG) with labeled edges has been chosen for this purpose, considering that it provides a balanced format among other alternatives supporting the representation of a relational schema, an object-oriented schema, etc. as a graph. Furthermore, existing graph theory concepts and algorithms can help compare two graphs. In summary, preparation step deals with transforming database schema information into DAG format using publicly available graph libraries.

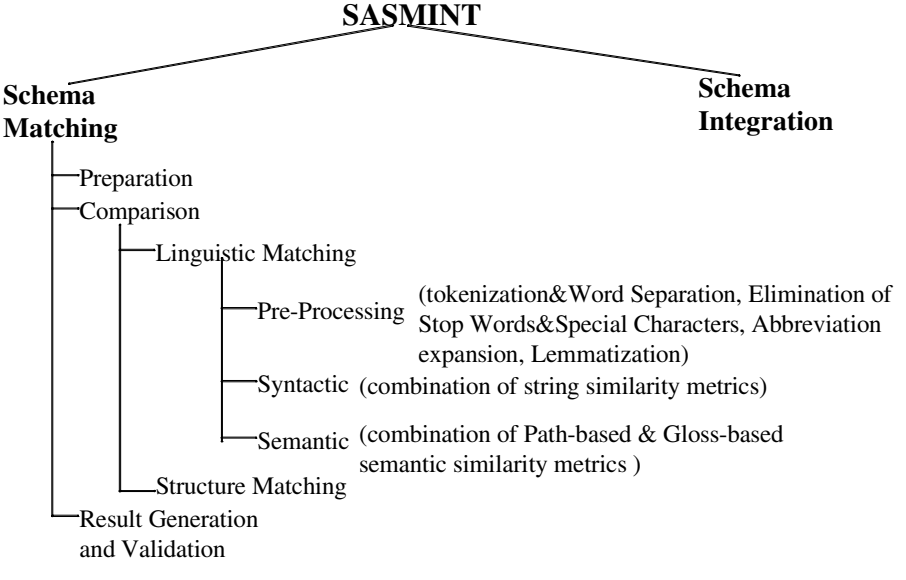


Fig. 7. Processing Steps of SASMINT

2) Comparison
The second step of SASMINT, *Comparison*, is responsible for automatically identifying the likely matches between two schemas, by using a number of algorithms from Natural Language Processing (NLP) and Graph Theory to resolve the syntactic and

semantic as well as structural heterogeneities. The Comparison involves two kinds of matching: Linguistic and Structure, as addressed below. Linguistic matching considers only the names of schema elements. On the other hand, structure matching takes into account the structural aspects of schemas. Results from linguistic and structure matching are combined by weighted summation to determine the similarity of schema elements being compared.

A- Linguistic Matching

This step consists of Syntactical Matching and Semantic Matching, details of which are provided below. The result of the Linguistic Matching is a value between [0..1], for pairs of element names from the two schemas. Before any matching occurs, element names (strings) from the two schemas must be pre-processed to bring them into a common representation, which is called as pre-processing and involves the following operations:

1. ***Tokenization and Word Separation:*** Strings containing multiple words are split into lists of words or tokens.
2. ***Elimination of stop words:*** Stop words are common words such as prepositions, adjectives, and adverbs, e.g. 'of', 'the', etc., that occur frequently but do not have much effect on the meaning of strings. Hence, they are removed from the names.
3. ***Elimination of special characters and De-hyphenation:*** The characters such as '/', '-', etc., are considered as useless and removed from the names.
4. ***Abbreviation expansion:*** Since abbreviations are used in names extensively, they need to be identified and expanded. For this purpose, a dictionary of well-known abbreviations as well as the ones specific to a domain can be used.
5. ***Normalizing terms to a standard form using Lemmatization:*** Multiple forms of the same word need to be brought into a common base form. By means of lemmatization, verb forms are reduced to the infinitive and plural nouns are converted to their singular forms.

After element names are brought into a common representation, a variety of algorithms or metrics from the NLP research field are applied to identify the syntactic and semantic similarities, as addressed below.

I. ***Syntactic Similarity***

Syntactic Similarity uses the algorithms (metrics) from the NLP research field, for comparing two character strings syntactically. As addressed in the Related Work section, previous schema matching approaches typically depend on only one metric. However, each of these metrics is best suited for a different type of strings, meaning that for some types of element names, syntactic similarity in these approaches performs badly. Considering the limitations of utilizing only one metric, SASMINT uses a combination of several main syntactic similarity metrics, making it a more generic tool. These metrics are briefly explained below:

1. *Levenshtein Distance (Edit Distance):* Levenshtein Distance[20], also known as Edit Distance, is based on the idea of minimum number of modifications, in the type of changing, deleting, or inserting a character, required to change a string into another. The costs of modifications are defined as 1 for each operation.

Levenshtein distance is a string-based distance metric, meaning that it does not consider the multi-word string as a combination of tokens but rather as a single string.

2. *Monge-Elkan Distance:* Monge and Elkan [21] proposed another string-based distance function using an affine gap model. Monge-Elkan Distance allows for gaps of unmatched characters. Affine gap costs are specified in two ways: one with a cost for starting a gap and a second for the cost of continuation of the gap.
3. *Jaro:* Jaro [22], a string-based metric well known in the record linkage community, is intended for short strings and considers insertions, deletions, and transpositions. It also takes into account typical spelling deviations.
4. *TF*IDF (Term Frequency*Inverse Document Frequency): TF*IDF* [23] is a vector-based approach from the information retrieval literature that assigns weights to terms. For each of the documents to be compared, first a weighted term vector is composed. Then, the similarity between the documents is computed as the cosine between their weighted term vectors.
5. *Jaccard Similarity:* Jaccard Similarity [24] is a token-based similarity measure, defined for two strings A and B, consisting of one or more words, as the ratio of the number of shared words of A and B to the number owned by A or B.
6. *Longest Common Substring (LCS):* The longest common substring of A and B is the longest run of characters that appear in order inside both A and B.

Syntactic Similarity Metrics Used in SASMINT

The SASMINT system uses all six metrics described above to automatically identify syntactic similarity between element names. Considering that each metric is suitable for a different type of strings and schemas usually consist of mixed sets of element names (strings), SASMINT uses a combination of these metrics to obtain more accurate results. Metrics are combined by means of a weighted summation using the following formula:

$$sim_W(a,b) = w_{lv} * sm_{lv}(a,b) + w_{me} * sm_{me}(a,b) + w_{jr} * sm_{jr}(a,b) + w_{jc} * sm_{jc}(a,b) + w_{tf} * sm_{tf}(a,b) + w_{lc} * sm_{lc}(a,b) \quad (1)$$

where ‘lv’ stands for Levenstein, ‘me’ for Monge-Elkan, ‘jr’ for Jaro, ‘jc’ for Jaccard, ‘tf’ for TF-IDF, and ‘lc’ for Longest Common Substring.

In addition to being suitable for different types of strings, as another contribution, SASMINT proposes a new *recursive weighted* metric to better support the matching of element names containing more than one token. Depending on whether the names contain one or more tokens, the user can choose between the weighted and recursive weighted metric. This new metric is a modified version of Monge-Elkan's recursive metric [21]. Although both are tokenized, our recursive weighted metric is different from Monge-Elkan's metric on one hand that it applies more than one similarity metric to the string pairs, and on the other hand unlike Monge-Elkan’s metric, our metric is symmetric, generating the same result for sim(a,b) and sim(b,a), and thus making our pair matching associative. Given two strings a and b that are tokenized into $a = s_1, s_2, ..s_l$ and $b = t_1, t_2, ...t_m$, the recursive weighted metric is as follows:

$$sim(a,b) = \frac{1}{2l} \sum_{i=1}^{l} \max_{j=1}^{m} sim_W(a_i, b_j) + \frac{1}{2m} \sum_{j=1}^{m} \max_{i=1}^{l} sim_W(a_i, b_j) \quad (2)$$

Tests for Determining the Accuracy of Weighted Metrics

We carried out a number of tests to validate the accuracy of our syntactic similarity approach by using the test data of Similarity Flooding [13] and Clio [16], which are from different domains of finance, library, university, etc. Due to space constraints, we show only two of our test results in Figures 8 and 9. Syntactic similarity between each possible combination of element name pairs from two schemas were calculated in three different ways: *1)* Using the six metrics individually, *2)* using the SASMINT's weighted sum of the individual metrics, and *3)* using the SASMINT's modified recursive weighted metric. If the similarity is above the threshold then this match is accepted. In case of more than one match for the same element, the higher valued one is selected. Figures 8 and 9 show the evaluation results where six leftmost sets of 3 columns represent the results for individual metrics, while the remaining two sets are weighted and recursive weighted metrics respectively.

In the evaluation process, we used the concepts of precision and recall from the information retrieval field [25]. Precision (P) and Recall (R) are computed as follows:

$$P = \frac{x}{x+z} \qquad R = \frac{x}{x+y} \quad (3)$$

where x is the number of correctly identified similar strings, z is the number of strings found as similar, which are actually not similar (called as false positives), and y is the number of similar strings, which could not be identified by the system (called as false negatives).

Since neither of precision and recall measures can accurately assess the match quality alone, another measure, called as F-measure, is proposed in the literature [26], combining recall and precision as follows:

$$F = \frac{2}{\frac{1}{P} + \frac{1}{R}} \quad (4)$$

Although in Figures 8 and 9 the values of precision and recall are also shown, for the purpose of the assessment of the similarity metrics, values of F-measure are considered. Based on the results of tests, it is important to note that in general while one metric among the six may perform well on one data set, it may not on another. This is due to the fact that different test sets, from different domains, contain elements with different characteristics, while at the same time each of the six metrics is usually suitable for specific type(s) of strings. However, SASMINT's approach is more generic and in all our similarity evaluation tests we observed that the weighted sum of metrics consistently performed almost as good as the best metric among the six. Thus, we concluded that using our weighted metric generally generates better results on ad-hoc schemas.

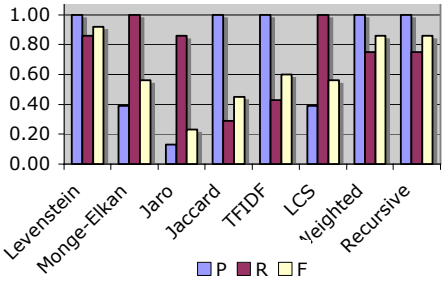


Fig. 8. Test Result 1

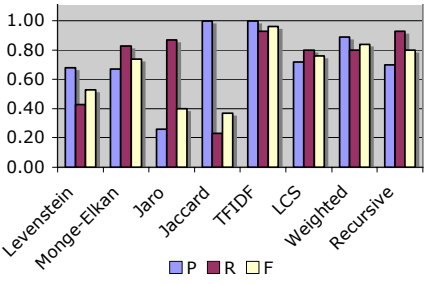


Fig. 9. Test Result 2

II. ***Semantic Similarity***

SASMINT uses two semantic similarity algorithms from the NLP domain: 1) path based measures and 2) gloss-based measures, as briefly explained below.

1. *Path-based Measures:* SASMINT uses the measure of Wu and Palmer [27]. It is based on the idea of calculating the shortest path between the concepts in the IS-A hierarchy of WordNet, a lexical dictionary [28]. Wu and Palmer calculates the semantic similarity of two concepts using the following formula:

$$sim_{wp}(c_1, c_2) = \frac{2 * depth(lcs(c_1, c_2))}{depth(c_1) + depth(c_2)} \quad (5)$$

   where *depth* is the distance from the root node and $lcs(c_1, c_2)$ is the maximally specific superclass of c_1 and c_2, which is also known as the lowest common subsumer.
2. *Gloss-based Measures:* Another type of measure used in SASMINT to determine semantic similarity is based on gloss overlaps. *G*loss refers to a brief description of a word. The measure of Lesk [29], which forms the base for the gloss-based measure used in SASMINT, counts the number of common words between each sense of the target word and the sense of other words in a sentence. A word can have different senses, depending on the context. We convert the algorithm of Lesk to compute the semantic similarity of two concepts c_1 and c_2 as follows: for each of the senses of c_1, we compute the number of common words between its glosses and the glosses of each of the senses of c_2.

Semantic Similarity Metrics Used in SASMINT

Similar to the case in the syntactic similarity component of the system, semantic similarity also calculates the weighted sum of the two semantic similarity measures addressed above. The default value of the weights is 0.5. Alternatively, users can run the sampler functionality of the system, introduced below, to determine the weights. SASMINT uses WordNet to identify the path between the concepts being compared. Similarly, it benefits from the gloss information provided in WordNet for calculating the Gloss-based similarity.

Sampler Component for Identifying the Weights Automatically

Another key innovation of SASMINT is its *Sampler Component* for automatically identifying the weights of metrics used in the syntactic and semantic similarity steps. Since it may not be easy for the user to identify the weights for each of the metrics used, the sampler component helps the user with this task. The sampler component works with up to ten known sample pairs, that are syntactically or semantically similar, depending on for which type of metrics the user wants to run the Sampler. These pairs are provided by the user from his schema domain, for example, "student_name" and "name_of_student" for a syntactically similar pair and “employee” and “worker” for a semantically similar pair. Then, the sampler runs the metrics for syntactic or semantic similarity individually over these pairs, and determines their calculated similarities (between 0 and 1) for each pair. After that, it measures the accuracy level of each metric in terms of F-measure. Using the following formula, the Sampler calculates the weight for each metric; where $\sum F$ represents the sum of F-measure values resulted for all metrics used, and F_m represents the F-measure value calculated for metric ‘m’.

$$w_m = \frac{1}{\sum F} * F_m \tag{6}$$

Finally, after all “weights” for metrics are determined by the sampler component, they are presented to the user, who can accept or modify them. If the user does not run the sampler, then he/she either will define them, or averaging the metrics (equal weights for all) is the only mechanism that SASMINT can use, although it may not produce desirable results.

B- Structure Matching

The second step of schema matching in SASMINT is structure matching, which uses the result of linguistic matching to identify the structural similarity of two schemas represented as graphs. It is based on the idea that if two elements have been found to be similar, their adjacent elements (parent and children nodes) may also match. Moreover, the similarity of two nodes is affected by the number and quality of the similarity of their children.

For the purpose of structure matching, a variety of graph similarity and matching algorithms from Graph Theory and other areas like Web searching and schema matching were considered. Among different alternatives, three of the approaches were found to be most relevant including: graph similarity algorithm proposed in [30], structure matching of Similarity Flooding [13], and that of Cupid [11].

The authors of [30] compute the similarity of two graphs G_A and G_B with the vertices n_A and n_B and edges E_A and E_B. For $i = 1,..,n_B$ and $j = 1,..,n_A$ the scores are updated iteratively using the following equation:

$$X_{k+1} = BX_kA^T + B^TX_kA \tag{7}$$

where X_k is the n_B x n_A matrix of entries x_{ij} at iteration k, A and B are the adjacency matrices of G_A and G_B, and A^T and B^T are the transpose of A and B.

Then, based on this equation, the authors define the following:

$$Z_{k+1} = \frac{BZ_kA^T + B^TZ_kA}{\left\|BZ_kA^T + B^TZ_kA\right\|_F} \quad \text{k=0,1,....} \tag{8}$$

The matrix norm $\|.\|_F$ used here is known as the Euclidean or Frobenius norm and equals to the square root of the sum of all squared entries. The matrix subsequences Z_{2k} and Z_{2k+1} converge to Z_{even} and Z_{odd}.

Structure matching of Similarity Flooding [13] is based on a fix point computation. It is based on the assumption that whenever any two elements are found to be similar, similarity of their adjacent elements increases. Over a number of iterations, the initial similarity of any two nodes propagates through the graphs. The algorithm terminates after the similarities of all model elements stabilize.

Structure matching in Cupid on the other hand, exploits a tree match algorithm which is based on the following perceptions [11]: Atomic elements in the two trees are similar if they are individually similar and if their ancestors and siblings are similar. Two non-leaf elements are similar if they are linguistically similar and the subtrees rooted at the two elements are similar. Two non-leaf schema elements are structurally similar if their leaf sets are highly similar, even if their immediate children are not.

Structure Similarity Algorithms used in SASMINT

All the above algorithms form the base for the structure matching component of SASMINT. Similar to the method followed in linguistic matching, structure matching uses the weighted sum of the three structural similarity algorithms introduced above.

Result of Comparison Step

The final similarity of two schema elements being compared is the weighted summation of their linguistic and structure similarity values. In other words, similarity for a pair of schema elements (n_i, n_j) is calculated as follows:

$$S_{i,j} = ls_{i,j} * w_{ls} + ss_{i,j} * w_{ss} \tag{9}$$

where $ls_{i,j}$ and $ss_{i,j}$ represent the similarity of n_i and n_j in [0..1], identified at the linguistic and structure matching steps respectively, and w_{ls} is the weight of the linguistic matching, and w_{ss} is the weight of the structure matching.

3) Final Result Generation and Validation

After the Comparison step identifies the correspondences between schema elements, the resulting matches need to be displayed to the user by means of a friendly GUI. This step of schema matching has not gained sufficient interest in other approaches to schema matching. However, a clever and flexible GUI is an indispensable part of a matching system, because it is not possible to determine all possible matches automatically and not all the identified matches may be correct, especially considering the existence of large amount of semantics involved in schema descriptions.

4.2 Schema Integration

For every two schemas, after saving the results of their validated schema matching, the user has the option to generate an integrated schema. Schema integration is a difficult process because of the structural and linguistic conflicts among schemas, for which a number of examples are given in Section 3. SASMINT aims to facilitate this task by exploiting the validated results of semi-automatic schema matching that minimizes the required user input.

Considering different conflicts to be resolved, currently, a number of rules for integrating relational schemas have been defined for SASMINT, which can also be applied to object-oriented databases. Since these rules are still under development, they are not explained in this paper, but will be addressed in future publications. It is important to note that they are meant for integrating the schemas automatically. Nevertheless, there are some cases, like 1-to-n match (one column in the first schema matches n columns in the second schema), for which an automatic integration is not possible. For these cases, the user needs to specify the special mapping at the Final Result Generation and Validation Step. For example, suppose that Schema Matching has found a match between "address" in one schema and "addr" and "dress" in the second schema. In this case, user deletes the match between "address" and "dress" as it is meaningless. However, if there is a match between "address" in one schema and "street", "zip", and "city" in the second schema, the user specifies that address is the concatenation of "street", "zip", and "city".

Using the integration rules and user validated mappings, Schema Integration component of SASMINT proposes an integrated schema. After the user checks, modifies, and requests to save the integrated schema, the resulting schema is stored using a derivation language [31]. There are two types of derivation for relational schemas: Table and Column Derivation, as explained below.

1) Table Derivation: A Derived Table is defined by the following expression:

derived –table-definition :=derived-table-name = <T-expr>

T-list:= <T-expr> | <T-expr> , <T-list>

T-expr:= table-name@schema-name | union (<T-expr> , <T-list>) |
subtract (<T-expr> , <T-expr>) | restrict (<T-expr> , <restriction>)

Table derivation primates, used in the expression above are defined below, where every T_i stands for *table-name@schema-name* and *T* represents *derived table*:

1. *Table Rename*
   T = T_1
   Example: Section = Sect@S1
2. *Table Union*
   T = $union(T_1,..,T_n)$
   Example: Course = union(Course@S1,Course@S2)
3. *Table Subtract*
   T = $subtract(T_1,T_2)$
   Example: MathStudents = subtract(Students@S1, PhysicsStudents@S1)

4. *Table Restrict*
   T = $restrict(T_1, restriction)$
   Example: StudentsPassed = restrict(Students@S1,[grade > 60])

2) Column Derivation: A Derived Column is defined by the following expression:
derived–column-definition :=derived-column-name = <c-expr>
c-list:= <c-expr> | <c-expr> , <c-list>
c-expr:= column-name@table-name@schema-name | {<c-list>} |
<c-expr> OPR <c-expr>

Following is the list of derivation primitives for column integration, where every c_i stands for *column-name@table-name@schema-name* and *c* stands for *derived column@table-name*, union primitive is represented by "*{,}*", and extraction primitive is represented by "*OPR*":

1. *Column Rename*
   c = c_1
   Example: grade@Student = GPA@Student@S1
2. *Column Union*
   c = $\{c_1,..,c_n\}$
   Example: day@Section = {day@Course@S1,day@Section@S1}
3. *Column Extraction*
   c = $c_1 OPR c_2 OPR,..c_m$ where *OPR* can be any type of arithmetic operation if c_i's are of type numeric and string operation if they are of type string. Right and left hand side of the operation must be the same type.
   Example: address@Student = street@Student@S1 + zip@Student@S1 + city@Student@S1
   Here "+" represents *string concatenation*.

Among the derivation primitives defined above, only *table rename & union* and *column rename & union* can be automatically determined by the SASMINT system. For the remaining, user input is required at the end of schema matching, as they are related to semantics of tables and columns as well as data instances.

The schema integration component enables iterative development of a common integrated schema for collaborative network of nodes, two schemas at a time, as follows: First, schemas S_1 and S_2 of two nodes are selected by the user. After schema matching and necessary user modifications and validation, they are integrated into S_{int1} and the result is saved. Then, the user selects S_{int1} and the schema of another node S_3 integrating them into S_{int2}. This process continues until the schemas of all nodes are integrated, resulting in a final integrated schema S_{int}. In Figures 10 and 11 a simple example for schema integration is shown, where first the matching results are presented by the system (in Figure 10), and then, after user validation, this result is used to produce a proposed integrated schema and its definition in derivation language (in Figure 11).

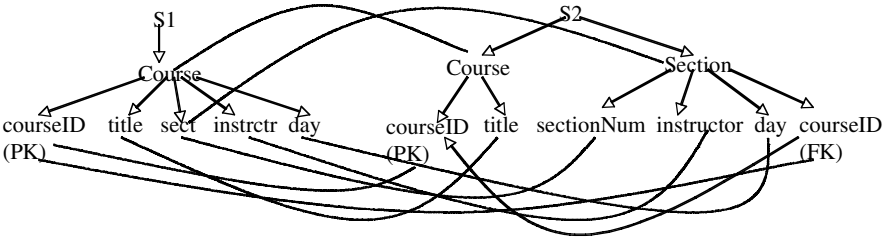
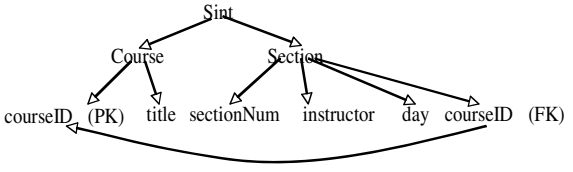


Fig. 10. Result of Schema Matching



Course = union(Course@S1, Course@S2)
courseID (PK)@Course = {courseID(PK)@Course@S1,courseID(PK)@Course@S2}
title@Course = {title@@Course@S1,title@Course@S2}
Section = Section@S2
sectionNum@Section = {sect@Course@S1,sectionNum@Section@S2}
instructor@Section = {instrctr@Course@S1,instructor@Section@S2}
day@Section = {day@Course@S1,day@Section@S2}
courseID (FK)@Section = courseID (FK)@Section@S2

Fig. 11. Proposed Integrated Schema

5 Conclusion

This paper introduces the SASMINT system for resolution of syntactic, semantic, and structural heterogeneity among database schemas in Collaborative Networks. SASMINT first enables semi-automatic schema matching by identifying and resolving schema heterogeneity, and after users' validation of its matching results, generates a new extended integrated schema. The approach introduced and exploited in this system is generic in that it aims to identify the correspondences between different schemas, as automatically as possible. Since SASMINT uses a combination of several schema matching techniques and a sampler tool for users to influence the weights for applying these techniques, it is better equipped with handling the variety of types of ad-hoc strings in different schemas. It simultaneously uses a number of NLP algorithms that together with the structure matching enables achievement of a more generic schema matching. Furthermore, SASMINT requires minimum user involvement for generation of the integrated schema. The automatic use of the results produced by schema matching for generation of a new extended integrated schema as well as a derivation language for defining this schema are parts of the contribution of SASMINT presented in this paper.

References

1. Camarinha-Matos, L.M., H. Afsarmanesh, and M. Ollus. ECOLEAD: A Holistic Approach to Creation and Management of Dynamic Virtual Organizations. In Proc. of PRO-VE'05, p. 3-16, 2005.
2. Afsarmanesh, H. and L.M. Camarinha-Matos. A Framework for Management of Virtual Organizations Breeding Environments. In Proc. of PRO-VE'05, p. 35-48, 2005.
3. Sheth, A. and J. Larson, Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases. ACM Computing Surveys, 22(3): p. 183-236, 1990.
4. Tuijnman, F. and H. Afsarmanesh, Management of Shared Data in Federated Cooperative PEER Environment. *International Journal of Intelligent & Cooperative Information Systems*, 2(4): p. 451-473, 1993.
5. Unal, O. and H. Afsarmanesh. Interoperability in Collaborative Network of Biodiversity Organizations. In Proc. of PRO-VE'06, Accepted for Publication, 2006.
6. Hammer, J. and D. McLeod, An Approach to Resolving Semantic Heterogeneity in a Federation of Autonomous, Heterogeneous Database Systems. *International Journal of Intelligent & Cooperative Information Systems*, World Scientific, 2(1): p. 51-83, 1993.
7. Bergamaschi, S., et al. A Semantic Approach to Information Integration: the MOMIS project. In Proc. of Sesto Convegno della Associazione Italiana per l'Intelligenza Artificiale, AI*IA 98, 1998.
8. Bayardo, R.J., et al. InfoSleuth: Agent-Based Semantic Integration of Information in Open and Dynamic Environments. In Proc. of ACM SIGMOD International Conference on Management of Data, 1997.
9. Arens, Y., C.A. Knoblock, and W.-M. Shen, Query Reformulation for Dynamic Information Integration. *Journal of Intelligent Information Systems*: p. 99-130, 1996.
10. Mena, E., et al., OBSERVER: An Approach for Query Processing in Global Information Systems based on Interoperation across Pre-existing Ontologies. *Distributed and Parallel Databases Journal*, 8(2): p. 223-271, 2000.
11. Madhavan, J., P.A. Bernstein, and E. Rahm. Generic Schema Matching with Cupid. In Proc. of VLDB, p. 49-58, 2001.
12. Do, H.H. and E. Rahm. COMA - A System for Flexible Combination of Schema Matching Approaches. In Proc. of VLDB, p. 610-621, 2002.
13. Melnik, S., H. Garcia-Molina, and E. Rahm. Similarity Flooding: A Versatile Graph Matching Algorithm and its Application to Schema Matching. In Proc. of ICDE, 2002.
14. Mitra, P., G. Wiederhold, and S. Decker. A scalable framework for the interoperation of information sources. In Proc. of. International Semantic Web Working Symposium, 2001.
15. Doan, A., et al. Learning to Map between Ontologies on the Semantic Web. In Proc. of World-Wide Web Conf. (WWW-2002), 2002.
16. Miller, R.J., L.M. Haas, and M.A. Hernandez. Schema Mapping as Query Discovery. In Proc. of VLDB, p. 77-88, 2000.
17. Embley, D.W., L. Xu, and Y. Ding, Automatic direct and indirect schema mapping: Experiences and lessons learned. *ACM SIGMOD Record*, 33(4): p. 14-19, December 2004.
18. Bernstein, P.A., et al., Industrial-Strength Schema Matching. *ACM SIGMOD Record*, 33(4): p. 38-43, December 2004.
19. Rahm, E., H.-H. Do, and S. Maßmann, Matching Large XML Schemas. *ACM SIGMOD Record*, 33(4): p. 26-31, December 2004.
20. Levenshtein, V.I., Binary codes capable of correcting deletions, insertions, and reversals. *Cybernetics and Control Theory*, 10(8): p. 707-710, 1966.

21. Monge, A.E. and C. Elkan. The Field Matching Problem: Algorithms and Applications. In Second International Conference on Knowledge Discovery and Data Mining, 267-270. 1996.
22. Jaro, M.A., Probabilistic linkage of large public health. *Statistics in Medicine*: p. 14:491-498, 1995.
23. Salton, G. and C.S. Yang, On the specification of term values in automatic indexing. *Journal of Documentation*, (29): p. 351-372, 1973.
24. Jaccard, P., The distribution of flora in the alpine zone. *The New Phytologist*, 11(2): p. 37-50, 1912.
25. Cleverdon, C.W. and E.M. Keen, Factors determining the performance of indexing systems, volume 2: test results, Aslib Cranfield Research Project. Cranfield Institute of Technology, 1966.
26. Rijsbergen, C.J.v., Information Retrieval: Butterworths, London, 1979.
27. Wu, Z. and M. Palmer. Verb Semantics and Lexical Selection. In 32nd Annual Meeting of the Association for Computational Linguistics, 1994.
28. Fellbaum, C., An Electronic Lexical Database.: Cambridge: MIT press, 1998.
29. Lesk, M. Automatic Sense Disambiguation Using Machine Readable Dictionaries: How to Tell a Pine Code from an Ice Cream Cone. In Proc. of 5th International Conference on Systems Documentation, 24-26. 1986.
30. Blondel, V., et al., A Measure of Similarity between Graph Vertices: Applications to Synonym Extraction and Web Searching. *Journal of SIAM Review*, 46(4): p. 647-666, 2004.
31. Afsarmanesh, H., et al., The PEER Information Management Language User Manual. Technical Report *CS-94-14*, Department of Computer Systems, University of Amsterdam, 1994.

Querying E-Catalogs Using Content Summaries

Aixin Sun[1,⋆], Boualem Benatallah[1], Mohand-Saïd Hacid[2], and Mahbub Hassan[1]

[1] School of Computer Science and Engineering, University of New South Wales, Sydney, NSW 2052, Australia
{boualem, mahbub}@cse.unsw.edu.au
[2] LIRIS - UFR d'Informatique, Universite Claude Bernard Lyon 1, 69622 Villeurbanne cedex, France
mshacid@bat710.univ-lyon1.fr

Abstract. With the rapid development of e-services on the Web, increasing number of e-catalogs are becoming accessible to users. A large number of e-catalogs provide information about similar type of products/services. To simplify users information searching effort, data integration systems have being developed to integrate e-catalogs providing similar type of information such that users can query those e-catalogs with a mediator through an uniform query interface. The conventional approach to answer a query received by a mediator is to select e-catalogs purely based on their query capabilities, i.e., query interface specifications. However, an e-catalog having the capability to answer a query does not mean it has relevant answers to the query. To remedy the wasted resources of querying catalogs that do not generate an answer, in this paper, we propose to use catalog content summary as a filter and select the relevant e-catalogs to answer a given query based not only on their query capabilities but also on their content relevance to the query. A multi-attribute content (MAC) summary is proposed to describe an e-catalog with respect to its content. With MAC summary, an e-catalog is selected to answer a query only if the e-catalog is likely having answers to the query. MAC summary can be constructed and updated using answers returned from e-catalogs and therefore the e-catalogs need not be cooperative. We evaluated MAC summary on 50 e-catalogs, and the experimental results were promising.

1 Introduction

Nowadays, a large number of suppliers are offering access to their *products* or *information portals* (also known as *e-catalogs*) using structured query interfaces (e.g., via Web forms and Web Services). According to a recent survey on Web Services [9], 45% of the public Web Services are wrappers of data sources. For instance, Amazon.com is offering a Web Service that provides operations to access detailed product information. The two important features of the Amazon

⋆ Aixin Sun is currently with the School of Computer Engineering, Nanyang Technological University, Singapore. Email: axsun@ntu.edu.sg

R. Meersman, Z. Tari et al. (Eds.): OTM 2006, LNCS 4275, pp. 109–126, 2006.

E-Commerce Service[1] are "detailed product information access" and "extended search". The former allows access to detailed product attribute descriptions (e.g., color, luster, size, and clarity of a pearl in Amazon.com's Jewerly store). The latter allows attribute-based product search. For instance, books in Amazon.com can be searched by any combination of attributes including *author*, *title*, *subject*, *publisher*, and *ISBN*. Attribute-based product search is clearly much more powerful for users to locate products than commonly used keyword-based search [1,4].

While structured data access makes it much easier to express more precise queries and increases the effectiveness of querying Web accessible data sources, the large number of e-catalogs specialized in various domains (e.g., online-shopping, travel portals), makes information access a time-consuming and complex process for end-users. To save users time and effort, data integration systems are being developed to integrate e-catalogs providing similar type of products/services together to form e-catalog communities [2,3,14,15]. An e-catalog community (or catalog community for short) is a mediator of multiple e-catalogs where all those e-catalogs can be queried over the community schema.

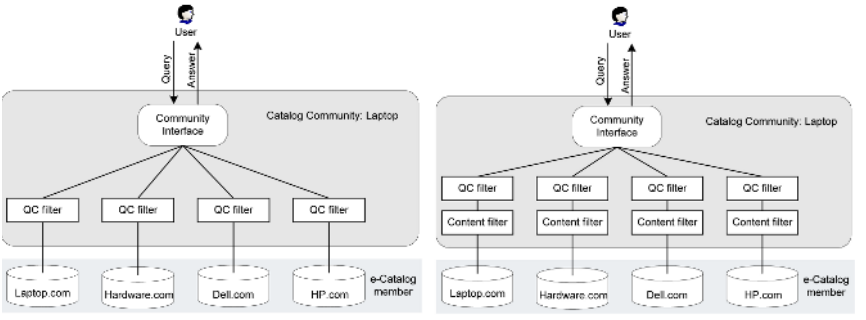
To query e-catalogs in a catalog community, users formulate queries over community schema and submit them to the catalog community. Given a query, the catalog community selects the set of e-catalogs having the query capability (QC) to answer the query. Query capability of an e-catalog refers to its query interface specifications, e.g., input/output parameters and constraints on those parameters. Figure 1(a) shows a catalog community example integrating four e-catalogs selling laptops. The e-catalog selection process in catalog community is illustrated as a process of a query passing through QC filter of each e-catalog[2], where the QC filter contains the query capability of the e-catalog. Intuitively, if an e-catalog has the query capability to answer a query, the query is able to pass through its QC filter; hence the e-catalog is selected to answer the query. If an e-catalog is selected, the query is first translated according to the semantic mapping [16] between the community schema and the e-catalog's local schema and then forwarded to the e-catalog for execution. Answers received from the selected e-catalogs are presented to the user after post-processing[3].

Having query capability to answer a query does not mean an e-catalog has answers to the query. If an e-catalog is selected to answer queries not relevant to its content, its workload is unnecessarily increased; moreover, the effort of translating queries from global schema to its local schema contributes to higher query processing cost inside a catalog community. This calls for an approach that could effectively identify the e-catalogs that are most likely to return answers to a query. The idea is to select e-catalogs not only based on their query capabilities but also based on how likely their contents are relevant to answer a given query. In this paper, we address the problem of selecting e-catalogs by resorting

[1] URL: http://www.amazon.com/gp/aws/landing.html.

[2] Here, for easy illustration purpose, only selection query is considered without join operations.

[3] Example post-processing operations include duplicates removal, ranking of results and so on. Detailed description of post-processing is outside the scope of this paper.



(a) Query capability

(b) Query capability and content summary

Fig. 1. E-catalog selection in catalog community

to content filter(s) as shown in Figure 1(b). Each content filter contains a content summary of an e-catalog that can be used to predict whether the e-catalog has answers to a given query. In other words, by accommodating content filter, an e-catalog is selected to answer a query if the query can pass through both its QC filter and content filter. For instance, the query Q: `find laptops with brand='Acer' and weight<1.0kg` may not pass through the content filter of "Hardware.com" as it has no laptops less than 1.0kg; hence "Hardware.com" is not selected to answer Q. Clearly, by using content filter, the number of e-catalogs selected to answer a query is reduced. By doing so, first, workload of individual e-catalogs is reduced as e-catalogs need not process queries irrelevant to them; second, the cost of query processing inside a catalog community is reduced as less number of e-catalogs are selected to answer queries; third, network communication cost is reduced. Network communication cost refers to the cost of setting up connections between the mediator and e-catalogs, and the cost of transferring data between them. Reducing network communication cost is crucial for applications where the bandwidth is limited such as on-board communication [19].

Building e-catalog content summary is challenging due to the following reasons:

- *E-catalogs are autonomous.* Not all e-catalogs would be cooperative to export catalog content summaries required by the catalog community. It is also very unlikely that catalog community can directly access an e-catalog's database. Hence, building catalog summaries should rely on as less support as possible from e-catalogs.
- *E-catalogs are dynamic.* Number of e-catalogs in a catalog community may change from time to time. Content summary of a newly joined member needs to be built without affecting other summaries; similarly, once a member leaves a community, deletion of its summary should not affect other

summaries. Moreover, it is also challenging to keep content summaries accurate as members update their own content from time to time.

– *False positive and false negative selections lead to uneven cost for end users.* If an e-catalog is false positively selected to answer a query, the e-catalog returns no answer to the query and a user is on no risk to lose any answer to the query. However, a user may unnecessarily pay more for a product if the e-catalog offering the best price is false negatively selected. Minimizing false negative selection rate is therefore one of the challenges in designing and using content summaries.

In our earlier work, we have proposed an infrastructure for integrating and querying e-catalogs, known as E-CatalogNet [2,3]. In E-CatalogNet, e-catalogs offering similar types of products/services are integrated to form catalog communities. In each catalog community, schemas of e-catalogs are mapped to the community schema following a *LAV* (Local As View) approach [11,22]. A query rewriting algorithm has also been proposed in E-CatalogNet to select e-catalogs and generate query plans based on their query capabilities to answer user queries [3]. In this paper, we further extend the E-CatalogNet to build and use content summaries in e-catalog selection. This work makes two major contributions:

1. We introduce a *Multi-Attribute Content (MAC) summary structure*. A MAC summary of an e-catalog is a set of conjunctive conditions. A boolean value is associated with each conjunctive condition which indicates whether the e-catalog has answers to queries satisfying the condition.
2. We provide techniques to build MAC summary, to use MAC summary to select e-catalog for a given query, and to update MAC summary. MAC summary can be built and updated by catalog community using query answers returned from e-catalogs, hence both the building and update processes do not rely on cooperation from e-catalogs.

We present our experiments with 50 e-catalogs simulated using 45,221 tuples in one catalog community. The accuracies of e-catalog selection using MAC summary were evaluated using 5000 randomly generated queries. With MAC summary, on average, less than 70% of e-catalogs were selected to answer a query. The effectiveness of summary update was also experimented with simulated content updates in e-catalogs. The results confirmed that MAC summary was resilient to change and could be updated effectively to reflect content changes inside e-catalogs. Experimental results also showed that MAC summary was able to achieve low false negative selection rate.

The rest of the paper is organized as follows. In Section 2, we discuss the different approaches in building and using content summaries. The proposed MAC summary structure is presented in Section 3, followed by selecting e-catalogs using MAC summaries in Section 4. How to build and update MAC summary is addressed in Section 5. Our experiments and results are presented in Section 6 and then we conclude in Section 7.

2 Related Work

The idea of using content summary for data source selection has been mainly studied in distributed information retrieval where the data sources are collections of text documents [5,8,20]. A content summary of a textual data source is a set of words each associated with a weight derived from a set of sample documents retrieved from the data source. In data integration, there are less studies on using content summaries for data source selection.

Content summary of an e-catalog can be either built by the e-catalog itself or built by catalog community. If all e-catalogs are cooperative, each e-catalog can build its own summary using techniques such as database generalization [13,21]. The locally built summaries can then be uploaded to catalog community for catalog selection. However, e-catalogs are owned by different providers and it is unlikely that all e-catalogs would be cooperative. We therefore focus mainly on those approaches that build summaries at catalog community without e-catalog's cooperation. Roughly speaking, the existing approaches can be divided into three categories.

Constraint-based summary
In Information Manifold [14], a content summary of a data source is one or more conjunctive conditions that all tuples in the data source must satisfy. For instance, all laptops in e-catalog "Dell.com" satisfy `brand='Dell' and weight>= 1.0kg`. A similar approach is also used in a query routing system presented in [15] and constraint-based data sources integration [7]. As the constraints need to be satisfied by all tuples in an e-catalog throughout their life span regardless of data trend, the approach is not aimed at capturing data distribution inside e-catalog. This approach therefore often leads to high false positive selection rate. The summary, however, is resilient to change as it is less affected by content update inside an e-catalog.

Sample-based summary
Yu *et. al.* [23] proposed the FQ (Frequent Queries) method to address the problem of distributed top-N query processing by caching frequent queries and their answers from a number of data sources sharing the same global schema. The frequent queries are maintained by the mediator. The content summary of a data source is the collection of top ranked answers from the source, one for each frequent query. As the best matching answer for each frequent query from each source needs to be stored, the summary could be relatively large in size and leads to high computational and storage cost. The summary, however, can be updated when processing user queries by replacing the old cached answers with updated answers.

Rule-based global summary
In StatMiner [18], data sources are selected based on coverage and overlap statistics learned from the answers of a set of sample queries where all data sources share the same global schema. Coverage of a source with respect to a query is the probability that a random answer tuple belongs to it. Overlap among a set of sources with respect to a query is the probability that a random answer tuple for the query

is presented in each source of the set. There is no individual content summary for each data source; rather a global summary is used for all the sources. The global summary is a set of association rules and each rule contains a query class definition and a set of sources. A query class is defined as a conjunctive condition on one or more attributes from global schema. The set of sources associated with a query class are the subset of sources covering all distinct answers for any query belonging to the query class. As global summary could capture the overlapping information among sources, the list of sources selected for a given query could be the smallest subset of sources that return all distinct answers to the query (not necessarily all the relevant sources). On the other hand, any change of the number of integrated sources or the content change inside one or more data sources could make the summary invalid. The entire summary may therefore need to be rebuilt to reflect the changes. If the changes are frequent, the maintenance cost of global summary could be very high.

The three approaches discussed above can be grouped under two general categories, namely, *local summarization* and *global summarization*. With the former, an individual content summary is built for each data source. The summaries are therefore more resilient to changes as summaries for newly joined (or left) e-catalog can be built (or deleted) without affecting the others. A summary can also be updated to reflect its corresponding e-catalog's content change. Nevertheless, given a query, each summary needs to be tested against the query to find out whether the data source has answers to the query. All the relevant sources must be queried and the duplicate answers are then removed in post-processing. With global summarization, a global summary is built for all data sources and any given query is tested against the only summary to get the list of relevant sources covering all distinct answers to the query. However, any change, either in terms of the number of e-catalogs in the community or in terms of content of any particular e-catalog, could make the summary invalid. In a nutshell, global summarization is more suitable for problems where duplicates commonly exist in member e-catalogs and e-catalogs update their contents less frequently. Local summarization, on the other hand, is more resilient to changes and suitable for problems where e-catalogs update their contents more frequently. In this paper, we chose to use local summarization approach and build a summary for each e-catalog as e-catalogs are dynamic.

3 Multi-Attribute Content Summary

Given a catalog community $\mathcal{C}$, let $R_c(A_1, A_2, \ldots, A_n)$ be the *community relation* defined by the community administrator, where A_i $(1 \leq i \leq n)$ is an attribute. $R_c(A_1, A_2, \ldots, A_n)$ can be seen as the global schema. Each e-catalog offers a view over R_c with one or more attributes from $A_1, A_2, \ldots, A_n$. In this paper, we use an example community relation $\texttt{laptop}(m, b, w, p, r)$ to illustrate our approach where m, b, w, p and r stand for *model_number*, *brand*, *weight*, *price*, and *review* respectively. We also use three e-catalogs with different query capabilities over R_c (see Table 1) as examples to illustrate the construction and usage of

Table 1. Example e-catalogs and their query capabilities

E-catalog	Attributes supported
Hardware.com(c_H)	m, b, w, p
Dell.com(c_D)	m, w, p
Review.com(c_R)	m, b, r

ID	Query bucket for c_H	$m(B_k, c_H)$
B_{H1}	b=`'Acer'` $\wedge\ 0 \leq w < 1.5$	0
B_{H2}	b=`'Dell'` $\wedge\ 0 \leq w < 1.5$	0
B_{H3}	b=`'Sony'` $\wedge\ 0 \leq w < 1.5$	1
B_{H4}	b=`'Acer'` $\wedge\ 1.5 \leq w < 3$	1
B_{H5}	b=`'Dell'` $\wedge\ 1.5 \leq w < 3$	0
B_{H6}	b=`'Sony'` $\wedge\ 1.5 \leq w < 3$	1
B_{H7}	b=`'Acer'` $\wedge\ 3 \leq w < 5$	0
B_{H8}	b=`'Dell'` $\wedge\ 3 \leq w < 5$	0
B_{H9}	b=`'Sony'` $\wedge\ 3 \leq w < 5$	0

(a) Summary of Hardware.com (c_H)

ID	Query bucket for c_R	$m(B_k, c_R)$
B_{R1}	b=`'Acer'` $\wedge\ r$=`'A'`	1
B_{R2}	b=`'Dell'` $\wedge\ r$=`'A'`	0
B_{R3}	b=`'Sony'` $\wedge\ r$=`'A'`	1
B_{R4}	b=`'Acer'` $\wedge\ r$=`'B'`	1
B_{R5}	b=`'Dell'` $\wedge\ r$=`'B'`	0
B_{R6}	b=`'Sony'` $\wedge\ r$=`'B'`	1
B_{R7}	b=`'Acer'` $\wedge\ r$=`'C'`	0
B_{R8}	b=`'Dell'` $\wedge\ r$=`'C'`	1
B_{R9}	b=`'Sony'` $\wedge\ r$=`'C'`	0

(b) Summary of Review.com (c_R)

ID	Query bucket for c_D	$m(B_k, c_D)$
B_{D1}	$0 \leq w < 1.5$	1
B_{D2}	$1.5 \leq w < 3$	1
B_{D3}	$3 \leq w < 5$	1

(c) Summary of Dell.com (c_D)

Fig. 2. Example summaries of "Hardware.com", "Dell.com", and "Review.com"

content summaries. Query capability of an e-catalog refers to its query interface specification. It defines the set of attributes where input parameters can be bound with (e.g., attributes listed in a Web form when accessing an e-catalog), and the set of attributes that describe answers returned by the e-catalog (see [14] for more details). In our example, an e-catalog can be queried on the same set of attributes as the ones describing answers returned from it. As shown in Table 1, "Hardware.com" and "Dell.com" are laptop sellers and can be queried with attributes over R_c except r (by default, brand is `'Dell'` for "Dell.com"). "Review.com" can be queried over *model_number*, *brand* and *review*.

A Multi-Attribute Content (MAC) summary of an e-catalog c consists of a list of query buckets and each query bucket is associated with a boolean value. A *query bucket* is a conjunctive formula consisting of equality and order comparisons among pre-defined constants and/or attribute values over selected attributes from community relation. An example query bucket defined over the `laptop` community relation is B_k:b=`'Acer'` $\wedge 0 \leq w < 1.5$. The boolean value associated with B_k, denoted by $m(B_k, c)$, indicates whether c has answers to B_k. In other words, let $B_k(c)$ be the set of answers to B_k from c. If $B_k(c) \neq \emptyset$,

Table 2. Example AVGs of summary attributes

Attribute	AVGs
brand	'Acer', 'Dell', and 'Sony'
weight	$0 < w \leq 1.5$, $1.5 < w \leq 3$, and $3 < w \leq 5$
review	'A', 'B', and 'C'

$m(B_k, c) = 1$, and $m(B_k, c) = 0$ otherwise. MAC summaries of the three example e-catalogs are given in Figure 2.

To construct MAC summaries for e-catalogs, community administrator needs to select *summary attributes* on which the conjunctive range conditions are to be specified. Note that, not all attributes in community relation R_c are necessary to be included in catalog summaries. Let A_i be a summary attribute selected by the community administrator and let D_i be the domain of A_i. To define range conditions, D_i is divided into a set of non-overlapping ranges known as *Attribute Value Groups* (AVGs). For nominal attributes, an AVG contains one or more distinct nominal values; for continuous attributes, an AVG specifies a value range. Note that, union of AVGs of A_i is equivalent to D_i. Table 2 lists examples of AVGs used in this paper, assuming *brand*, *weight* and *review* are selected as summary attributes. The AVGs of attributes can be either manually defined by community administrator or mined from query logs using techniques such as the one presented in [6].

Once AVGs are defined, query buckets can be automatically generated on the basis of the cartesian products of AVGs of the summary attributes supported by an e-catalog. For example, "Hardware.com" supports b and w (among the three summary attributes), 9 query buckets are generated as shown in Figure 2(a) since b and w each has 3 AVGs (see Table 2). With query buckets generated, the task of building MAC summary is to associate each query bucket with a boolean value, where 1 means that the e-catalog has products matching the query bucket and 0 otherwise (see Section 5). For instance, summaries in Figure 2 show that "Hardware.com" has laptops branded 'Acer' with weight between 1.5 and 3kg and 'Sony' with weight less than 3kg, while "Dell.com" has laptops in all weight ranges.

4 Catalog Selection Using MAC Summary

Catalog selection operation aims at selecting relevant e-catalogs to answer a query. Given a user query, a query rewriting algorithm in e-catalog community first identifies the set of e-catalogs, when put together, can answer the query [3]. Based on e-catalog's query capability, the query rewriting algorithm is also responsible for generating one or more query plans in the form of $Q = Q_1 \wedge Q_2 \wedge \ldots \wedge Q_q$ where Q_p $(1 \leq p \leq q)$ is a sub-query to be answered by exactly one e-catalog. With local summaries, given a sub-query and a MAC summary, catalog selection operation returns a boolean decision whether the e-catalog has answers to the sub-query or not. A query plan needs to be executed only if all e-catalogs involved in the query plan have answers to their corresponding sub-queries.

Let $A_1, A_2, \ldots, A_m$ be summary attributes and $A_{m+1}, \ldots, A_n$ $(n > m)$ be non-summary attributes on R_c. A sub-query Q_p may specify conditions on summary attributes only, non-summary attributes only, or both summary and non-summary attributes. Let Q_p^s be the part of Q_p specifying conditions on summary attributes and Q_p^{ns} be the part of Q_p specifying conditions on non-summary attributes. With content summary, it is possible to predict whether an e-catalog c is likely to have answers to Q_p^s only. Therefore, c is said to have no answer to Q_p if either of the two conditions are satisfied: (i) $Q_p^s(c) = \emptyset$ and $Q_p = Q_p^s \wedge Q_p^{ns}$ and (ii) $Q_p^s(c) = \emptyset$ and $Q_p = Q_p^s$. The key task here is to determine whether c has answers to Q_p^s. To simplify our discussion, in the following, query Q refers to Q_p^s to be answered by one e-catalog only unless otherwise specified.

4.1 Query Relationships

Given an e-catalog c and a query Q, let $Q(c)$ be the set of answers to Q from c. Given any two queries Q_1 and Q_2, we consider two relationships between them, namely, containment and overlapping. Query containment is a fundamental concept in database systems and has been used in data integration systems and query optimization [17,24]. Both containment and overlapping[4] relationships are decidable given e-catalogs' query specification [10,12].

- *Containment*
  Q_1 is *contained* in Q_2, written $Q_1 \sqsubseteq Q_2$, if $Q_1(c)$ is a subset of $Q_2(c)$ for any e-catalog c defined over the community relation $R_c(A_1, A_2, \ldots, A_n)$.
- *Overlapping*
  Q_1 and Q_2 are overlapping if there exist at least one e-catalog c, such that $Q_1(c) \cap Q_2(c) \neq \emptyset$. In this paper, we say Q_1 and Q_2 are overlapping only if neither one is contained in another.

We use two query examples on the community relation `laptop` to illustrate relationships between a query and query buckets[5]: Q_1: $w \geq 2.5$ and Q_2: $b =$ `'Acer'` $\wedge w \leq 1$. Given the 9 query buckets defined for "Hardware.com", Q_1 contains B_{H7}, B_{H8}, and B_{H9} and overlaps with B_{H4}, B_{H5}, and B_{H6}; Q_2 is contained in B_{H1} (assuming that w is always ≥ 0).

4.2 E-Catalog Selection

By construction of MAC summary, given a query Q, if $Q(c) \neq \emptyset$, there is a finite union of query buckets, denoted by B_Q, such that $Q(c) \subseteq B_Q(c)$, i.e., $Q \sqsubseteq B_Q$. The relationship between any bucket $B_k \in B_Q$ and Q can be: (i) B_k overlaps with Q, (ii) $B_k \sqsubseteq Q$, or (iii) $Q \sqsubseteq B_k$. For instance, given Q: $1 < w < 2$ and the MAC summary for "Dell.com" (see Figure 2(c)), $B_Q = \{B_{D1}, B_{D2}\}$. Whether c has answers to Q can therefore be predicted based on whether c has answers to

[4] Which can be reduced to satisfiability.

[5] Note that a query bucket can be treated as a conjunctive query by its definition.

any $B_k \in B_Q$. That is, if c has no answer to any $B_k \in B_Q$, it is clear that c has no answer to Q because $Q(c) \subseteq B_Q(c)$. However, if c has answers for at least one $B_k \in B_Q$, c may have answers to Q and hence c should be selected to answer Q to avoid false negative selection. In short, c is selected to answer a query Q iff $Q \sqsubseteq B_Q$ and $\exists B_k \in B_Q$ s.t. $m(B_k, c) = 1$.

Given a query Q, the problem of selection of e-catalogs reduces to the problem of finding B_Q containing Q. The cost of finding B_Q is $O(n)$ where n is the number of query buckets in the MAC summary in the worst case (if query buckets are not indexed). In the following, we use two example queries to illustrate how the summary can be used in e-catalog selection.

- Q_1 : `find laptops with weight greater than 3kg`
  Based on query capability, both "Hardware.com" (c_H) and "Dell.com" (c_D) can answer the query. The two query plans are $Q_1(c_H)$ and $Q_1(c_D)$ respectively. Given the summary of c_H shown in Figure 2, Q_1 is contained in $B_Q = \{B_{H7}, B_{H8}, B_{H9}\}$, and none of them is associated with 1. $Q_1(c_H)$ therefore leads to no answer and is not necessary to be executed. For c_D, Q_1 is contained in B_{D3} and $m(B_{D_3}, c_D) = 1$. Hence $Q_1(c_D)$ is the only query plan that may lead to answers.
- Q_2 : `find laptops with weight less than 1kg and review is 'A'`
  As review can only be found in "Review.com" (c_R) and weight can be found in c_H or c_D, the two query plans are: (i) $Q_{21}(c_H) \wedge Q_{22}(c_R)$ and (ii) $Q_{21}(c_D) \wedge Q_{22}(c_R)$, where $Q_{21} : w < 1$ and $Q_{22} : r =$`'A'`. Based on the summaries, $Q_{21}(c_H)$,$Q_{21}(c_D)$, and $Q_{22}(c_R)$ all have answers. Both the two query plans may lead to answers.

5 Building and Updating MAC Summary

Before we describe how to build MAC summary, we first discuss how an existing MAC summary can be updated.

E-catalogs are dynamic and their contents are updated by their providers from time to time. To keep the content summaries up to date, the summaries have to be updated accordingly. In this section, we first discuss the type of changes and their effects on MAC summaries. We then discuss how MAC summary can be updated with query answers from e-catalogs.

5.1 MAC Summary Update

Content updates[6] to e-catalog contents can be classified into four categories. For each category, we discuss how the MAC summary can be affected.

- E-catalog changes tuple values of non-summary attributes. The summary is not affected as the non-summary attributes are not involved in the summary.

[6] Note that, our discussion limits to content updates only and not schema updates. If an e-catalog updates its schema (e.g., add or remove supported summary attributes), the query buckets may have to be regenerated and the summary is then rebuilt.

- E-catalog deletes tuples. Let t be a deleted tuple and $t \in B_k(c)$. After deletion, if there is at least one tuple t' in c such that $t' \in B_k(c)$, the summary is not affected as $m(B_k, c) = 1$. However, if there is no such a tuple t' in c, the e-catalog c may be false positively selected to answer queries contained in (or overlapping with) B_k as $m(B_k, c) = 1$ in the summary has not been updated with $m(B_k, c) = 0$.
- E-catalog adds new tuples. Let t be the newly added tuple and $t \in B_k(c)$. If $m(B_k, c) = 1$ in the summary, the summary is not affected. However, if $m(B_k, c) = 0$, catalog c will be false negatively selected to answer queries contained in (or overlapping with) B_k and user may miss the newly added tuple before $m(B_k, c)$ is updated to 1.
- E-catalog changes summary attribute values of tuples. This is equivalent to delete old tuples (with old attribute values) and add new tuples with updated attribute values.

From the above discussion, it is clear that not every tuple insertion/deletion results in invalidation of the summary, thus MAC summary is relatively resilient to changes. Nevertheless, user may miss the newly inserted tuples without summary updating. Deleting tuples from e-catalogs, on the other hand, only increases the chance of an e-catalog being false positively selected and will not lead to missing answers for users.

A straightforward approach to update a catalog summary is to periodically rebuild it. However, it is always hard to define the update frequency. In this paper, we propose to update MAC summary using the answers returned from e-catalogs.

A catalog summary should be updated by changing $m(B_k, c)$ from 0 to 1 if there is at least one tuple t which is inserted in c and $t \in B_k(c)$ or changing $m(B_k, c)$ from 1 to 0, if there is no t such that $t \in B_k(c)$ after tuple deletion. As $B_k(c)$ can be derived or partially derived from $Q(c)$ if B_k is contained in (or overlaps with) Q, $m(B_k, c)$ can be effectively updated based on $Q(c)$. More specifically, given a query Q, summary of e-catalog c can be updated by using $Q(c)$ with two rules:

- A query bucket B_k is contained in Q: if there exist at least one tuple $t \in Q(c)$, and $t \in B_k(c)$, then set $m(B_k, c)$ to 1; and if there is no such a tuple in $Q(c)$, set $m(B_k, c)$ to 0.
- A query bucket B_k overlaps with Q: if there exist at least one tuple $t \in Q(c)$ and $t \in B_k(c)$, then set $m(B_k, c)$ to 1; if there is no such a tuple in $Q(c)$, $m(B_k, c)$ remains unchanged.

With the above rules, query buckets contained in (or overlapping with) frequently asked queries are updated more frequently. The cost of the updating is $O(mn)$ where m is the number of query buckets in B_Q where $Q \sqsubseteq B_Q$ and n is the number of answers to Q.

Consider the query Q_1: $w \geq 2.5$ and the catalog summary of "Hardware.com" in Figure 2. Let us say there are 10 tuples returned from "Hardware.com" for Q_1, and one of them is t_1: b=`'Acer'` $\wedge$ w=4.2. $t_1 \in B_{H7}$ where B_{H7}: b=`'Acer'` $\wedge 3 < w \leq 5$. However, 0 was associated with B_{H7} in the summary of "Hardware.com".

This means at least one tuple (e.g., t_1) has been inserted into "Hardware.com" and the summary needs to be updated by changing the boolean value associated with B_{H7} to 1. Similarly, if the answers to Q_1 returned from "Dell.com" do not contain any tuple covered by B_{D3}: $3 < w \leq 5$, the boolean value needs to be changed to 0 in the summary of "Dell.com".

The above update operations can be further extended to update AVGs of summary attributes and query bucket definitions (and hence making the MAC summary relatively adaptive). For instance, in answering the query Q_1: $w \geq 2.5$, "Hardware.com" returns a tuple t_f with b=`'Fujitsu'` $\wedge w = 3.3$. As "Fujitsu" is not covered in the defined domain of *brand*, a new AVG is formed with value "Fujitsu" to extend the domain of *brand*. With this newly formed AVG, a new set of query buckets are defined with combinations of AVGs of other summary attributes. The boolean values associated with these newly added query buckets are set to 1 to avoid false negative selection. Nevertheless, these values can be updated during query processing.

5.2 Building Catalog Summary

In MAC summary, query buckets are defined once summary attributes are selected and their AVGs are defined. Given an e-catalog, the task of building its summary is to associate a boolean value with each query bucket in the summary. In our framework, there are two approaches to build a catalog summary. A *lazy* approach is to utilize the update operation and build catalog summaries during the query answering process. The other is the *eager* approach to build summaries with sample range queries.

The lazy approach of building a catalog summary starts with the assumption that an e-catalog has answers to all query buckets. That is, associates each query bucket with a boolean value of 1. The summary is updated during the query-answering process as discussed in Section 5.1. With this approach, the building of catalog summary leads to no additional cost (e.g., answering sample queries) to an e-catalog. Nevertheless, an e-catalog may often be false positively selected to answer queries that are not relevant to it during the summary building process.

The eager approach is to send each query bucket (a range query) directly to an e-catalog and find out whether the e-catalog has answers or not. If the e-catalog replies with "zero match found", a 0 is associated with the query bucket; otherwise 1 is associated with the query bucket. Note that, no actual answer needs to be retrieved from e-catalogs for each query bucket as MAC summary of an e-catalog is a list of boolean values. The only assumption here is that an e-catalog is able to indicate whether or not it has products matching the range query.

6 Experiments

We implemented MAC summary structure in Java and simulated e-catalogs with MS Access databases using `adult` dataset from UCI Repository[7]. After cleaning,

[7] `http://www.ics.uci.edu/~mlearn/MLSummary.html`

Table 3. Summary attributes

Name	Type	# AVGs
age	continuous	6
education	nominal	16
hour_per_week	continuous	7
marital_status	nominal	7

45,221 tuples were randomly divided into 50 e-catalogs with about 904 tuples in each e-catalog. Two continuous and two nominal attributes were selected as summary attributes; the AVGs of the 4 summary attributes were arbitrarily created and the number of AVGs for each summary attribute is given in Table 3. All simulated e-catalogs support these 4 summary attributes. With the defined AVGs, the number of query buckets in MAC summary is 4704. To evaluate the MAC summaries of the 50 e-catalogs, we randomly generated 5000 user queries. Each query may specify condition on any one summary attribute, or any combination of two, three or four summary attributes. The numbers of queries with conditions on one, two, three or four attributes are the same.

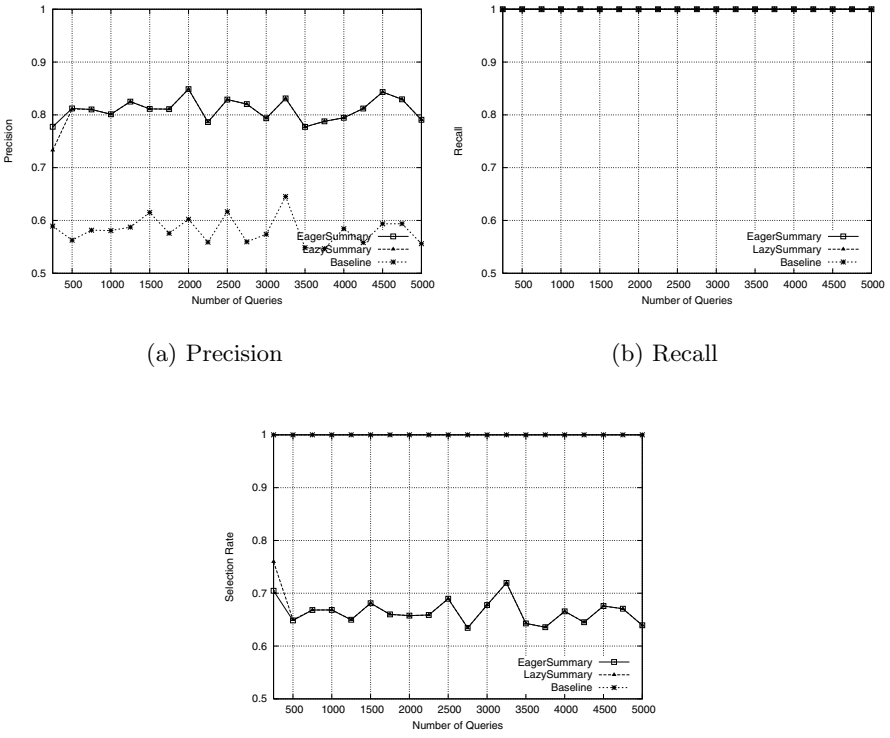
We adopted measures from Information Retrieval including *precision* and *recall* to evaluate e-catalog selection. Precision is the proportion of selected e-catalogs having answers to a given query; 100% precision means that all the selected e-catalogs have answers to the query. Recall is the proportion of the selected e-catalogs among all the e-catalogs having answers to a given query; 100% recall means all e-catalogs having answers are selected to answer the query. We also report *selection rate* referring to the percentage of e-catalogs selected to answer a given query among all available e-catalogs; 100% selection rate means all e-catalogs are selected to answer a query.

6.1 Building of MAC Summary

As eager approach is straightforward, we only experimented lazy approach in building MAC summary, with an assumption that there were no content updates in e-catalogs. Starting with a blank summary where each query bucket was associated with a boolean value 1, MAC summary was built with update operations when processing user queries. After answering a user query, some query buckets were updated: the summary was about 90% matching the final state after answering 100 user queries and was fully built after answering 387 user queries. The number of queries used in building the summaries was much less than the number of query buckets. We call the summary built using lazy approach *lazy summary*.

6.2 Evaluation of E-Catalog Selection

We conducted two sets of experiments to evaluate the accuracy of e-catalog selection using MAC summaries. In the first set of experiments, e-catalogs did



(a) Precision

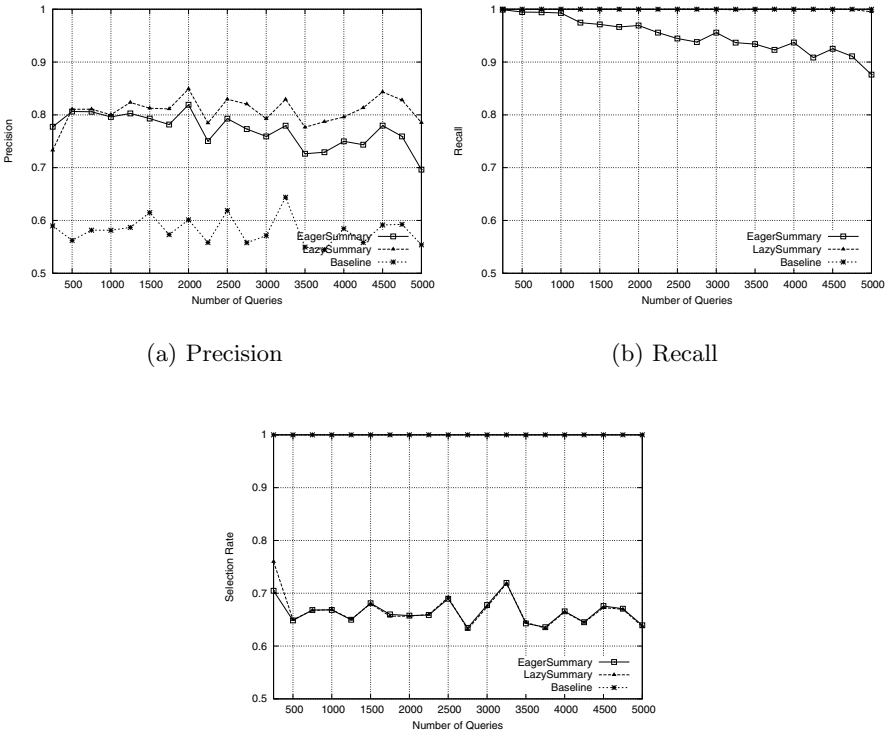
(b) Recall

(c) Selection Rate

Fig. 3. Precision, Recall, and Selection Rate with no content updates in e-catalogs

not update their contents. In the second set of experiments, we simulated content updates in e-catalogs. For each set of experiments, both eager and lazy approaches were experimented. With eager approach, the MAC summary was pre-built before evaluation using the 5000 user queries and the summary was not updated regardless whether there was content updates in the e-catalogs. With lazy approach, the summary was built and updated during query processing. Note that the same 5000 user queries were used in all experiments. Both the eager and lazy approaches were also compared with baseline approach where a query is simply forwarded to all e-catalogs.

The precision, recall and selection rate using eager summary, lazy summary and baseline approaches when there was no content update in e-catalogs are shown in Figure 3. The values reported are the average of every 250 queries. As shown in Figure 3(a), baseline approach clearly delivered lowest precision by selecting all e-catalogs to answer a query. This experimental results also demonstrated that not all e-catalogs would have answers to a given query. Lazy summary delivered slight lower precision for the first 500 user queries and the



(a) Precision

(b) Recall

(c) Selection Rate

Fig. 4. Precision, Recall, and Selection Rate with content updates in e-catalogs

same precision for the rest of user queries. The reason is that before the lazy summary was fully built, some query buckets were incorrectly associated with 1 and therefore e-catalogs were false positively selected to answer queries. After fully built, lazy summary and eager summary became identical and yielded the same precision. The overall precision was about 80% which was fairly high. As shown in Figure 3(b), both eager and lazy summaries gave 100% recall which means that all the e-catalogs having answers for any user query were selected. These results confirmed our discussion in Section 5.1 and false negative selection only occurs when e-catalogs add new tuples to their content. The selection rate for both eager and lazy summaries were on average below 70%, and 100% for baseline approach as shown in Figure 3(c).

We simulated content updates in e-catalogs by swapping tuples among neighboring e-catalogs. After evaluation of every 50 user queries, 5 tuples were deleted from c_j and inserted to c_{j+1}. After processing the 5000 evaluation queries, in total 500 tuples (i.e., 55% of the 904 tuples in each catalog) were swapped among neighboring e-catalogs. With eager approach, the summary was pre-built

before evaluating the user queries, and once built, the summary was not updated. With lazy approach, the summary was built and updated during the process of answering 5000 user queries.

The precision, recall and selection rate with content updates in e-catalogs are shown in Figure 4. For the first 250 queries, eager summary yielded much better precision than lazy summary as shown in Figure 4(a) since lazy summary was not fully built. Comparable precisions were observed after processing 500 queries. When more queries processed (and more content updates in e-catalogs performed), precision of lazy summary became better than that of eager summary as lazy summary was updated during the query processing. The difference between the two precisions became wider with more content updates occurred in e-catalogs. After processing the 5000 queries, lazy summary (with updates) was able to maintain its precision at around 80% while precision of eager summary (without updates) dropped to around 70%. The precision of baseline approach was again the lowest. For recall measure, as shown in Figure 4(b), lazy summary with updates achieved 99.96% which was nearly 100%. Such an experimental result gives strong evidence on the effectiveness of MAC summary update operation. Without summary updates, recall of eager summary kept on dropping when e-catalog updated their contents. After processing all queries, the recall dropped below 90%[8]. The recall of baseline approach was 100% as all e-catalogs were selected to answer queries. On the selection rate, as shown in Figure 4(c), there were no much difference between eager and lazy approaches as e-catalog updates were simulated by swapping tuples among neighbors and the number of e-catalogs having answers to a query did not change much.

Three observations can be made from these experiments: (1) summary update was crucial in maintaining summaries accuracy, (ii) MAC summary was resilient to changes, and (iii) recall in e-catalog selection was fairly high meaning that user is on low risk to miss answers from e-catalogs. Note that, as the e-catalogs were simulated using local databases in our experiments, the effectiveness of communication cost reduction using MAC summary was not studied.

7 Conclusion and Future Work

We studied the problem of e-catalogs selection by considering both their query capability and content. For each e-catalog, a catalog summary is built to reflect its data distribution and the catalog community selects only the relevant e-catalogs to answer user queries. We proposed a multi-attribute catalog summary structure, and discussed how to build/update MAC summary for an e-catalog and use MAC summary in e-catalog selection. Our experiments demonstrated that MAC summaries could be effectively used in e-catalog selection with fairly high precision and nearly 100% recall.

[8] Note that eager summary might deliver comparable or better selection accuracy if periodical updates were performed. However, choosing the optimal update frequency is not trivial as the update frequencies of the member catalogs have to be derived.

A limitation of the proposed MAC summary is that it is not adaptive. In other words, once summary attributes are selected and AVGs are defined, the query buckets are fixed. In our future research, we plan to investigate adaptive MAC summary such that the MAC summaries can be dynamically adjusted with respect to both e-catalog data distribution and user query distribution.

Acknowledgement

The work is partially funded by Australian Research Council Discovery Grant DP0452942. In addition, the authors would like to thank Quang-Khai Pham for his suggestions and effort in conducting experiments.

References

1. R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison Wesley, 1st edition, May, 1999.
2. K. Baina, B. Benatallah, H.-Y. Paik, F. Toumani, C. Rey, A. Rutkowska, and H. Susanto. WS-CatalogNet: An infrastructure for creating, peering, and querying e-catalog communities. In *Proc. of VLDB'04*, pages 1325–1328, Toronto, Canada, August 2004.
3. B. Benatallah, M.-S. Hacid, H.-Y. Paik, C. Rey, and F. Toumani. Towards semantic-driven, flexible and scalable framework for peering and quering e-catalog communities. *Information Systems*, 31(4):266 – 294, 2006.
4. S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks*, 30(1-7):107–117, 1998.
5. J. Caverlee, L. Liu, and D. Rocco. Discovering and ranking web services with basil: a personalized approach with biased focus. In *Proc. of ICSOC'04*, pages 153–162, New York, NY, 2004. ACM Press.
6. K. Chakrabarti, S. Chaudhuri, and S. won Hwang. Automatic categorization of query results. In *Proc. of ACM SIGMOD'04*, pages 755–766, Paris, France, June 2004. ACM Press.
7. X. Cheng, G. Dong, T. Lau, and J. Su. Data integration by describing sources with constraint databases. In *Proc. of ICDE'99*, Sydney, Australia, 1999. IEEE Computer Society.
8. J. G. Conrad and J. R. S. Claussen. Early user—system interaction for database selection in massive domain-specific online environments. *ACM Trans. Inf. Syst.*, 21(1):94–131, 2003.
9. J. Fan and S. Kambhampati. A snapshot of public web services. *SIGMOD Record*, 34(1):24–32, 2005.
10. E. Gelle and B. Faltings. Solving mixed and conditional constraint satisfaction problems. *Constraints*, 8(2):107–141, 2003.
11. A. Y. Halevy. Answering queries using views: A survey. *VLDB Journal*, 10(4):270–294, 2001.
12. O. H. Ibarra and J. Su. On the containment and equivalence of database queries with linear constraints (extended abstract). In *Proc. of PODS'97*, pages 32–43, Tucson, Arizona, 1997. ACM Press.
13. D. H. Lee and M. H. Kim. Database summarization using fuzzy isa hierarchies. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 27(1):68–78, 1997.

14. A. Y. Levy, A. Rajaraman, and J. J. Ordille. Querying heterogeneous information sources using source descriptions. In *Proc. of VLDB'96*, pages 251–262, Bombay, India, 1996. Morgan Kaufmann.
15. L. Liu. Query routing in large-scale digital library systems. In *Proc. of ICDE'99*, pages 154–163, Washington DC, 1999. IEEE Computer Society.
16. R. McCann, B. K. AlShebli, Q. Le, H. Nguyen, L. Vu, and A. Doan. Mapping maintenance for data integration systems. In *Proc. of VLDB'05*, Trondheim, Norway, 2005.
17. T. Millstein, A. Levy, and M. Friedman. Query containment for data integration systems. In *Proc. of PODS'00*, pages 67–75, Dallas, Texas, 2000. ACM Press.
18. Z. Nie, S. Kambhampati, and U. Nambiar. Effectively mining and using coverage and overlap statistics for data integration. *IEEE Trans. on Knowledge and Data Eng.*, 17(5):638–651, May 2005.
19. OCEAN. On-board Communication, Entertainment, And iNformation. Available at `http://www.ocean.cse.unsw.edu.au`.
20. A. L. Powell and J. C. French. Comparing the performance of collection selection algorithms. *ACM Trans. Inf. Syst.*, 21(4):412–456, 2003.
21. R. Saint-Paul, G. Raschia, and N. Mouaddib. General purpose dataset summarization. In *Proc. of VLDB'05*, Trondheim, Norway, 2005.
22. J. D. Ullman. Information integration using logical views. In *Proc. of ICDT'97*, pages 19–40, Delphi, Greece, January 1997.
23. C. T. Yu, G. Philip, and W. Meng. Distributed top-n query processing with possibly uncooperative local systems. In *Proc. of VLDB'03*, pages 117–128, Berlin, Germany, September 2003. Morgan Kaufmann.
24. C. Zhang, J. Naughton, D. DeWitt, Q. Luo, and G. Lohman. On supporting containment queries in relational database management systems. *SIGMOD Rec.*, 30(2):425–436, 2001.

WorkflowNet2BPEL4WS: A Tool for Translating Unstructured Workflow Processes to Readable BPEL

Kristian Bisgaard Lassen[1] and Wil M.P. van der Aalst[1,2]

[1] Department of Computer Science, University of Aarhus, IT-parken, Aabogade 34, DK-8200 Aarhus N, Denmark
k.b.lassen@daimi.au.dk
[2] Department of Information Systems, Eindhoven University of Technology, P.O. Box 513, NL-5600 MB, Eindhoven, The Netherlands
w.m.p.v.d.aalst@tm.tue.nl

Abstract. This paper presents *WorkflowNet2BPEL4WS* a tool to automatically map a graphical workflow model expressed in terms of Workflow Nets (WF-nets) onto BPEL. The *Business Process Execution Language for Web Services* (BPEL) has emerged as the de-facto standard for implementing processes and is supported by an increasing number of systems (cf. the IBM WebSphere Choreographer and the Oracle BPEL Process Manager). While being a powerful language, BPEL is difficult to use. Its XML representation is very verbose and only readable for the trained eye. It offers many constructs and typically things can be implemented in many ways, e.g., using links and the flow construct or using sequences and switches. As a result only experienced users are able to select the right construct. Some vendors offer a graphical interface that generates BPEL code. However, the graphical representations are a direct reflection of the BPEL code and not easy to use by end-users. Therefore, we provide a mapping from WF-nets to BPEL. This mapping builds on the rich theory of Petri nets and can also be used to map other languages (e.g., UML, EPC, BPMN, etc.) onto BPEL. To evaluate WorkflowNet2BPEL4WS we used more than 100 processes modeled using Protos (the most widely used business process modeling tool in the Netherlands), automatically converted these into CPN Tools, and applied our mapping. The results of these evaluation are very encouraging and show the applicability of our approach.

Keywords: BPEL4WS, Petri nets, workflow management, business process management.

1 Introduction

The primary goal of this paper is to present *WorkflowNet2BPEL4WS*, a tool to automatically map *Workflow nets* (WF-nets) [1,2,3] onto *Business Process Execution Language for Web Services* (BPEL) [10]. The tool uses the approach described in [9] and assumes a WF-net modeled using CPN Tools [11,27] and the resulting BPEL code can be used in systems such as the IBM WebSphere Choreographer [22] and the Oracle BPEL Process Manager [32]. Note that the approach is also applicable to other Petri-net-based tools (e.g., systems such as ProM [12], Yasper, WoPeD, and Protos [36] that are able to export *Petri Net Markup Language* (PNML)). Moreover, the ideas are also applicable

R. Meersman, Z. Tari et al. (Eds.): OTM 2006, LNCS 4275, pp. 127–144, 2006.

to other graph-based languages such as *Business Process Modeling Notation* (BPMN) [44], UML activity diagrams [21], *Event-driven Process Chains* (EPCs) [24, 37], etc.

This introduction motivates the need for a tool like *WorkflowNet2BPEL4WS*. To do this we start by introducing BPEL followed by a brief discussion of WF-nets, an introduction to the tool, and some information about the evaluation using 100 Protos models.

1.1 BPEL

After more than a decade of attempts to standardize workflow languages (cf. [5, 31]), it seems that the *Business Process Execution Language for Web Services* (*BPEL4WS* or *BPEL* for short) [10] is emerging as the de-facto standard for executable process specification. Systems such as Oracle BPEL Process Manager, IBM WebSphere Application Server Enterprise, IBM WebSphere Studio Application Developer Integration Edition, and Microsoft BizTalk Server 2004 support BPEL, thus illustrating the practical relevance of this language.

Interestingly, BPEL was intended initially for cross-organizational processes in a web services context: "BPEL4WS provides a language for the formal specification of business processes and business interaction protocols. By doing so, it extends the Web Services interaction model and enables it to support business transactions." (see page 1 in [10]). However, it can also be used to support intra-organizational processes. The authors of BPEL [10] envision two possible uses of the language: "Business processes can be described in two ways. Executable business processes model actual behavior of a participant in a business interaction. Business protocols, in contrast, use process descriptions that specify the mutually visible message exchange behavior of each of the parties involved in the protocol, without revealing their internal behavior. The process descriptions for business protocols are called abstract processes. BPEL4WS is meant to be used to model the behavior of both executable and abstract processes." In this paper we only consider the use of BPEL as an execution language.

BPEL is an expressive language [45] (i.e., it can specify highly complex processes) and is supported by many systems. Unfortunately, BPEL is not a very intuitive language. Its XML representation is very verbose and there are many, rather advanced, constructs. Clearly, it is at another level than the graphical languages used by the traditional workflow management systems (e.g., Staffware, FileNet, COSA, Lotus Domino Workflow, SAP Workflow, etc.). This is the primary motivation of this for developing WorkflowNet2BPEL4WS, i.e., a tool to generate BPEL code from a graphical workflow language.

The modeling languages of traditional workflow management systems are executable but at the same time they appeal to managers and business analysts. Clearly, managers and business analysts will have problems understanding BPEL code. As a Turing complete[1] language BPEL can do, well, anything, but to do this it uses two styles of modeling: graph-based and structured. This can be explained by looking at its history: BPEL builds on IBM's WSFL (Web Services Flow Language) [28] and Microsoft's XLANG (Web Services for Business Process Design) [39] and combines accordingly the features

[1] Since BPEL offers typical constructs of programming languages, e.g., loops and if-the-else constructs, and XML data types it is easy to show that BPEL is Turing complete.

of a block structured language inherited from XLANG with those for directed graphs originating from WSFL. As a result simple things can be implemented in two ways. For example a sequence can be realized using the `sequence` or `flow` elements, a choice based on certain data values can be realized using the `switch` or `flow` elements, etc. However, for certain constructs one is forced to use the block structured part of the language, e.g., a *deferred choice* [7] can only be modeled using the `pick` construct. For other constructs one is forced to use the links, i.e., the more graph-based oriented part of the language, e.g., two parallel processes with a one-way synchronization require a `link` inside a `flow`. In addition, there are very subtle restrictions on the use of links: "A link MUST NOT cross the boundary of a while activity, a serializable scope, an event handler or a compensation handler... In addition, a link that crosses a fault-handler boundary MUST be outbound, that is, it MUST have its source activity within the fault handler and its target activity within a scope that encloses the scope associated with the fault handler. Finally, a link MUST NOT create a control cycle, that is, the source activity must not have the target activity as a logically preceding activity, where an activity A logically precedes an activity B if the initiation of B semantically requires the completion of A. Therefore, directed graphs created by links are always acyclic." (see page 64 in [10]). All of this makes the language complex for end-users. Therefore, there is a need for a "higher level" language for which one can generate *intuitive* and *maintainable* BPEL code.

Such a "higher level" language will not describe certain implementation details, e.g., particularities of a given legacy application. This needs to be added to the generated BPEL code. *Therefore, it is important that the generated BPEL code is intuitive and maintainable.* If the generated BPEL code is unnecessary complex or counter-intuitive, it cannot be extended or customized.

Note that tools such as Oracle BPEL Process Manager and IBM WebSphere Studio offer graphical modeling tools. *However, these tools reflect directly the BPEL code*, i.e., the designer needs to be aware of structure of the XML document and required BPEL constructs. For example, to model a *deferred choice* in the context of a parallel process [7] the user needs to add a level to the hierarchy (i.e., a `pick` defined at a lower level than the `flow`). Moreover, subtle requirements such as links not creating a cycle still need to be respected in the graphical representation. Therefore, it is interesting to look at a truly graph-based language with no technological-oriented syntactical restrictions and see whether it is possible to generate BPEL code.

1.2 WF-Nets

In this paper we use a specific class of Petri nets, named *WorkFlow nets* (WF-nets) [1,2,3], as a *source language* to be mapped onto the *target language* BPEL. There are several reasons for selecting Petri nets as a source language. It is a simple graphical language with a strong theoretical foundation. Petri nets can express all the routing constructs present in existing workflow languages [4,17,42] and enforce no technological-oriented syntactical restrictions (e.g., no loops). Note that WF-nets are classical Petri nets without data, hierarchy, time and other extensions. Therefore, their applicability is limited. However, we do *not* propose WF-nets as the language to be used by end-users; we merely use it as the theoretical foundation. It can capture the control-flow

structures present in other graphical languages, but it abstracts from other aspects such as data flow, work distribution, etc. Note that there are many Petri-net based modeling tools, e.g., general tools such as ExSpect, CPN Tools, etc. and more dedicated Petri-net-based workflow modeling and analysis tools such COSA, Protos, WoPeD, Yasper, and Protos. Clearly, these tools can be used to model WF-nets (possibly extended with time, data, hierarchy, etc.). Moreover, as demonstrated in the context of tools such as ProM [12] and Woflan [41], it is possible to map (abstractions of) languages like Staffware, MQSeries Workflow, EPCs, YAWL, etc. onto WF-nets. Hence, the mapping presented in this paper can be used as a basis for translations from other source languages such as UML activity diagrams [21], Event-driven Process Chains (EPCs) [24,37], and the Business Process Modeling Notation (BPMN) [44]. Moreover, the basic ideas can also be used to map graph-based languages onto other (partly) block-structured languages.

1.3 WorkflowNet2BPEL4WS

In a technical report [9], we introduced an approach to automatically map a WF-net onto BPEL using an iterative approach. To support this approach, we implemented the tool *WorkflowNet2BPEL4WS*. This tool automatically translates Colored Petri Nets (CPNs, [27]) into BPEL code. These CPNs are specified using CPN Tools [11]. Note that a CPN may also contain detailed data transformations and stochastic information (e.g., delay distributions and probabilities). However, in the transformation, we abstract from data, time, and probabilities and mainly focus on the WF-net structure of the CPN. The code generated by WorkflowNet2BPEL4WS can be imported into any system that supports BPEL, e.g., IBM's WebSphere Studio [22].

1.4 Evaluation Using 100 Protos Models

To evaluate the applicability of our approach we used 100 processes modeled using Protos [36]. These models were automatically converted into CPN Tools using ProM [12] and then we used WorkflowNet2BPEL4WS to map them onto BPEL. Protos (Pallas Athena) uses a Petri-net-based modeling notation [36] and is a widely used business process modeling tool. It is used by more than 1500 organizations in more than 20 countries. The number of users that use Protos for designing processes is estimated to be 25000. Some of the organizations have modeled more than 1500 processes. The 100 process models used for the evaluation resulted from student projects where students had to model and redesign realistic business cases.

1.5 Outline

The remainder of this paper is organized as follows. First, we provide an overview of related work. Section 3 describes the approach used to map WF-nets onto BPEL using an iterative approach. Section 4 presents the implementation of WorkflowNet2BPEL4WS. Section 5 evaluates our approach using 100 Protos models. These models where then executed using IBM's WebSphere Studio. Section 6 concludes the paper.

2 Related Work

Since the early nineties workflow technology has matured [20] and several textbooks have been published, e.g., [6,13,23,29]. During this period many languages for modeling workflows have been proposed, i.e., languages ranging from generic Petri-net-based languages to tailor-made domain-specific languages. The Workflow Management Coalition (WfMC) has tried to standardize workflow languages since 1994 but failed to do so [17]. XPDL, the language proposed by the WfMC, has semantic problems [4] and is rarely used. In a way BPEL [10] succeeded in doing what the WfMC was aiming at. However, BPEL is really at the implementation level rather than the workflow modeling level or the requirements level (thus providing the motivation for this paper).

Several attempts have been made to capture the behavior of BPEL [10] in some formal way. Some advocate the use of finite state machines [18,19], others process algebras [16,26], and yet others abstract state machines [14,15] or Petri nets [33,30,38,40]. For a detailed analysis of BPEL based on the workflow patterns [7] we refer to [45].

The work reported in this paper is also related to the various tools and mappings used to generate BPEL code being developed in industry. Tools such as the IBM WebSphere Choreographer and the Oracle BPEL Process Manager offer a graphical notation for BPEL. However, this notation directly reflects the code and there is no intelligent mapping as shown in this paper. This implies that users have to think in terms of BPEL constructs (e.g., blocks, syntactical restrictions on links, etc.). More related is the work of Steven White that discusses the mapping of BPMN onto BPEL [43] and the work by Jana Koehler and Rainer Hauser on removing loops in the context of BPEL [25]. Note that none of these publications provides a mapping of some (graphical) process modeling language onto BPEL: [43] merely presents the problem and discusses some issues using examples and [25] focusses on only one piece of the puzzle. Also related is the mapping presented in [34] where a subclass of BPMN is mapped onto BPEL using ECA rules as an intermediate format. Then these ECA rules are realized by BPEL event handlers (onEvent). Note that this mapping heavily relies on the implementation of events in BPEL. Moreover, the resulting code is not very readable for humans because this mapping does not try to identify patterns close to the BPEL constructs. A more recent mapping tries to overcome this problem [35] but has not been implemented yet.

The tool presented in this paper uses our translation which was described in detail in a technical report [9]. The work is also related to [8] where we describe a case study where for a new bank system requirements are mapped onto Colored Workflow Nets (a subclass of Colored Petri Nets) which are then implemented using BPEL in the IBM WebSphere environment.

3 Mapping WF-Nets to BPEL

In this paper, we would like to focus on the WorkflowNet2BPEL4WS tool and the evaluation of it. Therefore, we do not show any details for the algorithms being used. Moreover, we do not give any proof of the correctness of our approach. For this we refer to the technical report [9] mentioned before. We also assume that the reader has

basic knowledge of BPEL and Petri nets. This allows us to focus on the application of our translation from WF-nets to BPEL.

As indicated in the introduction, it is important that the generated BPEL code is intuitive and maintainable. If the generated BPEL code is unnecessary complex or counter-intuitive, it cannot be extended or customized. Therefore, we try to map parts of the WF-net onto BPEL constructs that fit best. For example, a sequence of transitions connected through places should be mapped onto a BPEL `sequence`. We aim at recognizing "sequences", "switches", "picks", "while's", and "flows" where the most specific construct has our preference, e.g., for a sequence we prefer to use the `sequence` element over the `flow` element even though both are possible. We aim at an iterative approach where the WF-net is reduced by packaging parts of the network into suitable BPEL constructs.

We would like to stress that our goal is not to provide just any mapping of WF-nets onto BPEL. Note that a large class of WF-nets can be mapped directly onto a BPEL `flow` construct. However, such a translation results in unreadable BPEL code. Instead we would like to map a graph-based language like WF-nets onto a hierarchical decomposition of specific BPEL constructs. For example, if the WF-net contains a sequence of transitions (i.e., activities) this should be mapped onto the more specific `sequence` construct rather than the more general (and more verbose) `flow` construct. Hence, our goal is to generate readable and compact code.

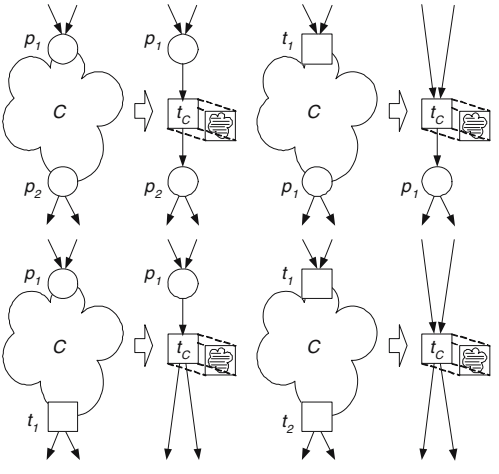


Fig. 1. Folding a component C into a single transition t_C

To map WF-nets onto (readable) BPEL code, we need to transform a graph structure to a block structure. For this purpose we use *components*. A component should be seen as a selected part of the WF-net that has a clear start and end. One can think of it as subnet satisfying properties similar to a WF-net. However, unlike a WF-net, a component may start and/or end with a transition, i.e., WF-nets are "place bordered" while components may be "place and/or transition bordered". The goal is to map components

onto "BPEL blocks". For example, a component holding a purely sequential structure should be mapped onto a BPEL `sequence` while a component holding a parallel structure should be mapped onto a `flow`. Figure 1 shows the basic idea. We try to identify a place and/or transition bordered component C and fold this into a single transition t_C. The annotation of t_C hold the BPEL code corresponding to C. By repeating this process we hope to find a single transition annotated with the BPEL code of the entire process.

So we can summarize our approach as follows: *The idea is to start with an annotated WF-net where each transition is labeled with references to primitive activities such as* `invoke` *(invoking an operation on some web service),* `receive` *(waiting for a message from an external source),* `reply` *(replying to an external source),* `wait` *(waiting for some time),* `assign` *(copying data from one place to another),* `throw` *(indicating errors in the execution), and* `empty` *(doing nothing). Taking this as starting point, a component in the annotated WF-net is mapped onto BPEL code. The component C is replaced by transition t_C whose inscription (cf. Figure 1) describes the BPEL code associated to the whole component. This process is repeated until there is just a single transition whose inscription corresponds to the BPEL specification of the entire process.* How this can be done is detailed using an example.

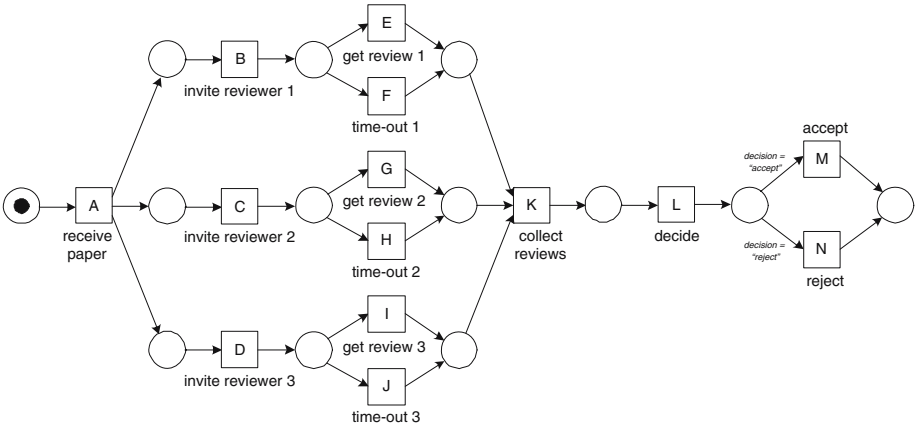


Fig. 2. A Petri net describing the process of reviewing papers

Figure 2 shows a Petri net corresponding to a reviewing process. First we look for a *maximal sequence component*, i.e., a component representing a sequence that is chosen as large as possible. This component is mapped into a new transition with the corresponding BPEL annotation. In Figure 2 there is a sequence consisting of transition K and L and we replace this component by a transition $F1$. The resulting WF-net and the annotation of transition $F1$ is shown in Figure 3.

After folding K and L into $F1$ there is no sequence component remaining. Therefore, we replace M and N by a transition $F2$ tagged with a `switch` expression. Note that this component is not mapped onto a `pick` construct because of the inscriptions

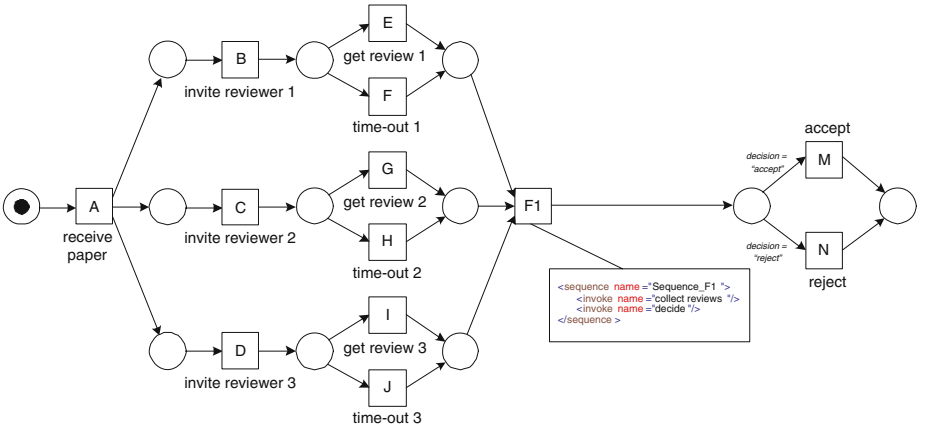


Fig. 3. The Petri net after replacing K and L by a transition $F1$ tagged with a `sequence` expression

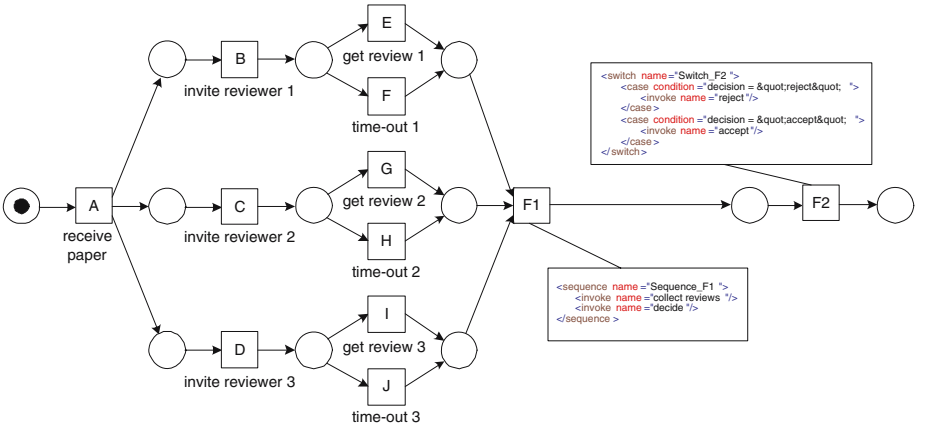


Fig. 4. The Petri net after replacing M and N by a transition $F2$ tagged with a `switch` expression

on the arcs suggesting some choice based on data rather than a time or message trigger. In our tool, we use a set of annotations to guide the generation of BPEL code. However, for the basic idea this is of less importance. Figure 4 shows the resulting WF-net.

Because of the introduction of $F2$ a new sequence is created. Clearly, this sequence is maximal and we can replace is by a transition $F3$ tagged with a `sequence` expression as shown in Figure 5.

In Figure 5 there are three components representing a `pick` component. Note that we assume that these represent a `pick` because there are no conditions on the arcs or the places with multiple outgoing arcs. As indicated, the WorkflowNet2BPEL4WS tool can handle a variety of tags directing the mapping process. In case of a `pick` it

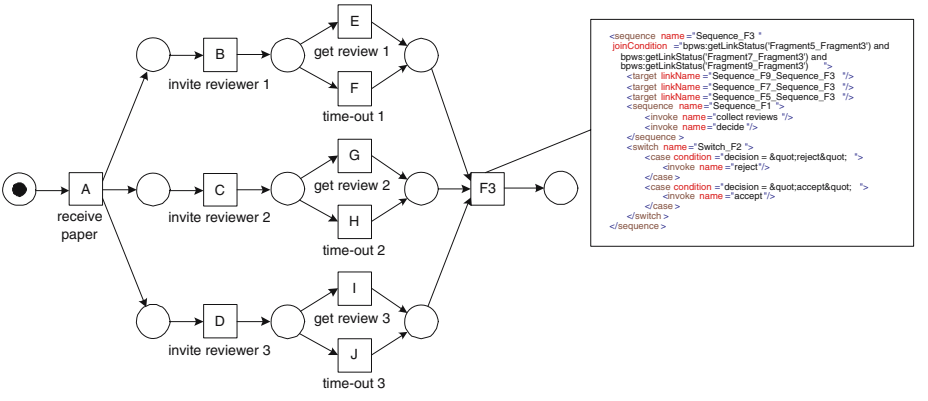


Fig. 5. The Petri net after replacing $F1$ and $F2$ by a transition $F3$ tagged with a `sequence` expression

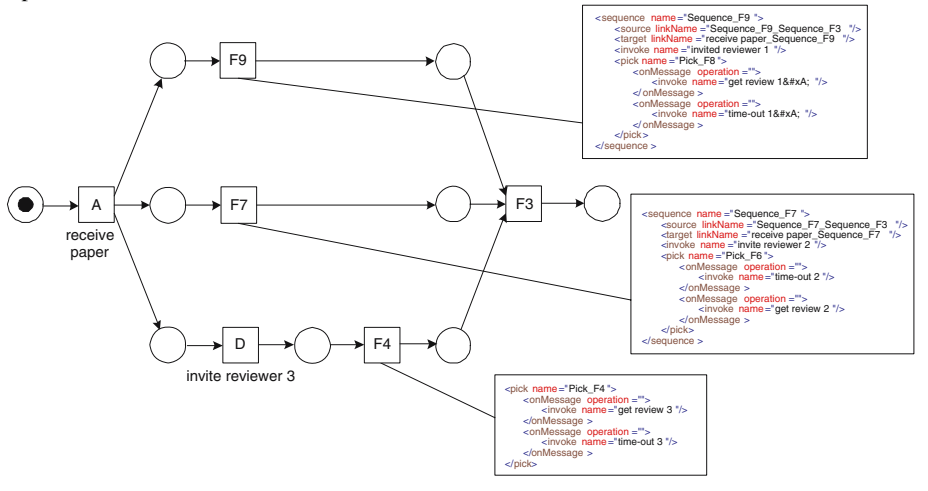


Fig. 6. The Petri net after replacing I and J by a transition $F4$ tagged with a `pick` expression. Then D and $F4$ can be merged into a sequence $F5$. Similarly, $F7$ and $F9$ can be created.

is possible to describe more about the nature of the choice (e.g., events, timers, etc.). However, in this paper we focus on the control-flow. Each of the `pick` components can be replaced by a single transition which is then merged with the preceding transition into a `sequence` transition.

Figure 6 illustrates the sequence of steps that are taken. Note that the intermediate result shown is actually not possible, i.e., first D and $F4$ would be merged into $F5$ before moving to the other two parallel branches. However, D and $F4$ are depicted separately to show the process in more intuitive manner. If D and $F4$ are also merged (i.e., the sequence of D and $F4$ is replaced by $F5$), there are 5 transitions remaining: A, $F3$, $F5$, $F7$, and $F9$. Together they form a `flow` component.

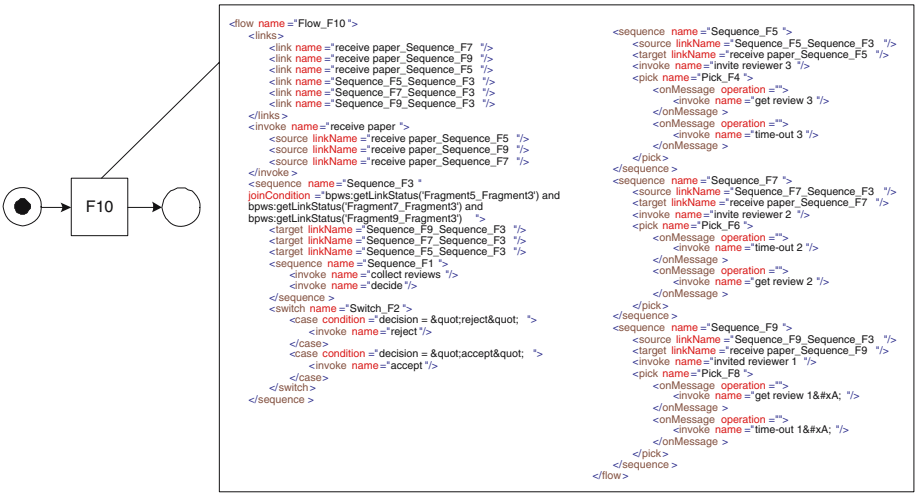


Fig. 7. The Petri net after replacing the remaining part by one transition $F10$ tagged with a `flow` expression

Figure 7 shows the result after applying the last step. Note that this is the 10th folding and the result is a WF-net consisting of only one transition $F10$. The annotation of $F10$ is the BPEL code for the entire process. Hence, we provided an iterative approach to translate the Wf-net shown Figure 2 into BPEL template code.

4 Implementation

For a detailed description of the algorithm we refer to [9]. In this section, we only highlight the basic structure of the tool. The starting point is a WF-net. We assume that this WF-net is modeled using CPN Tools [11,27]. Note that through ProM [12] and Woflan [41], it is possible to map (abstractions of) languages like Protos, Staffware, MQSeries Workflow, EPCs, YAWL, etc. onto WF-nets and export them to CPN Tools. In CPN Tools we assume some annotation describing e.g. the nature of choice (`pick` or `switch`) and the content of the activity represented by an atomic transition. We allow for the annotation of activity types like `invoke` (invoking an operation on some web service), `receive` (waiting for a message from an external source), `reply` (replying to an external source), `wait` (waiting for some time), `assign` (copying data from one place to another), `throw` (indicating errors in the execution), and `empty` (doing nothing).

In principle, no annotations are needed for the translation of Workflow nets in CPN Tools to BPEL. If a choice construct (place with outgoing arcs) is not annotated, it is assumed to be part of a `switch`. If an atomic transition is not annotated, it is assumed to be an `invoke` activity.

An important assumption for the correctness of our approach is that the initial WF-net is safe and sound. This can be checked with tools such as ProM [12] and Woflan [41] and also the state-space tool of CPN Tools [11].

In [9] we described that the various components can be detected in a WF-net. Based on this, WorkflowNet2BPEL4WS uses the following algorithm to produce BPEL code.

Definition 1 (Algorithm). *Let $PN = (P, T, F, \tau_P, \tau_G, \tau_{MA}, \tau_T)$ be a safe and sound annotated WF-net.*

(i) $X := PN$
(ii) *while* $[X] \neq \emptyset$ *(i.e., X contains a non-trivial component)*[2]
(iii-a) *If there is a maximal SEQUENCE component $C \in [X]$, select it and goto (vi).*
(iii-b) *If there is a SWITCH component $C \in [X]$, select it and goto (vi).*
(iii-c) *If there is a PICK component $C \in [X]$, select it and goto (vi).*
(iii-d) *If there is a WHILE component $C \in [X]$, select it and goto (vi).*
(iii-e) *If there is a maximal FLOW component $C \in [X]$, select it and goto (vi).*
(iv) *If there is a component $C \in [X]$ that appears in the component library, select it and goto (vi).*
(v) *Select a component $C \in [X]$ to be manually mapped onto BPEL and add it to the component library.*
(vi) *Attach the BPEL translation of C to t_C as illustrated in Figure 1.*
(vii) $X := fold(PN, C)$ *and return to (ii).*
(viii) *Output the BPEL code attached to the transition in X.*

The actual translation of components is done in step *(vi)* followed by the folding in step *(vii)*. The component to be translated/folded is selected in steps *(iii)*. If there is still a sequence remaining in the net, this is selected. A maximal sequence is selected to keep the translation as compact and simple as possible. Only if there are no sequences left in the WF-net, other components are considered. The next one in line is the SWITCH component followed by the PICK component and the WHILE component. Given the fact that SWITCH, PICK and WHILE components are disjoint, the order of steps *(iii-b)*, *(iii-c)*, and *(iii-d)* is irrelevant. Finally, maximal FLOW components are considered.

Not every net can be reduced into SEQUENCE, SWITCH, PICK, WHILE, and FLOW components. Therefore, steps *(iv)* and *(v)* have been added. The basic idea is to allow for ad-hoc translations. These translations are stored in a component library. If the WF-net cannot be reduced any further using the standard SEQUENCE, SWITCH, PICK, WHILE and FLOW components, then the algorithm searches the component library (note that it only has to consider the network structure and not the specific names and annotations). If the search is successful, the stored BPEL mapping can be applied (modulo renaming of nodes and arc and transition annotations). If there is not a matching component, the tool will save the irreducible net along with each of the components in the net. A manual translation can be then provided for one of the components and stored in the component library for future use.

The component library is composed of pairs of component and the translation of it into BPEL activity. When we match a component against a library component we take each pair of transitions that where matched successfully by that component and substitute the BPEL activity in the library BPEL specification code by that of the transition in the net that is being translated. So if a transition is annotated with a sequence and the

[2] Note that this is the case as long as X is not reduced to a WF-net with just a single transition.

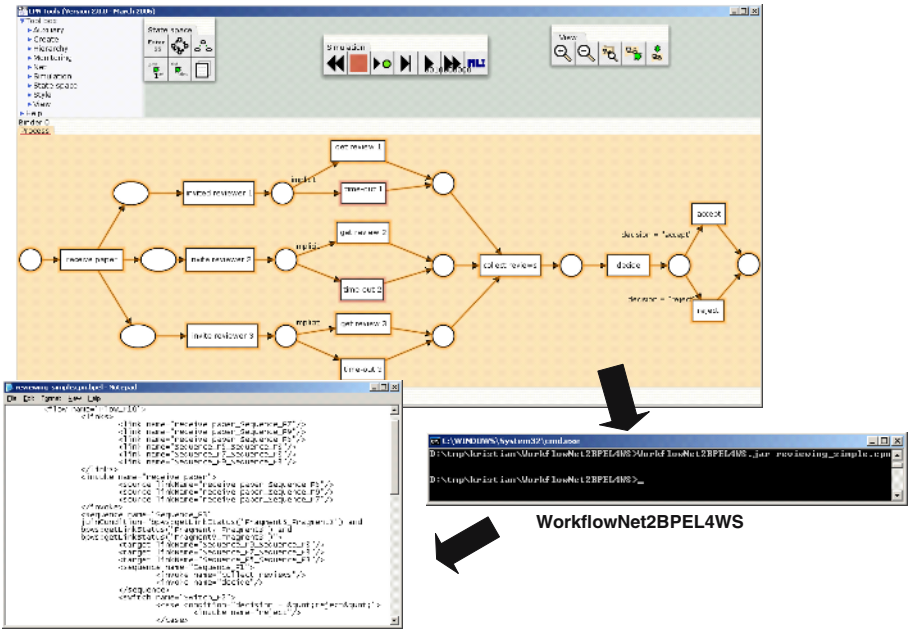


Fig. 8. A screenshot of CPN Tools and the resulting BPEL code after applying the WorkflowNet2BPEL4WS tool

corresponding transition in the library component is an invoke then the invoke in the library BPEL invoke is replaced by the sequence. If we did not do this, then each time a library component is used for reduction, all previous translations would get lost.

Figure 8 shows a screenshot of WorkflowNet2BPEL4WS in action. The WF-net Figure 2 is loaded into CPN Tools and WorkflowNet2BPEL4WS iteratively creates BPEL template code. Note that WorkflowNet2BPEL4WS saves the result of each step, i.e., for Figure 2 there are 9 intermediate and one final model generated. This allows the designer to check the translation process. If the net is irreducible, i.e. there is no predefined or library component that can match a component in the net, then WorkflowNet2BPEL4WS stores the irreducible net along with a copy of the components that exists in the irreducible net. This makes it easy to develop the library to cope with a particular Workflow net, by choosing a component in the list that WorkflowNet2BPEL provides and translating it, and adding the component and the translation of it to the library.

The default component matching order in the tool is the one described in [9], but it is possible to change the order in which components are matched. It is possible express that FLOW components are selected before SEQUENCE components, or that a user-defined component has priority over FLOW components. By doing this, the user of the tool can adjust the "style" of the generated BPEL and not just settle with some fixed order.

5 Evaluation

To evaluate our approach and to test the WorkflowNet2BPEL4WS tool, we used 100 process models developed by students in group projects. Each group modeled the processes related to a realistic business case using Protos. Each group was free to select their own business case. Using interviews and documentation, their assignment was to model the business processes, to analyze them, and to propose redesigns. It is important to stress that the processes were not selected or modeled by the authors of this paper, i.e., the groups were free to choose a business case. There were only requirements with respect to the size and complexity of the models. Moreover, the models had to be correct. Using Woflan the groups could verify the soundness of their models [41]. As a modeling tool they used Protos. The reason is that this is the most widely used business process modeling tool in the Netherlands (see Section 1.4). Moreover, we have developed interfaces to Petri-net-based analysis tools such as Woflan, ProM, ExSpect, and CPN Tools.

Although there were more than Protos 100 models, we made an random selection of 100 models. All these models where exported to our input format by, first importing them into ProM and then exporting them to the CPN Tools format. On average the generated CPN models contained 24 places and 27 transitions.

The goal of the evaluation was to take the WorkflowNet2BPEL4WS tool and convert each of the 100 Protos models into BPEL. This was not trivial since only 11 of the 100 models could be reduced using the standard predefined components of the algorithm, i.e., just 11 models could be completely reduced into SEQUENCE, SWITCH, PICK, WHILE, and FLOW components. Therefore, we were forced to add components to our component library. We started with an empty component library and each time we encountered a WF-net that could not be reduced completely, we added a manual translation of the smallest irreducible component to our library. After adding 76 components to the library we succeeded in completely reducing all Protos models.

In Figure 9(a) we see the relationship between components added to the library and the number of nets reduced. It starts out in (0,11) (11 nets could be reduced using the predefined components) and ends up in (100,76) (76 components had to be added to reduce all 100 nets).

For each library component we added, we translated all of the 100 nets to check the distribution of components types. Figure 9(b) shows how often each component type could be applied to reduce the WF-nets in each of the 76 steps. For example, the number of times a component could be reduced by mapping it onto a SEQUENCE increases from less than 700 until 743 (top line). This number increases because each time a component is added to the library a further reduction is possible and new SEQUENCE components may surface. The line “x” in Figure 9(b) shows the number of reductions possible because of the component library. Initially, this value is 0 because the component library is empty. After adding 76 ad-hoc components, 132 reductions result from the component library. After adding these 76 each WF-net can be reduced completely and as a result we obtain the BPEL code for all 100 models.

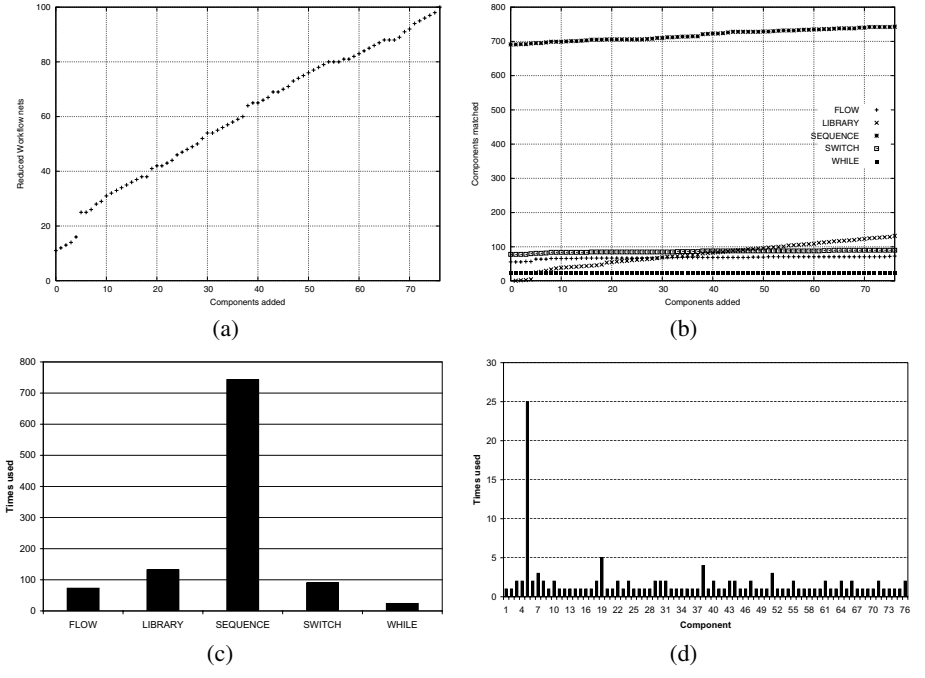


Fig. 9. Graphs displaying the performance of the algorithm

Figure 9(c) shows the final distribution of reductions, i.e., while generating the BPEL code 73 FLOW components, 132 LIBRARY components, 743 SEQUENCE components, 90 SWITCH components, and 24 WHILE components are encountered.[3] These numbers show that the standard reductions using the SEQUENCE, SWITCH, PICK, WHILE, and FLOW components are doing remarkably well. On the total of 1062 reductions required to generate the BPEL code, only 132 were due to LIBRARY components ($\simeq$12.4%).

After adding all 76 components to the library we checked to see how often a library component was used when translating all of the 100 WF-nets. This is shown in Figure 9(d). The figure shows that most of the library components we used only once. Only one library component (number 5) was used frequently, i.e., of the 76 library components 50 components were used only once ($\simeq$ 66%), 21 were used twice ($\simeq$ 28%), 2 were used three times ($\simeq$ 3%), 1 was used four times ($\simeq$ 1%), 1 was used five times ($\simeq$ 1%), and 1 was used 25 times ($\simeq$ 1%).

[3] The reason that no PICK components where found was that the current export facility of ProM to CPN Tools do not annotate the WF-nets with this information. Since the only difference between a match for a PICK and a SWITCH component is the annotation of the place splitting the flow, the reduction process is not influenced. If we would have transferred this information from Protos, the sum of the number of SWITCH and PICK reductions would be 90 and the other numbers would not be affected. Note that this information is available in Protos but cannot (yet) be stored in the intermediate PNML format.

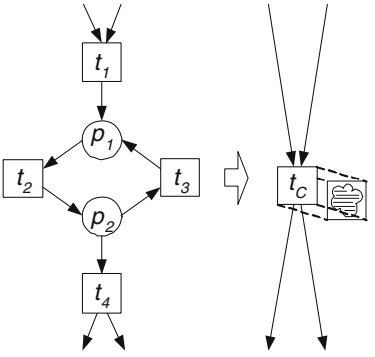


Fig. 10. The reduction using the library component that was used 25 times

Figure 10 shows the library component that could be applied most frequently. Clearly, this is a good candidate to be added to the set of standard components. None of the other library components is used very frequently. This shows that some manual work will always be necessary. We would like to point out that students were encouraged to select complicated business processes, i.e., part of the grading was based on the use of as many workflow patterns as possible [7]. As a result we expect the selected set of Protos models to be more complicated than usual. For example, the students were encouraged to use the "Milestone Pattern" [7] which cannot be reduced using any of the standard components and is difficult to capture this pattern in a single component to be added to the library. Therefore, the results should be interpreted as a worst-case scenario. We expect that in a most situations, more than 95% of the components can be reduced using the standard components and the component depicted in Figure 10.

The performance of the reductions and BPEL generation is not an issue. After adding the 76 library components, each of the 100 Workflow nets was translated within a few seconds.

6 Conclusion and Future Work

In this paper we presented an approach to generate BPEL code from WF-nets using reductions, i.e., components are folded into transitions labeled with BPEL code. The main goal is to generate *readable* BPEL code. Therefore, we did not aim at a complicated full translation (e.g., using events handler [34]) but at recognizing "natural BPEL constructs". The approach is supported by the translation tool *WorkflowNet2BPEL4WS*. The tool also supports an extensible component library, i.e., components and their preferred BPEL translations can be added thus allowing for different translation styles.

We have evaluated WorkflowNet2BPEL4WS and the underlying ideas using 100 complex Protos models. Based on this evaluation we estimate that more than 95% of real-life process models can be automatically translated into readable BPEL code. However, some manual interventions are needed to translate the remaining 5%. As demonstrated, larger component libraries could be used to achieve a fully automatic translation in most cases.

The current implementation can be used in conjunction with a wide variety of Petri net based tools, e.g., CPN Tools, ProM, Yasper, WoPeD, PNK, CPN-AMI, and Protos. Moreover, the ideas are also applicable to other graph-based languages such as BPMN, UML activity diagrams, EPCs , and proprietary workflow languages.

In the near future we plan to implement this algorithm directly into ProM [12] so it will be possible to translate a wide variety of process modeling languages to BPEL. Moreover, Pallas Athena is interested in integrating WorkflowNet2BPEL4WS into Protos. Give the widespread use of Protos, such an implementation would allow many organizations to generate BPEL code.

References

1. W.M.P. van der Aalst. Verification of Workflow Nets. In P. Azéma and G. Balbo, editors, *Application and Theory of Petri Nets 1997*, volume 1248 of *Lecture Notes in Computer Science*, pages 407–426. Springer-Verlag, Berlin, 1997.
2. W.M.P. van der Aalst. The Application of Petri Nets to Workflow Management. *The Journal of Circuits, Systems and Computers*, 8(1):21–66, 1998.
3. W.M.P. van der Aalst. Workflow Verification: Finding Control-Flow Errors using Petri-net-based Techniques. In W.M.P. van der Aalst, J. Desel, and A. Oberweis, editors, *Business Process Management: Models, Techniques, and Empirical Studies*, volume 1806 of *Lecture Notes in Computer Science*, pages 161–183. Springer-Verlag, Berlin, 2000.
4. W.M.P. van der Aalst. Business Process Management Demystified: A Tutorial on Models, Systems and Standards for Workflow Management. In J. Desel, W. Reisig, and G. Rozenberg, editors, *Lectures on Concurrency and Petri Nets*, volume 3098 of *Lecture Notes in Computer Science*, pages 1–65. Springer-Verlag, Berlin, 2004.
5. W.M.P. van der Aalst, M. Dumas, and A.H.M. ter Hofstede. Web Service Composition Languages: Old Wine in New Bottles? In G. Chroust and C. Hofer, editors, *Proceeding of the 29th EUROMICRO Conference: New Waves in System Architecture*, pages 298–305. IEEE Computer Society, Los Alamitos, CA, 2003.
6. W.M.P. van der Aalst and K.M. van Hee. *Workflow Management: Models, Methods, and Systems*. MIT press, Cambridge, MA, 2002.
7. W.M.P. van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski, and A.P. Barros. Workflow Patterns. *Distributed and Parallel Databases*, 14(1):5–51, 2003.
8. W.M.P. van der Aalst, J.B. Jørgensen, and K.B. Lassen. Let's Go All the Way: From Requirements via Colored Workflow Nets to a BPEL Implementation of a New Bank System Paper. In R. Meersman and Z. Tari et al., editors, *On the Move to Meaningful Internet Systems 2005: CoopIS, DOA, and ODBASE: OTM Confederated International Conferences, CoopIS, DOA, and ODBASE 2005*, volume 3760 of *Lecture Notes in Computer Science*, pages 22–39. Springer-Verlag, Berlin, 2005.
9. W.M.P. van der Aalst and K.B. Lassen. Translating Workflow Nets to BPEL4WS. BETA Working Paper Series, WP 145, Eindhoven University of Technology, Eindhoven, 2005.
10. T. Andrews, F. Curbera, H. Dholakia, Y. Goland, J. Klein, F. Leymann, K. Liu, D. Roller, D. Smith, S. Thatte, I. Trickovic, and S. Weerawarana. Business Process Execution Language for Web Services, Version 1.1. Standards proposal by BEA Systems, International Business Machines Corporation, and Microsoft Corporation, 2003.
11. CPN Group, University of Aarhus, Denmark. CPN Tools Home Page. http://wiki.daimi.au.dk/cpntools/.

12. B. van Dongen, A.K. Alves de Medeiros, H.M.W. Verbeek, A.J.M.M. Weijters, and W.M.P. van der Aalst. The ProM framework: A New Era in Process Mining Tool Support. In G. Ciardo and P. Darondeau, editors, *Application and Theory of Petri Nets 2005*, volume 3536 of *Lecture Notes in Computer Science*, pages 444–454. Springer-Verlag, Berlin, 2005.
13. M. Dumas, W.M.P. van der Aalst, and A.H.M. ter Hofstede. *Process-Aware Information Systems: Bridging People and Software through Process Technology*. Wiley & Sons, 2005.
14. D. Fahland and W. Reisig. ASM-based semantics for BPEL: The negative control flow. In D. Beauquier and E. Börger and A. Slissenko, editor, *Proc. 12th International Workshop on Abstract State Machines*, pages 131–151, Paris, France, March 2005.
15. R. Farahbod, U. Glässer, and M. Vajihollahi. Specification and validation of the business process execution language for web services. In W. Zimmermann and B. Thalheim, editors, *Abstract State Machines 2004*, volume 3052 of *Lecture Notes in Computer Science*, pages 79–94, Lutherstadt Wittenberg, Germany, May 2004. Springer-Verlag, Berlin.
16. A. Ferrara. Web services: A process algebra approach. In *Proceedings of the 2nd international conference on Service oriented computing*, pages 242–251, New York, NY, USA, 2004. ACM Press.
17. L. Fischer, editor. *Workflow Handbook 2003, Workflow Management Coalition*. Future Strategies, Lighthouse Point, Florida, 2003.
18. J.A. Fisteus, L.S. Fernández, and C.D. Kloos. Formal verification of BPEL4WS business collaborations. In K. Bauknecht, M. Bichler, and B. Proll, editors, *Proceedings of the 5th International Conference on Electronic Commerce and Web Technologies (EC-Web '04)*, volume 3182 of *Lecture Notes in Computer Science*, pages 79–94, Zaragoza, Spain, August 2004. Springer-Verlag, Berlin.
19. X. Fu, T. Bultan, and J. Su. Analysis of Interacting BPEL Web Services. In *International World Wide Web Conference: Proceedings of the 13th international conference on World Wide Web*, pages 621–630, New York, NY, USA, 2004. ACM Press.
20. D. Georgakopoulos, M. Hornick, and A. Sheth. An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure. *Distributed and Parallel Databases*, 3:119–153, 1995.
21. Object Management Group. *OMG Unified Modeling Language 2.0*. OMG, http://www.omg.com/uml/, 2005.
22. IBM WebSphere. www-306.ibm.com/software/websphere.
23. S. Jablonski and C. Bussler. *Workflow Management: Modeling Concepts, Architecture, and Implementation*. International Thomson Computer Press, London, UK, 1996.
24. G. Keller, M. Nüttgens, and A.W. Scheer. Semantische Processmodellierung auf der Grundlage Ereignisgesteuerter Processketten (EPK). Veröffentlichungen des Instituts für Wirtschaftsinformatik, Heft 89 (in German), University of Saarland, Saarbrücken, 1992.
25. J. Koehler and R. Hauser. Untangling Unstructured Cyclic Flows A Solution Based on Continuations. In R. Meersman, Z. Tari, W.M.P. van der Aalst, C. Bussler, and A. Gal et al., editors, *On the Move to Meaningful Internet Systems 2004: CoopIS, DOA, and ODBASE: OTM Confederated International Conferences, CoopIS, DOA, and ODBASE 2004*, volume 3290 of *Lecture Notes in Computer Science*, pages 121–138, 2004.
26. M. Koshkina and F. van Breugel. Verification of Business Processes for Web Services. Technical report CS-2003-11, York University, October 2003. Available from: http://www.cs.yorku.ca/techreports/2003/.
27. L.M. Kristensen, S. Christensen, and K. Jensen. The Practitioner's Guide to Coloured Petri Nets. *International Journal on Software Tools for Technology Transfer*, 2(2):98–132, 1998.
28. F. Leymann. Web Services Flow Language, Version 1.0, 2001.
29. F. Leymann and D. Roller. *Production Workflow: Concepts and Techniques*. Prentice-Hall PTR, Upper Saddle River, New Jersey, USA, 1999.

30. A. Martens. Analyzing Web Service Based Business Processes. In M. Cerioli, editor, *Proceedings of the 8th International Conference on Fundamental Approaches to Software Engineering (FASE 2005)*, volume 3442 of *Lecture Notes in Computer Science*, pages 19–33. Springer-Verlag, Berlin, 2005.
31. M. zur Muehlen. *Workflow-based Process Controlling: Foundation, Design and Application of workflow-driven Process Information Systems*. Logos, Berlin, 2004.
32. Oracle BPEL Process Manager. www.oracle.com/technology/products/ias/bpel.
33. C. Ouyang, W.M.P. van der Aalst, S. Breutel, M. Dumas, , and H.M.W. Verbeek. Formal Semantics and Analysis of Control Flow in WS-BPEL. BPM Center Report BPM-05-15, BPMcenter.org, 2005.
34. C. Ouyang, M. Dumas, S. Breutel, and A.H.M. ter Hofstede. Translating Standard Process Models to BPEL. BPM Center Report BPM-05-27, BPMcenter.org, 2005.
35. C. Ouyang, W.M.P. van der Aalst, M. Dumas, and A.H.M. ter Hofstede. Translating BPMN to BPEL. BPM Center Report BPM-06-02, BPMcenter.org, 2006.
36. Pallas Athena. *Protos User Manual*. Pallas Athena BV, Plasmolen, The Netherlands, 2004.
37. A.W. Scheer. *ARIS: Business Process Modelling*. Springer-Verlag, Berlin, 2000.
38. C. Stahl. Transformation von BPEL4WS in Petrinetze (In German). Master's thesis, Humboldt University, Berlin, Germany, 2004.
39. S. Thatte. XLANG Web Services for Business Process Design, 2001.
40. H.M.W. Verbeek and W.M.P. van der Aalst. Analyzing BPEL Processes using Petri Nets. In D. Marinescu, editor, *Proceedings of the Second International Workshop on Applications of Petri Nets to Coordination, Workflow and Business Process Management*, pages 59–78. Florida International University, Miami, Florida, USA, 2005.
41. H.M.W. Verbeek, T. Basten, and W.M.P. van der Aalst. Diagnosing Workflow Processes using Woflan. *The Computer Journal*, 44(4):246–279, 2001.
42. WFMC. Workflow Management Coalition Workflow Standard: Workflow Process Definition Interface – XML Process Definition Language (XPDL) (WFMC-TC-1025). Technical report, Workflow Management Coalition, Lighthouse Point, Florida, USA, 2002.
43. S. White. Using BPMN to Model a BPEL Process. *BPTrends*, 3(3):1–18, March 2005.
44. S.A. White et al. Business Process Modeling Notation (BPML), Version 1.0, 2004.
45. P. Wohed, W.M.P. van der Aalst, M. Dumas, and A.H.M. ter Hofstede. Analysis of Web Services Composition Languages: The Case of BPEL4WS. In I.Y. Song, S.W. Liddle, T.W. Ling, and P. Scheuermann, editors, *22nd International Conference on Conceptual Modeling (ER 2003)*, volume 2813 of *Lecture Notes in Computer Science*, pages 200–215. Springer-Verlag, Berlin, 2003.

Let's Dance: A Language for Service Behavior Modeling

Johannes Maria Zaha[1], Alistair Barros[2], Marlon Dumas[1], and Arthur ter Hofstede[1]

[1] Queensland University of Technology, Brisbane, Australia
{j.zaha, m.dumas, a.terhofstede}@qut.edu.au
[2] SAP Research Centre, Brisbane, Australia
alistair.barros@sap.com

Abstract. In Service-Oriented Architectures (SOAs), software systems are decomposed into independent units, namely services, that interact with one another through message exchanges. To promote reuse and evolvability, these interactions are explicitly described right from the early phases of the development lifecycle. Up to now, emphasis has been placed on capturing structural aspects of service interactions. Gradually though, the description of behavioral dependencies between service interactions is gaining increasing attention as a means to push forward the SOA vision. This paper deals with the description of these behavioral dependencies during the analysis and design phases. The paper outlines a set of requirements that a language for modeling service interactions at this level should fulfill, and proposes a language whose design is driven by these requirements.

1 Introduction

As the first generation of web service technology based on XML, SOAP, and WSDL, reaches a certain level of maturity and adoption, a second generation based on richer service descriptions is gestating. Whereas in first-generation web services, service descriptions are usually equated to sets of operations and message types, in the second generation the description of behavioral dependencies between service interactions (e.g. the order in which messages must be exchanged) plays a central role.

Two standardization initiatives, BPEL [1] and WS-CDL [3], have promoted the description of behavioral aspects of service interactions. However, these proposals focus on the implementation phase of the service development lifecycle. Indeed, although WS-CDL claims to be at a higher level than BPEL, they both rely on procedural programming constructs such as variable assignment and instruction sequencing. One can argue that during analysis and design, capturing constraints on possible interactions is more relevant than prescribing interaction procedures. Furthermore, while BPEL focuses on describing interactions from a local perspective, i.e. from the perspective of a specific service, WS-CDL emphasizes global descriptions (also called choreographies).

R. Meersman, Z. Tari et al. (Eds.): OTM 2006, LNCS 4275, pp. 145–162, 2006.

Previous proposals have defined extensions of well-known behavior modeling paradigms (e.g. Activity Diagrams) to capture message exchanges in the style of service interactions. This is the case for example of BPMN [18] and BPSS (or ebBP) [5]. However, these languages do not treat service interactions as first-class citizens but rather as extensions to a core language centered around the notion of actions and dependencies between actions. In particular, service interactions in these languages are not composable, meaning that it is not possible to aggregate several interactions and treat them as a single interaction to which it is possible to apply the same operators as for elementary interactions. It is thus questionable whether these languages meet the requirements of a language for service interaction modeling. Also, these languages do not provide a unified framework for capturing interactions both from a local and from a global viewpoint.

In prior work, we have captured suitability (i.e. fit-to-purpose) requirements for service interaction languages and documented them in the form of 13 patterns [2]. In this paper, we formulate additional requirements for a service interaction modeling language and use these as the basis for a language proposal. The main contribution of the paper is thus a language, namely "Let's Dance", for modeling behavioral dependencies between service interactions. The language is targeted at business analysts and software architects involved in the initial phases of the service development lifecycle. Accordingly, it abstracts away from implementation details and avoids reliance on imperative programming constructs. At the same time, models defined in the proposed language contain information that can be leveraged upon during the implementation and operations phases, for example to generate BPEL templates or to ensure, through monitoring or log analysis, that a given service implementation conforms to the requirements expressed in the models.

The paper is structured as follows. In Section 2 the requirements for a service behavior modeling language are formulated. Section 3 gives an overview of the Let's Dance language. In Section 4 the suitability of the language is illustrated through scenarios corresponding to some of the interaction patterns of [2]. (An extended version of this paper provides a full suitability evaluation against all the patterns [19]). Section 5 introduces the meta model of the language as well as its informal semantics. Finally, Section 6 concludes and gives an outlook on future work.

2 Requirements and Related Work

The intended scope of the Let's Dance language is to capture models of service interactions from a behavioral perspective. The language is targeted at the analysis and design phases of the systems development lifecycle. As such, it should be sufficiently *abstract* (i.e. *conceptual*), meaning that it should allow modelers to focus on the essence of their service interaction analysis and design problems, abstracting away from implementation details. Accordingly, we aim at defining a language that is as free as possible from programming constructs such as explicit

variable assignment. Examples of what the language should not force modelers to do include:

- Having to intersperse variable assignment actions for book-keeping purposes, e.g. using variables as counters or as "flags" to indicate that a state has been reached, or as "buffers" to store the result of intermediate computations.
- Having to introduce unnecessary ordering or synchronization constraints, i.e. sequentializing interactions that could otherwise be performed in any order or in parallel.

A major use case of the language is to define models of service interactions that can be refined into implementations, or compared against the behavior exhibited by an existing implementation. Thus, the language should have an unambiguous semantics and it should be possible to compare the semantics of a model with the behavior exhibited by an implementation. In addition, it is desirable to reason about the models defined in the language, e.g. to verify or test certain properties through static analysis or simulation. One way to achieve these requirements is to endow the language with a *formal* and *executable* semantics.

Another major purpose of the language is to facilitate communication between analysts, designers, and other stakeholders involved in the development of service-oriented systems. Users of the language, including non-technical users, will need to understand, critique, modify or suggest modifications to models or parts thereof. The language must hence be *comprehensible*. This comprehensibility can be achieved by: (i) attaching a graphical syntax to the language to better exploit the perceptual capabilities of users, and (ii) allowing models to be captured at different levels of detail and from different viewpoints. In particular, the language should allow interactions to be decomposed into other interactions so that users can choose the level of details at which they wish to view an interaction model. Also, users should be able to design and view models from a local and from a global perspective. In the global (or choreography) perspective, interactions are described from the perspective of a collection of services (abstracted as roles). This is useful when communicating about how services should behave in order to seamlessly interact with one another. On the other hand, local models focus on the perspective of either an existing or a "to be" service, capturing only those interactions that involve that particular service. A possible usage scenario is one where global models are produced by analysts to agree on major interaction scenarios, while local models are produced during system design and handed on to implementers. To ensure proper handovers between these users, it is necessary to have a mapping from global to local models and/or to be able to check that a local model is consistent with a global one. In other scenarios, reverse mappings from local to global models may also be useful.

Finally, the language must be *expressive* and *suitable*. Given the intended scope, expressiveness refers to the ability to capture any set of service interactions and their associated behavioral dependencies. Suitability on the other hand, refers to the ability to capture common scenarios in an intuitive manner. Languages such as Colored Petri nets [12] or Live Sequence Charts [6] allow one to capture any service interaction scenario (in computational terms, they

are Turing complete). However, they are not necessarily suitable for the task at hand: modeling certain common interaction scenarios would require the use of programming constructs. So while expressiveness is certainly important and we plan to investigate this in the future, our initial language design has been driven by suitability. Suitability is arguably a subjective property and is ultimately determined by the users and the use cases. We have chosen to take the service interaction patterns proposed in [2] as a way of evaluating suitability. These patterns have been derived from insights into real-scale B2B transaction processing, use cases gathered by standardization committees (e.g. BPEL and WS-CDL) during their requirements analysis phase, and scenarios identified in industry standards (e.g. xCBL choreographies and RosettaNet PIPs [17]). The proposed patterns, as such, are not complete but aim at consolidating recurrent scenarios, abstracting them in a way that provides reusable knowledge to service developers. Since the patterns are abstractions of recurrent scenarios, they can be used to assess the suitability of a given language for service behavior description. Accordingly, a driving requirement of our language design is that it should provide direct support for these patterns. Section 4 and [19] shows how these patterns are captured in Let's Dance.

Existing languages for capturing service interaction behavior include WS-CDL [3], BPEL [1], BPMN [18], UML Activity Diagrams [15] (and variants thereof as defined in related initiatives such as BPSS [5] or EDOC [14]). WS-CDL and BPEL are aimed at the detailed design and implementation stages of the development lifecycle. They both extensively rely on programming language constructs such as sequence, "repeat until" and "while" loops, and variable assignment. Capturing service interaction patterns related to multi-party or streamed interactions in these languages requires extensive use of variables for record-keeping purposes.[1] Also, BPEL does not allow one to represent global views. Moreover, these languages do not have a graphical syntax, although different tools assign them various graphical representations. UML activity diagrams and BPMN do have a prescribed graphical representations and are targeted at analysts and designers. However, in these languages interactions are not treated as first-class citizens. Elementary interactions (i.e. message exchanges) can be represented through "message flows" and "object flows" respectively. However, these constructs are not composable: It is not possible to represent an interaction that is composed of other interactions. And as in the case of BPEL and WS-CDL, UML and BPMN do not provide constructs to capture multi-party and streamed interactions and these would need to be encoded using loops and variable manipulation for record-keeping.

Another family of languages that have been proposed for capturing service interactions are the so-called "semantic web service" description languages. This family of languages includes OWL-S [13] and WSMO [16]. The basic idea of these languages is to use logical statements to capture and manipulate service-related information. In these languages, a service is described as a set of facts and rules covering three broad aspects:

[1] See sample code available at www.serviceinteraction.com.

- Capability: what can the service do?
- Non-functional properties: conditions of usage, contractual terms, contextual constraints, etc.
- Interface: preconditions, post-conditions and/or effects of each interaction in which the service can engage. Interface descriptions are akin to local models in Let's Dance, although they cover both structural and behavioral aspects.

While semantic web service descriptions are suitable in view of applying automated reasoning techniques (e.g. automated planning) over service descriptions, their suitability for use at the level of domain analysis and systems design is questionable. Domain analysts do not typically describe services down to the level of details required for non-trivial automated reasoning. Thus, semantic web service description languages can be seen as possible target languages for Let's Dance. For example, Let's Dance local models could be used to generate templates of WSMO descriptions, which after refinement, could be given to automated reasoning engines to determine if a service can be combined with other services to achieve a given goal.

In [4], Finite State Machines (FSMs) are put forward as a suitable language for modeling service interactions. However, while state machines lead to simple models for highly sequential scenarios, they may lead to complex, spaghetti-like models when used to capture scenarios with parallelism and cancellation (e.g. scenarios where a given interaction may occur at any time during the execution of another set of interactions).

Foster et al [9] suggest the use of Message Sequence Charts (MSCs) for describing global service interaction models. These global models are converted into local models expressed as FSMs for analysis purposes. It should be noted that MSCs are a notation for describing behavior scenarios as opposed to full behavior specifications. In particular, basic MSCs do not allow one to capture conditional branches, parallel branches, and iterations. Extensions to MSCs to capture complex behavior have been defined, but in realistic cases they lead to cluttered diagrams since MSCs are based on lifelines which are fundamentally targeted at capturing sequencing rather than branching.

3 Language Overview

We have used the cognitive dimensions for visual programming environments introduced in [10] as a guideline for optimizing the choice of constructs for the visual syntax of Let's Dance. For the design of a graphical modeling language, important dimensions to consider, among those presented in [10], are abstraction gradient, consistency and diffuseness. The abstraction gradient covers the minimum and maximum levels of abstraction that must be captured and the possibility to encapsulate fragments of a certain model. In a language for service behavior modeling there are two levels of abstraction: the global model is the higher level while the local model is the lower one. Closer inspection shows that these levels can themselves be described at different levels of abstraction and this can be achieved by ensuring composability. The consistency or harmony of

a language testifies how much of a language has to be learnt by a user in order to infer the rest. This inference should be easy for a language fulfilling the above requirements, since the same constructs ought to be used for describing local and global models. Finally, the dimension of diffuseness reflects how many symbols or graphic entities are required to express a given issue. This dimension is determined by the need to support local and global models and by the concepts involved in each of these models.

Service interactions can be described in terms of message exchanges. A message exchange consists of a message sending and a message receipt. Thus, at the lowest level of abstraction, a language for modeling service behavior must provide these two constructs. To optimize the abstraction gradient and consistency dimensions discussed above, we choose a notation wherein the visual juxtaposition of the symbols for sending and receiving messages leads to the symbol for elementary interactions (i.e. message exchanges). Interactions can then be related and composed to form choreographies. Communication actions are represented by the non-regular pentagons shown in the left-hand side of Figure 1. As illustrated in this figure, we distinguish between a message sending that requires an acknowledgment and one that does not. Similarly, we distinguish between message receipt actions that provide acknowledgment and those that do not.[2]

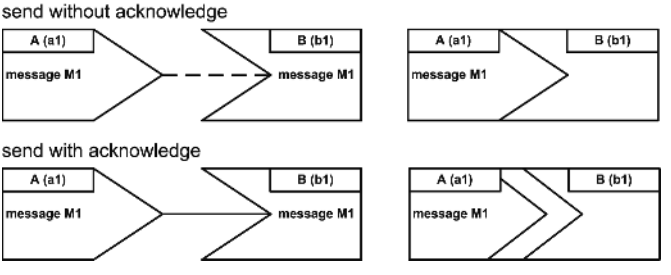


Fig. 1. Communication Actions and Interactions

A communication action is performed by an actor playing a role. This information is specified at the top corner of the pentagon denoting a communication action. Roles are written in uppercase and the actor playing this role (specifically, the "actor reference") is written in lowercase between brackets. The symbol for sending and receiving messages are combined to form elementary interactions by juxtaposing the respective symbols (see right-hand side of Figure 1). Again, we distinguish between elementary interactions with and without acknowledgment.

Interactions can be inter-related using the constructs depicted in Figure 2. The relationship on the left-hand side is called "precedes" and is depicted by a directed edge: the source interaction can only occur after the target interaction

[2] At the modeling level, we are only concerned with capturing whether an acknowledgment is needed or not. The protocol used for acknowledging is an implementation concern.

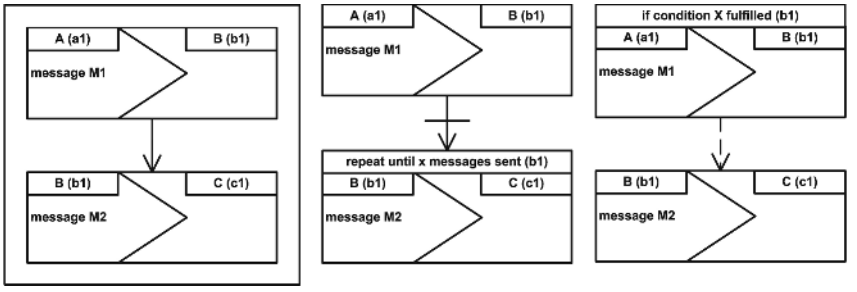


Fig. 2. Relationships between interactions

has occurred. That is, after the receipt of a message "M1" by "B", "B" is able to send a message "M2" to "C". The rectangle surrounding these two interactions denotes a composite interaction, which can be related with other interactions with any type of relationship. The sub-interactions of a composite interaction may, but need not be related. If there is no relationship between them, they can occur in any order or in parallel.

The relationship at the center of the figure is called "inhibits", depicted by a crossed directed edge. It denotes that after the source interaction has occurred, the target interaction can no longer occur. That is, after "B" has received a message "M1" from "A", it may not send a message "M2" to "C". The latter interaction can be repeated until "x" messages have been sent, which is indicated by the header on top of the interaction. The actor executing the repetition instruction is noted in brackets. Additional to the "Until" construct, Let's Dance provides two more constructs to denote the repetition of an interaction, namely "For each" and "While".

Finally, the relationship on the right-hand side of the figure, called "weak-precedes", denotes that "B" is not able to send a message "M2" until "A" has sent a message "M1" or until this interaction has been inhibited. That is, the target interaction can only occur after the source interaction has reached a final status, which may be "completed" or "skipped" (i.e. "inhibited"). In the example, the upper interaction has a guard assigned, which is denoted by the header on top of the interaction. Guards allow for the definition under which condition a given interaction can be executed and can be assigned to any interaction, whereby the actor evaluating a guard has to be named explicitly.

All these constructs will be exemplified in the following section, where we evaluate the suitability of Let's Dance with respect to the interaction patterns of [2].

4 Interaction Patterns in Let's Dance

The service interaction patterns introduced in [2] have been put forward as an instrument for benchmarking languages for service behavior modeling. In [2], solutions to the patterns in BPEL are sketched (the full solutions can be found in

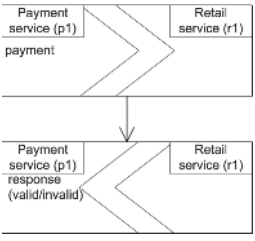


Fig. 3. Send/Receive (Pattern 3)

www.serviceinteraction.com). These patterns are divided in four categories: single-transmission bilateral interaction patterns, single-transmission multilateral interaction patterns, multi-transmission interaction patterns and routing patterns.

We have assessed the suitability of Let's Dance by modeling sample scenarios corresponding to all 13 patterns. Due to space limitations, in this paper we only consider representative patterns of each of the four categories. A discussion on how Let's Dance addresses the remaining patterns is given in [19]. Using the nomenclature and numbering of [2] we have chosen the following patterns: Send/Receive (pattern 3), Racing incoming messages (pattern 4), One-to-many send/receive (pattern 7), Multi-responses (pattern 8) and Relayed Request (pattern 12). For each of the patterns, its description and a model corresponding to one of the sample scenarios given in [2] are provided. Each scenario is then modeled in Let's Dance and the resulting models are informally discussed.

Send/Receive. The Send/Receive pattern is described as follows [2]: "A party X engages in two causally related interactions: in the first interaction X sends a message to another party Y (the request), while in the second one X receives a message from Y (the response)." The example for this pattern depicted in Figure 3 shows a payment service sending a payment to a retail service provider. The retail service provider sends a response indicating whether the payment details are valid or not. The interaction at the top shows the sending of the payment details by an actor playing the role "payment service" to an actor playing the role "retail service". These actors are referred to as "p1" and "s1" respectively. This interaction requires an acknowledgment (this is a design choice). The interaction at the bottom depicts the sending of the response. Again, this interaction requires acknowledgment. Both interactions are related via a precedes relationship, meaning that the lower interaction can only start if the upper interaction has been completed. In this case, the upper interaction is completed if the payment service has received the acknowledgment from the retail service. The local model for the actors participating in a given choreography would include every interaction where the party in question is participating. Thus, in the depicted example the local models for the participating parties "payment service" and "retail service" would be equivalent to the global model. Subsequently and for space reasons, we only show global models.

Racing incoming messages. In Figure 4 an example for the racing incoming messages pattern is depicted. This pattern is defined as follows [2]: "A party expects to receive one among a set of messages. These messages may be structurally different (i.e. different types) and may come from different categories of partners. The way a message is processed depends on its type and/or the category of partner from which it comes." The figure depicts the scenario of a manufacturing process, which "involves remote subcontractors and uses a pull-strategy to streamline its operations. Each step in the manufacturing process is undertaken by a subcontractor. A subcontractor signals intention to execute a step when it becomes available through a request. At the same time, progress is monitored by a quality assurance service. The service randomly issues quality check requests in addition to the pre-established quality checkpoints in the process. When a quality check request arrives, it is processed in full before processing any new quality check request or subcontractor intention. Similarly, when a subcontractor intention arrives, it is processed in full before processing any other check request or subcontractor intention. Thus, there are points in the process where quality checks and subcontractor intentions compete." Figure 4 describes this point in the choreography. The two interactions at the top of the figure show the possible receipt of two different types of messages by the manufacturer: "manufacturing request" and "quality check request". These interactions are connected via a two-way inhibits relationship. In the figure an undirected crossed edge is used as abbreviation for two directed crossed edges. This indicates, that after one of the two messages has been received, the other one can no longer be received. With skipping one of these elementary interactions, the following elementary interaction will also be skipped, since the according prerequisite will never be fulfilled. The "manufacturing request" interaction precedes a "manufacturing approval" interaction while a "quality check request" interaction precedes a "quality check response" interaction.

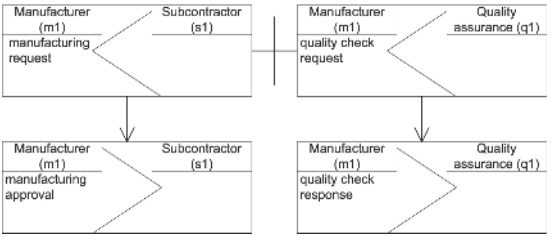


Fig. 4. Racing incoming messages (Pattern 4)

One-to-many send/receive. The one-to-many send/receive pattern goes as follows [2]: "A party sends a request to several other parties, which may all be identical or logically related. Responses are expected within a given timeframe. However, some responses may not arrive within the timeframe and some parties may even not respond at all. The interaction may complete successfully or not

depending on the set of responses gathered." The example for this pattern depicted in Figure 5 shows a scenario, where "an insurance company outsources some aspects of its claims validation to its external search brokers. Brokers are typically small agencies and have variable demands. For efficiency, the insurance company sends search requests to all the brokers, and accepts the first three responses to undertake the search." The depicted interaction shows the repetition of the sending of the search requests by the insurance company to the search brokers and the according responses. The actors playing the roles of the insurance company and the search brokers are referred to as "i1" and "B1". As introduced in Section 3, "i1" is named actor reference. "B1" is denoting a set of actor references, indicated by starting with a capital letter instead of a small letter for a single actor reference. The rectangle surrounding the two interactions is depicting a repeated composite interaction, whereby the repetition instruction and an additional stop-condition for the number of concurrent executions are noted in small rectangles on top of the composite interaction. Both, the repetition instruction and the stop-condition are executed and evaluated respectively by the actor referred to as "i1" and playing the role "Insurance". The repetition instruction denotes, that the message has to be sent to all actors that are referred to in the set of actor references "B1". This set of actor references is bound by the actor executing the repetition instruction. All iterations are executed concurrent, which is noted in brackets after the repetition instruction. The whole repeated interaction is initializing a variable "responses", which is indicated by the content of the small rectangle below the repeated interaction. This variable is increased by the lower interaction, showing the receipt of the responses from the brokers. The value of this variable is part of the stop-condition, denoting the iteration is stopped if this variable has the value 3. The usage of this variable does not contradict the postulation of omitting variables, noted in the requirements section. This variable is necessary from a business point of view, since even responses that have been gathered need not necessarily change the value, e.g. if the response does not contain the requested information.

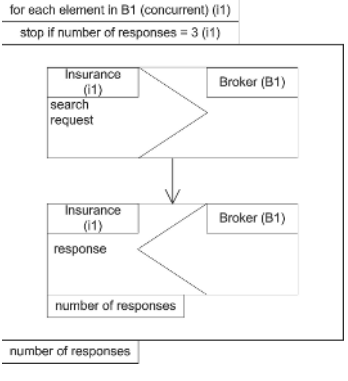


Fig. 5. One-to-many send/receive (Pattern 7)

Multi-responses. The pattern multi-responses is defined in [2] as follows: "A party X sends a request to another party Y. Subsequently, X receives any number of responses from Y until no further responses are required. The trigger of no further responses can arise from a temporal condition or message content, and can arise from either X or Ys side. Responses are no longer expected from Y after one or a combination of the following events: (i) X sends a notification to stop; (ii) a relative or absolute deadline indicated by X; (iii) an interval of inactivity during which X does not receive any response from Y; (iv) a message from Y indicating to X that no further responses will follow. From this point on, no further messages from Y will be accepted by X." The example for this pattern depicted in Figure 6 shows a goods deliverer who is providing an urgent transportation service. "For optimization of travel, it subscribes to a local traffic reporting service provides its destination nodes (goods dispatch and customer locations) and obtains regular feeds on traffic bottlenecks, until it indicates that no feeds are required." The top left interaction depicts the subscription of the traffic service by the goods deliverer. After the completion of this interaction the two remaining interactions of the choreography are enabled. The interaction on the right-hand side shows the receipt of traffic information by the goods deliverer. This interaction is repeated sequentially until the iteration of the interaction is interrupted, since the stop condition will is always false. Accordingly, the stop-condition is evaluated by the actor referred to as "t1" and playing the role "Traffic service". The interruption of the repetition occurs, if the third interaction is completed, which shows the sending of an unsubscribe message from the goods deliverer to the traffic service. This interaction is thus connected via a inhibits relationship with the repeated interaction.

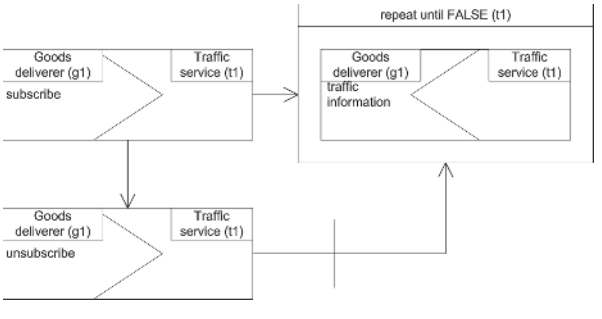


Fig. 6. Multi-responses (Pattern 8)

Relayed request. According to [2], the relayed request pattern is defined as follows: "Party A makes a request to party B which delegates the request to other parties (P1, ..., Pn). Parties P1, ..., Pn then continue interactions with party A while party B observes a view of the interactions including faults. The interacting parties are aware of this 'view' (as part of the condition to interact)." The example depicted for this pattern shows a government agency that outsources supportive work for managing regulatory provisions. Clients send requests to

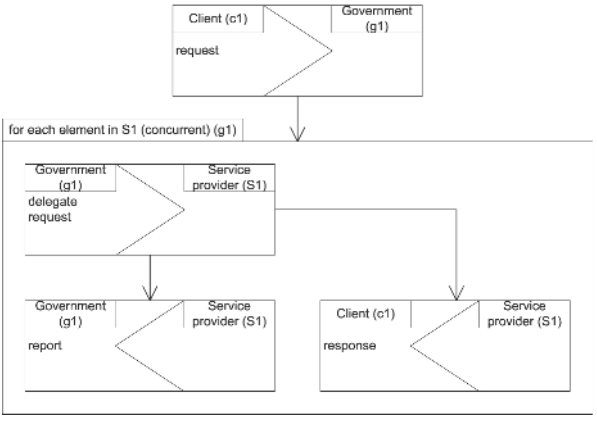


Fig. 7. Relayed Request (Pattern 12)

the government agency concerned by the regulation and the government agency forwards the request to the service providers. The government agency selects the service providers and the way they interact with the clients, e.g. key points of processing and key reports to be sent to the government agency. The above interaction shows the sending of a request from a client to the government agency. This interaction is connected via a precedes relationship with a repeated interaction, which is iterated concurrently for all service providers bound to a set of actor references "S1". The binding of this set of actor references, which is part of the repetition instruction, is executed by an actor referred to as "g1" and playing the role "Government". The government agency is delegating the request to each actor assigned to the set of actor references "S1" concurrently. This instruction for the number of executions is noted in a small rectangle on top of the repeated interaction. In this case there is no additional stop-condition for the "For each"-repetition, since it would equal to the maximum number of iterations expressed by the repetition instruction. During each iteration, the remaining two interactions are enabled after the service provider has received the request. The interaction on the right-hand side shows the sending of a response from the service provider to the client, while the interaction on the left side depicts the sending of a report to the government agency.

The solutions to the interaction patterns presented above and those given in [19] indicate that the Let's Dance language can deal with most relevant aspects necessary to model service choreographies. The following section presents a meta-model and an informal semantics of the language. In separate work [8] we have defined a formal semantics of the language by translation to π-calculus. We do not present details of this formalization here for space reasons.

5 Meta-model and Informal Semantics

Figure 8 provides an abstract syntax of the Let's Dance in the form of a meta-model captured in the Object-Role Modeling (ORM) notation [11]. The basic

concept of the meta-model is that of a Communication Action, which is performed by an Actor (not shown in the diagram) designated by exactly one Actor Reference. An Actor Reference (or more specifically the actor it refers to) plays at least one Role and one Role is played by at least one Actor Reference. Communication Actions have exactly one type, which can be either Message Sending or Message Receipt. These are the two basic communication primitives supported in service oriented architectures. Naturally, the party that performs the message sending is called the "sender" while the party that performs the message receipt is the "receiver". A Communication Action may require or may provide an acknowledgment. A Message Sending marked as "requiring acknowledgment" means that the sender expects to receive an "acknowledge" message from the receiver. In the case of a Message Receipt the receiver will send an "acknowledge" message after receiving a message if the Message Receipt is marked as "providing acknowledgment".

Communication Actions can write Variables and can be combined to form Interactions. Thereby a Message Sending action requiring acknowledgment (not requiring acknowledgment) can only be combined with a Message Receipt action providing acknowledgment (not providing acknowledgment).

An Interaction is a unit of information exchange and has exactly one type, which can be either Elementary (one-to-one) Interaction or Composite Interaction. If an Interaction is composed of other Interactions, we talk about the "sub-interactions" of a "super-interaction". Composite Interactions are a super-interaction of at least one Interaction and one Interaction can be the sub-interaction of at most one Composite Interaction. An Elementary Interaction involves two Communication Actions (one send and one receive) and corresponds to the logical exchange of a message from one party to another, that is, a party sends a message that the other party may receive. The sender can only perceive that the message was received if the interaction requires acknowledgment.

An Interaction has exactly one "Type of Repeated Interaction", which can be either None, Sequential for each, Concurrent for each, While or Until. Thus, the subtype Repeated Interaction has three specializations: For each, While and Until. The first one can refer to a Iteration Expression, which can be Actor-based (and thus consist of at least one Actor Reference) or Variable-based (and thus consist of at least one Variable). Additionally the Iteration Expression can have at most one Set expression assigned, which allows for the elaboration of the repetition instruction. Let's Dance does not impose any particular expression language for describing set expressions and conditions. Arguably, domain analysts would want to specify these in natural language and let developers refine them into an executable expression language.

Every repeated interaction has exactly one (stop) condition assigned. If this condition evaluates to true, it implies that the iteration must stop (in case of "Until" and "For each") or must continue (in case of "While"). A condition has at most one Conditional expression assigned, which allows for the elaboration of the repetition instruction. Moreover Conditions can also be assigned to each type

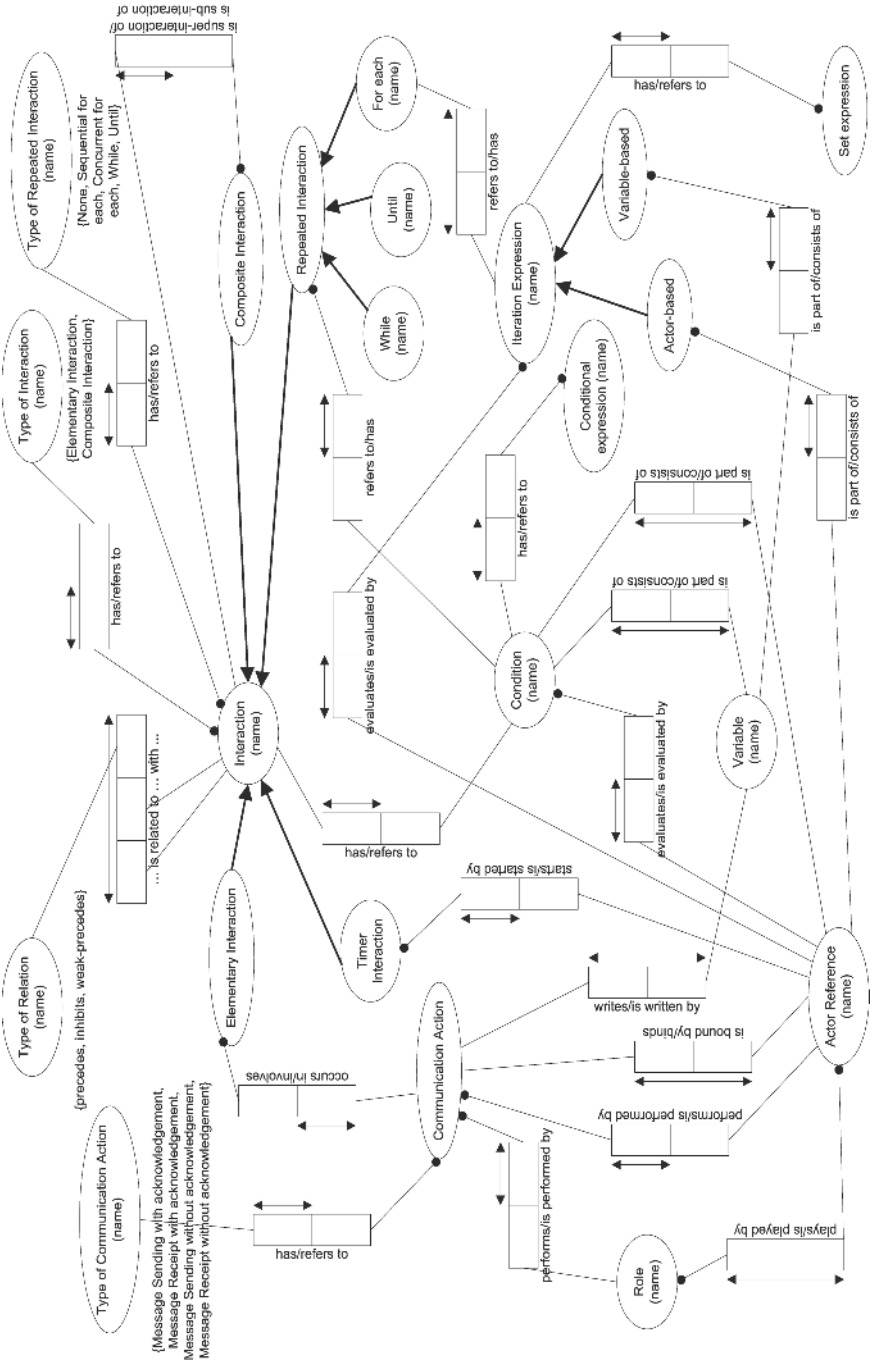


Fig. 8. Static Meta-model of the Language

of Interaction, denoting a guard for the Interaction, whereby each Interaction has at most one guard. The associated Conditional expression allows for the elaboration of the guard instruction. A Condition is evaluated by at least one Actor Reference and can consist of multiple Variables and Actor References.

Actor References are bound during the execution of at least one Communication Action and in a given scenario each Communication Action can alter bindings executed earlier. This enables the passing of information where specific messages should be sent to, e.g. in a buyer-seller-shipper scenario, where the buyer is binding the actor reference of the shipper in order to nominate a transport service. If the nominated shipper is not available, the seller might alter the binding of the actor reference and choose another transport service. For Composite Interactions this leads to the possibility to change the recipient of messages during the execution of an Interaction. During execution, the information about the binding has to be sent with every message following the Interaction during which execution a specific Actor Reference has been bound, until the Communication Action is reached, which is executed by the bound Actor Reference.

Since a Condition is evaluated by at least one Actor Reference, it is possible to describe dependencies between Communication Actions and Interactions occurring earlier. Thus Conditions have to be evaluated before executing the respective Communication Action. For a condition evaluated by an actor sending a message, this means before initiating the send action and for a condition evaluated by an actor receiving a message before the completion of the receipt action (in other words before the consumption of the message), since in the latter case it may be necessary for the receiving party to be aware of the content of the message in order to evaluate a stated Condition. For an Actor Reference to be able to evaluate a Condition at all, the necessary information has to be available. This can only be ensured, if the information needed for the evaluation of a Condition is sent with every message following the Communication Action which initialized a certain Variable or Actor Reference, until the information reached the specified Communication Action and Interaction respectively that has the Condition assigned.

Any two interactions may be related by a "precedes", a "weak-precedes" or an "inhibits" relation. These relationships are defined as follows: an interaction X is said to precede another interaction Y, if Y can not occur before X has been completed. If two interactions X and Y are connected with a weak-precedes relationship, then Y can not occur after X has been completed or after X has been skipped (i.e., seen from a global perspective, X will never occur). The inhibits relationship is defined as follows: an interaction X is said to inhibit another interaction Y, if Y can not occur any more after X has been completed. Moreover, the inhibits relationship is able not only to prevent the target interaction to be started, but also to interrupt it (and all of its sub-interactions), if it has already been started. These Relationships apply solely during the current execution of the least common super-interaction of the considered interactions. For example, if A and B are sub-interactions of interaction C and they are related through a

precedes relationship, then during each execution of C, B can only occur after A has been completed.

The subtype Timer of an Interaction allows one to enforce time limits on other interactions. It is started by exactly one actor (reference). A Timer is defined as a composition of two Elementary Interactions with a fixed party "Clock". The first Elementary Interaction is the arming of the Timer by sending a Message to the Clock (and the Clock receiving the Message), while the second Elementary Interaction is the sending of a Message from the Clock in order to indicate that the specified time period has expired. The latter of these messages is only sent if a condition is evaluated to true, which compares the elapsed time since the arming of the Timer with the period specified in the first Message. Since a Timer is a subtype of Interaction it can be related to other Interactions arbitrarily.

6 Conclusion and Future Research Directions

In this paper, we have motivated the need for a service behavior modeling language, spelled out a number of associated requirements, and proposed an initial version of a language (Let's Dance) that fulfills the most crucial of these requirements, namely abstraction, comprehensibility, and suitability. Unlike existing service behavior description languages such as BPEL and WS-CDL, which focus on supporting the implementation phase, the proposed language is not based on imperative programming constructs such as variable assignment, if-then-else and switch statements, sequence, and while loops. Also, the language supports the description of both local and global views of service interactions (i.e. behavioral interfaces and choreographies respectively).

The suitability of the language has been demonstrated on the basis of scenarios corresponding to 13 patterns of service interaction previously identified. The paper also presented an abstract syntax of the language in the form of a static meta-model, as well as an informal semantics. For a formal execution semantics of Let's Dance, defined in terms of a translation to π-calculus, we refer to [8]. This work also discusses the issue of reachability analysis of Let's Dance choreographies (i.e. detecting interactions in a choreography that will never be executed). A more in-depth discussion on desirable properties of Let's Dance choreographies is provided in [20]. In particular, this latter reference discusses the issue of local enforceability of Let's Dance choreographies, which is a prerequisite to generating local models from choreographies. It turns out that not all choreographies defined as flows of interactions (the paradigm adopted in Let's Dance) can be mapped into local models that satisfy the following conditions: (i) the local models contain only interactions described in the choreography; and (ii) they collectively enforce all the constraints in the choreography. Proposals around WS-CDL skirt this issue. Instead, they assume the existence of a state (i.e. a set of variables) shared by all participants. Participants synchronize with one another to maintain the shared state up-to-date. Thus, certain interactions take place between services for the sole purpose of synchronizing their local view on the shared state and these interactions are not defined in the choreography.

In the worst case, this leads to situations where a business analyst signs off on a choreography, and later it turns out that to execute this choreography a service provided by one organization must interact with a service provided by a competitor, unknowingly of the analyst. Thus, it is desirable to provide tool support to analyze choreographies to determine whether or not they are enforceable by some set of local models.

In [7], we present a tool that implements algorithms for static analysis of Let's Dance choreographies (including reachability and enforceability analysis) and for generation of local models. Ongoing work is concentrating on defining a translation from Let's Dance local models into BPEL code.

Acknowledgment. The first author is funded in part by SAP. The third author is funded by a "Smart State" Fellowship co-sponsored by Queensland Government and SAP.

References

1. T. Andrews, F. Curbera, H. Dholakia, Y. Goland, J. Klein, F. Leymann, K. Liu, D. Roller, D. Smith, S. Thatte, I. Trickovic, and S. Weerawarana. *Business Process Execution Language for Web Services, version 1.1*, May 2003. Available at: http://www-106.ibm.com/developerworks/webservices/library/ws-bpel
2. A. Barros, M. Dumas, and A. H.M. ter Hofstede. Service Interactions Patterns In *Proceedings of the 3rd International Conference on Business Process Management (BPM)*, Nancy, France, September 2005. Springer Verlag, pp. 302-218. Extended version available as Technical Report FIT-TR-2005-02, Faculty of IT, Queensland University of Technology, http://eprints.qut.edu.au/archive/00002295
3. N. Kavantzas, D. Burdett, G. Ritzinger, and Y. Lafon. *Web Services Choreography Description Language Version 1.0*, W3C Candidate Recommendation, November 2005. http://www.w3.org/TR/ws-cdl-10.
4. B. Benatallah, F. Casati, F. Toumani, and R. Hamadi. Conceptual Modelling of Web Service Conversations. In *Proceedings of 15th International Conference on Advanced Information Systems Engineering (CAiSE'03)*, Velden, Austria, June 2003. Springer Verlag, pp. 449-467.
5. J. Clark, C. Casanave, K. Kanaskie, B. Harvey, J. Clark, N. Smith, J. Yunker, K. Riemer (Eds). *ebXML Business Process Specification Schema Version 1.01*, UN/CEFACT and OASIS Specification, May 2001. http://www.ebxml.org/specs/ebBPSS.pdf.
6. W. Damm and David Harel. LSCs: Breathing Life into Message Sequence Charts. *Formal Methods in System Design*, 19(1), pages 45–80, Hingham, MA, USA, 2001. Kluwer Academic Publishers.
7. G. Decker, M. Kirov, J. M. Zaha, M. Dumas. Maestro for Let's Dance: An Environment for Modeling Service Interactions. In *Demonstration Session of the 4th International Conference on Business Process Management (BPM)*, Vienna, Austria, September 2006.
8. G. Decker, J. M. Zaha, M. Dumas. Execution Semantics for Service Choreographies. In *Proceedings of the 3rd International Workshop on Web Services and Formal Methods (WS-FM)*, Vienna, Austria, September 2006. Springer Verlag.

9. H. Foster, S. Uchitel, J. Magee, J. Kramer. Tool Support for Model-Based Engineering of Web Service Compositions. In *Proceedings of the IEEE International Conference on Web Servies (ICWS)*, Orlando FL, USA, July 2005. IEEE Computer Society.
10. T. R. G. Green, M. Petre. Usability Analysis of Visual Programming Environments: A 'Cognitive Dimensions' Framework. *Journal of Visual Languages and Computing* 7(2):131-174, 1996.
11. T. Halpin. *Information Modeling and Relational Databases - From onceptual Analysis to Logical Design.* Morgan Kaufman, 2001.
12. K. Jensen: *Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use – Volume 1.* Springer-Verlag, Berlin, 1997.
13. D. Martin, M. Paolucci, S. McIlraith, M. Burstein, D. McDermott, D. McGuinness, B. Parsia, T. Payne, M. Sabou, M. Solanki, N. Srinivasan, K. Sycara. Bringing Semantics to Web Services: The OWL-S Approach. In *Proceedings of the First International Workshop on Semantic Web Services and Web Process Composition (SWSWPC 2004)*, San Diego, California, USA, July 2004. Springer, pp. 26–42.
14. Object Mangement Group (OMG): *UML Profile for EDOC.* February 2004. http://www.omg.org/technology/documents/formal/edoc.htm
15. Object Management Group (OMG): *UML 2.0 Superstructure Specification.* OMG Document ptc/04-10-02, October 2004. http://www.omg.org/cgi-bin/doc?ptc/2004-10-02
16. D. Roman, U. Keller, H. Lausen, J. de Bruijn, R. Lara, M. Stollberg, A. Polleres, C. Feier, C. Bussler, and D. Fensel. Web Service Modeling Ontology. *Applied Ontology* 1(1):77–106, 2005.
17. RosettaNet: Partner Interface Protocols. http://www.rosettanet.org
18. S. White: *Business Process Modeling Notation (BPMN)* – Version 1.0, May 2004, http://www.bpmi.org.
19. J. M. Zaha, A. Barros, M. Dumas, A. ter Hofstede: *Let's Dance: A Unified Language for Service Behavior Modeling.* Technical Report FIT-2006, Faculty of IT, Queensland University of Technology. http://eprints.qut.edu.au/archive/00004468
20. J. M. Zaha, M. Dumas, A. ter Hofstede, A. Barros, G. Decker. Service Interaction Modeling: Bridging Global and Local Views. In *Proceedings of the Tenth IEEE International Conference on Enterprise Distributed Object Computing (EDOC)*, Hong Kong, China, October 2006. IEEE Computer Society.

Dependability and Flexibility Centered Approach for Composite Web Services Modeling

Neila Ben Lakhal[1], Takashi Kobayashi[2], and Haruo Yokota[1,2]

[1] Tokyo Institute of Technology, Department of Computer Science
neila@de.cs.titech.ac.jp
[2] Tokyo Institute of Technology, Global Scientific Information, and Computing Center
{tkobaya, yokota}@cs.titech.ac.jp

Abstract. The interest surrounding the Web services (WS) composition issue has been growing tremendously. In the near future, it is awaited to prompt a veritable shift in the distributed computing history, by making the Service-Oriented Architecture (SOA) a reality. Yet, the way ahead is still long. A careful investigation of a major part of the solutions proposed so far reveals that they follow a workflow-like composition approach and that they view failures as exceptional situations that need not to be a primary concern. In this paper, we claim that obeying these assumptions in the WS realm may constrain critically the chances to achieve a high-dependability level and may hamper significantly flexibility. Motivated with these arguments, we propose a WS composition modeling approach that accepts failures inevitability and enriches the composition with concepts that can add flexibility and dependability but that are not part from the WS architecture pillars, namely, the state, the transactional behavior, the vitality degree, and the failure recovery. In addition, we describe a WS composition in terms of definition rules, composability rules, and ordering rules, and we introduce a graphical and a formal notation to ensure that a WS composition is easily and dynamically adaptable to best suit the requirements of a continuously changing environment. Our approach can be seen as a higher level of abstraction of many of the current solutions, since it extends them with the required support to achieve higher flexibility, dependability, and expressiveness power.

1 Introduction

The interest surrounding the *Web services composition (WSC)* issue has been growing tremendously. To date, it has triggered a substantial amount of research efforts. A significant progress exemplified with the emergence of a myriad of specification languages (e.g., BPEL [1] and OWL-S [2]) and of a whole panoply of WSC strategies including dynamic WSC models (e.g., eFlow [3]), declarative WSC models (e.g., SELF-SERV [4]), and semantic WSC models (e.g., SHOP2 [5]), has been noticed.

Nevertheless, the WSC technology is still regarded as not mature enough and much more must happen before it reaches its apogee. A careful investigation of major part of the available solutions reveals that they share two limitations that preclude them from being solutions that meet modern IT environment requirements (i.e., high dependability and flexibility, automated discovery, composition, and enactment).

The first limitation is that the Workflow-based WSC approach is at the core of the majority of the current solutions. Representative examples are eFlow platform [3],

R. Meersman, Z. Tari et al. (Eds.): OTM 2006, LNCS 4275, pp. 163–182, 2006.

BPEL [1], and SELF-SERF framework [4]; all of these enable the composition of WS as Workflows. Actually, this was the direct consequence of the prevailing current of thought that the only major difference between Workflows and WSC is that the latter aims chiefly at taking XML-based standards (e.g., WSDL, SOAP), yet to reach the same goals. This line of thinking has yielded solutions that obey to restrictions that apply in the field of Workflow systems but that are not accounted for in the WS realm since:

- In Workflow systems, the cooperating parts are set a priori, and only those parts can cooperate within well-defined and closed boundaries with predefined policies and Service Level Agreements (SLAs). However, the WS environment is widely known for its heterogeneity, dynamism, and loose coupling, where different parts may interact beyond their boundaries, and where the interacting parts change even at runtime. Therefore, the possibility to continue the process enactment without interruption with the new changes is highly desired. Moreover, dealing with SLA in such a dynamic environment gets very complex since each WS may interact with many others simultaneously; if each interaction is governed by an SLA, managing the different SLAs becomes an overwhelming task [6].
- In Workflow systems, the definition of the process logic (i.e., decomposition of the process among participants) is rarely subject to changes. However, the ability to adapt easily, rapidly, and dynamically to changes without having to require the developer manual intervention to re-generate the overall WSC, is fundamental in the WS context. Otherwise, failure possibilities will be critical and the chances to succeed in doing the best matchmaking between process components and WS that are most likely to show better quality of execution will be seriously constrained.

As for the second limitation, many of the current WSC approaches, they only focus on how to interleave pre-existing WS into a WSC and they consider WS failures as exceptional situations that need not to be a primary concern. In fact, only very few cases provide simple failure handling mechanisms [1] [7]. We claim that in the WS realm, relying heavily on the Internet makes failures unavoidable. Adding to the Internet unreliability, a whole set of characteristics of the modern computing environments (e.g., unpredictability, heterogeneity, autonomy, dynamism, etc.) in which the deployed WS subsist, makes the most unexpected failure a normal part of the life of any WS. Furthermore, with assembling WS into WSC, failure frequency is more important than ever. In this paper, we opt for a radically different approach and we propose a novel WSC modeling approach that can be seen as a higher layer of abstraction of the solutions proposed so far, since it can extend them to support WSC in more powerful way, with an increasing level of flexibility and dependability.

The first key strategy in our approach to add expressiveness power and the required flexibility and dependability is to enrich the WSC model with concepts that are required but are not yet part from the WS architecture pillars. We emphasize that these concepts should be carefully selected to not restrict in any respect the WS architecture capabilities.

In our approach, we build mainly on the *state* (i.e., component behavior), *vitality degree*, and *transactional behavior* concepts, widely recognized by the database community to have contributed in achieving significant improvements in dependability [8].

The second key strategy to realize the desired flexibility is based on allowing the process definition (i.e., components order of execution, components nesting) changing

and the process components dynamic composition and/or decomposition in view of the WS availability/quality of execution. This will guarantee that the best matchmaking between the process components and the available WS could be done, and will ensure that the obtained WSC are easily adaptable to the environment change. We identify a process in terms of *Definition Rules(DR)* to provide the different components description, *Composability Rules(CR)* to inform about the relation between the components, and *Ordering Rule(OR)* to define the ordering relation between the components.

This paper is organized as follows. In Section 2, we introduce the limitations of adapting a workflow-based WSC approach, then, we discuss the key requirements for a flexible and dependable WSC. In Section 3, we define our model key concepts. In Section 4, we describe how our approach can be applied to an illustrative example. In Section 5, we discuss our approach contributions and show how the limitations identified in Section 2 can be addressed. In Section 6, we discuss several related work. In Section 7, we conclude our paper and introduce several future directions.

2 Motivating Example and Key Requirements

We describe a motivating example and discuss several specific enactment scenarios of a loan request process. In the light of this example, we will show the limitations of a Workflow-based WSC approach. In particular, we intend to show first, to what extent a such an approach may hamper significantly interoperation and leave no room for a dynamic process change to increase flexibility. Second, how such an approach may affect critically the dependability level of the overall composition. Throughout those scenarios, we identify the key requirements that a typical WSC approach must satisfy to achieve a higher level of flexibility and dependability. Specifically, we consider the case of BPEL.

We emphasize that we are perfectly aware that BPEL remains only a specification language, yet, in this paper, we propose a modeling approach, which would be assimilated more to proposals like [4] [7]. Our justification is that BPEL is the current defacto language and many are familiar with it. Thus, explaining our motivating example on the base of a BPEL description would be straightforward and clearer. Moreover, as we shall show later in this paper, our model can perfectly extend BPEL and can constitute a higher level of abstraction. Therefore, the discussion that will follow is not restricted to BPEL but it can be generalized to a wide range of solutions that share the same workflow-like WSC approach. [1]

2.1 Illustrative Example Description

We consider the case of a process where a customer requests a bank for a loan. The bank passes a subset of the customer information on to a credit rating agency that comes back with a credit report. The bank bases its decision on this data and responds to the customer, quoting an interest rate for the desired loan. The customer can then confirm the acceptance based on those terms or reject it. For illustration, the listing in (Appendix A.)

[1] For space limitation, we only added a comparison with [3] [4] [7]in the related work section.

shows the BPEL syntax for the bank loan request process from [9]. Each BPEL composition is a workflow that interacts with a set of WS to achieve a certain goal. BPEL WSC are called processes; the WS the process interacts with are called partners [10]. In the case of the bank loan request process, the WSC weaves together two WS — the creditAgencyService and the loanTermDetermination Service.

2.2 Limitations of a Workflow-Like Composition

A workflow-like composition does not support dynamic changes. For instance, in the BPEL listing of (Appendix A.), the only way to adapt the WSC to new changes in the process logic (e.g., new business rule, new management policy, etc.) is by reviewing all the interaction logic between the different partners and rewriting all the process. In the case of the above-described and simplified bank loan request process, regenerating the overall process specification can be somehow accepted; however, since processes tend to be complex, regenerating to overall process every time will be a time-consuming and costly task, and make such approach not satisfactory. The second limitation of a Workflow-like composition is the limited support for dynamic binding. We take again the case of BPEL. In a BPEL process, when a WSC is designed, it does not contain any reference to any specific WS; it only lists partner port types, thereby the abstract process appellation. To construct an executable process, the first alternative is when concrete WS are mapped to the partners. Once this mapping is defined, it is fixed for all invocations and cannot be changed automatically as the process runs. Even though this feature can allow some flexibility since it offers the possibility of assigning to the different partners different WS, this is far from being enough since the mapping is done statically before the WSC is invoked. The interaction between the different partners is only defined in one way and it presumes that the partners in question are always available. Since it is very likely that any of the partners become unavailable for any reason, the overall WSC success might be comprised. Adding the WS high-failure tendency, the WSC invocation failures rate will be unacceptable.

The second alternative introduced in BPEL is the possibility to select and assign actual partner services dynamically, and BPEL provides the mechanisms to do so via assignment of endpoint references. Even though this adds some form of dynamic binding, still it is not possible to change the assigned values to the activity as the process runs, to allow for instance, the execution with another alternative endpoint, in case of failure. So far, this mechanism was only added to extend the service endpoint information with instance-specific information to support the case of stateful partners.

2.3 Motivating Scenarios

Actually, there are many scenarios that a workflow-based composition can hardly support but that are highly desired to be feasible. We describe some of them in what follow, using the above BPEL example, and we point out the requirements that a typical WSC model must satisfy to support these scenarios:

Scenario 1. Based on the developer's decision, two WS are mapped to the two roles getCreditRating, getLoanTerms. If an error occurs at the WS bound statically to the role getLoanTerms, a compensation handler is invoked, and all the completed activities

will be compensated for. By providing the fault handlers and the compensation, BPEL allows to add some reliability support. However, declaring the process as failed should be the last resort if a successful execution is impossible. Envisaging the possibility of dynamically binding failed partners to other WS is more promising, especially in the WS context, since the probability of seeing a WS failing is high, and that WS providers offer their WS to several clients and clients can switch their providers. To this end, we introduce the *forward recovery* and *backward recovery*.

Scenario 2. When a BPEL process is to invoked, a mapping between WS and partners is set, and this mapping is fixed for all the executions. As the process runs, there is no mean to know the execution progress, since WS are generally stateless and BPEL provides only a correlation-based stateful interaction that allows only identifying instances. Another mechanism is required to identify the interacting parts progress as the process runs, and to derive the process instances progress. To fulfill this requirement, we propose to attach a *state* to each of the partners; we will detail its functioning later.

Scenario 3. BPEL assumes that there are imperatively two `partners` that must be defined a priori to enact the above process. However, knowing that the WS environment dynamism, it is very probable that new `partners`(WS) that can satisfy at the same time the functionalities of both of the two `partners` together are made available. For instance, assume that a WS that offers both of the two capabilities `creditRating` and `loanTermDetermination` is available. The way the WSC is defined in BPEL precludes from dynamically and transparently switching from two `partners` to only one `partner` without the process developers intervention to completely regenerate the BPEL process definition. Invoking the process with two `partners` instead of only one `partner`, will entail an unnecessary message exchange cost and a doubled failure risk. This can be avoided if the process definition is enriched with clauses that inform about the possibility of combining partners together, or exploding a partner in a set of partners and so forth, and dynamically taking such decision in view of the WS availability, without requiring the developer's intervention.

Scenario 4. Suppose that one of the process components is actually the execution of more than one operation; for instance, in partner `creditAgencyservice`, instead of having only one operation, two operations are invoked: `checkCreditType` and `checkCreditTypeRate`. Suppose also that in all the previous invocations of the service `creditAgencyservice`, a rather high failure tendency was noticed, and that in all these invocations, `checkCreditTypeRate` was almost always behind the failure.

In the other hand, assume that there are WS that can perform separately the functionalities of the two operations and with higher quality of execution. If the BPEL process definition can switch dynamically, and select two partners for `checkCreditType` and `checkCreditTypeRate`, then failure rate can be ameliorated and better results from the overall performance point of view can be acquired.

Scenario 5. Assume that a customer would like only to make a simulation of the loan process, that is, he only wants to have an idea about the credit rating and term, but does not desire to see its request considered for approval or rejection for the moment being. In the shown listing in (Appendix A.), the different activities in `loanApprovalProcess`

are combined in the structured activity `sequence`. Thus, a correct execution of the process requires the successful enactment of all the different combined primitive activities in the sequence. If any of those primitive activities will fail, a fault handler will be triggered. Adding a mechanism that will permit both, a loan simulation and a real loan request and decision, will increase the potential range of possible customers. To satisfy this requirement, we introduce the concepts of *vitality degree* where some partners are identified as optional whether the others are tailored as crucial for the overall process they compose. We shall detail the vitality degree in the following section.

To fulfill the requirements we identified using the above-described scenarios, and to make a composition suited in an ever-changing distributed environment, we present in detail how we address these requirements in the following section.

3 Composite Web Service Specification Model

In this section, we introduce the model we tailored to answer the need of a specification especially made to fit the specificities of the WS architecture. We first introduce the salient features of our model: the *process* and the *element* concepts, the *transactional behavior*, the *state*, the *vitality degree*, the *failure recovery*, and finally, the *flexibility support*. Then, we describe how we depict a WSC in terms of *Definition Rules*, *Composability Rules*, and *Ordering Rules*, how these rules are determined and noted.

3.1 Model Salient Features Description

Process and Element. The underpinning logic of a process (noted P_i) is depicted as a composite WS (noted CWS_i), with i ranging over the set of natural numbers $\mathbb{N}$ to designate different processes. Since we target a dynamic execution, where the process components are mapped on the fly to WS, instead of orchestrating a set of WS statically, we introduce the concept of *element* and use it as a unit of composition. Each *process* P_i is composed of different elements (noted $E_{i.k}$), where k is ranging over $[1..|P_i|]$ and $|P_i|$ is the cardinality of P_i, that is, the number of elements composing P_i. On executing composite WS, WS are to be mapped to the different elements. The possibility of choosing every time different WS and of composing and/or decomposing the process in various ways according to the WS availability, will increase the chances to achieve better quality of execution by doing the best matchmaking between elements and WS.

Transactional Behavior. It is widely recognized that there is a need for a transactional support in the WS world to leverage dependability. There is a number of proposal toward this direction including WS-Transaction [11], and WS-TXM [12]. The main limitation of these proposals is that they define models for centralized and peer-to-peer transactions, which support a two-phase coordination of WS. They do not support applications that involve dynamic composition of heterogeneous services in a peer-to-peer context. Moreover, we are strongly convinced that applying the conventional transaction model [8] would fail, given the loosely coupled nature of the WS environment and the locking mechanisms to preserve ACID properties inappropriateness.

There are several advanced transaction models to allow the definition of ACID-like properties. We investigated the applicability of several of them but we found out that

WSC warrant a particular transactional support specially shaped for them. We propose to inherit features of interest from several advanced transaction models [8], specifically, the *arbitrary nesting level*, the *relaxed ACID properties*, the transaction *compensation mechanism* for the backward recovery; and the *vitality degree*.

Failure Recovery. In our model, we define for each element, when possible, a *compensating* element. On a failure of an element to commit, we offer two choices: the first is to attempt the element *execution retrial* where the element is reattempted with another WS. If the first choice is not possible, then, a *backward recovery* is triggered by either compensating or aborting the element to bring back the overall CWS to a consistent state. To this end, for each compensatable element $E_{i.k}$, we define a compensating element, denoted $E'_{i.k}$, which will be invoked if a failure later in the execution of $E_{i.k}$ makes it necessary. $E'_{i.k}$ occurrence after $E_{i.k}$ will restore the CWS to a state which is an acceptable approximation of the state it had before the execution.

Vitality Degree. To improve the availability of the composite WS and to add flexibility in the way failures cascade through a process P_i, we distinguish a `vital` element ($E^{v}_{i.k}$) from a `non-vital` element ($E^{\bar{v}}_{i.k}$). The vitality degree obeys the following assumptions:

- A `vital` element execution must be successful for its parent process to commit. A `non-vital` element may abort without preventing its parent process commitment;
- Aborting a `vital` element will induce aborting the whole process it appertains to if there is no alternative WS to retry the failed element. Aborting a `non-vital` element will not be reflected on the execution of the process it appertains to.

In the remaining of this paper, the notation ($E_{i.k}$) without specifying the vitality degree is used for an element. The distinction between a `vital` element ($E^{v}_{i.k}$) and a `non-vital` element ($E^{\bar{v}}_{i.k}$) is only done when a special consideration is warranted.

State. Since the reasons behind an occurred failure should be investigated to avoid seeing the same scenario happening again, the WS stateless will make the task difficult, our model overcame this limitation by making the elements/processes stateful and attaching to each one of them a *state*. At any given time, the state of every element $E_{i.k}$ is assumed to be exclusively in one of the following predefined six states:

1. `Waiting`: $E_{i.k}$ is not yet submitted for execution;
2. `Executing`: $E_{i.k}$ is effectively being executed;
3. `Failed`: $E_{i.k}$ has encountered a failure;
4. `Aborted`: $E_{i.k}$ has received a request to abort itself and has obeyed to it;
5. `Committed`: $E_{i.k}$ has successfully terminated and was committed;
6. `Compensated`: $E_{i.k}$ has been compensated for.

Making the elements stateful helps increasingly to decide how to go forward in a process execution, to tell when a failure happens and the element(s) that is/are behind the failure. The state transition diagram is provided in [13].

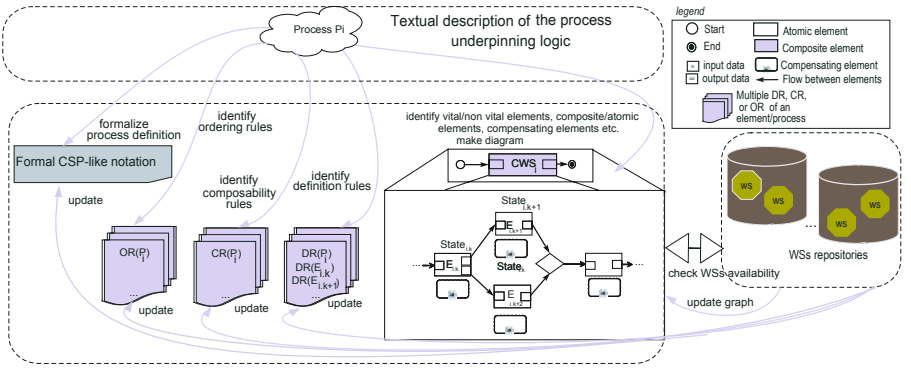


Fig. 1. Conceptual overview of the approach

Flexibility Support. We introduce the possibility of process components order changing, that is, we propose to allow the *arbitrary nesting* notion where the elements/processes can be recursively composed/decomposed in order to facilitate and ensure that the best matchmaking between the elements and the candidate WS is done.

Yet, this might not be possible/allowed for all kind of processes or it might be restricted to only some of its components. That is why it is essential to have a comprehensive knowledge of the process definition (i.e., its elements full description, distinguish optional from essential elements, in which order they can be interleaved). Since a textual description of the process underpinning logic may become complex and hard to update, we propose to describe a process by identifying three different kind of *Rules*, which share the same tuple-like notation. Yet, their semantics are different since the *Definition Rules (DR)* provide the different components description, the *Composability Rules (CR)* inform about the relationship between the different components, and the *Ordering Rule (OR)* specify the condition to which the relationship between the different components must verify. To understand the process logic and how its components relate to each other, we complement our specification model with a graphical notation we shall use later in a running example.

3.2 Definition Rules (DR) Determination

To describe a process P_i as composite WS CWS_i that follows our specification model, the first step is to identify the different *DR* on the base of the different business rules that the process logic defines (see Fig.1). Each *DR* gives relevant information about an *entity* that either relates to the composite WS specification (e.g., a process, an element, a component, etc.) or that intervenes in the composite WS execution (e.g., a WS, a coordinator, etc.). We adopt the following tuple-like generic notation to define a *DR*, where an *attribute* is an information about an *Entity*:

$$\begin{aligned} DR(Entity_i) &: \langle attribute_1, ... \rangle \\ DR(Entity_j) &: \langle attribute_2, ... \rangle \\ DR(Entity_k) &: \langle attribute_3, ... \rangle \ . \end{aligned} \tag{1}$$

Specifically, in our specification model, we define *DR* to provide relevant information about two different entities: *process* and *element*.

Definition 1 (Definition Rule of a Process ($DR(P_i)$)**).** *We define it by an ordered tuple that provides relevant information about P_i, namely its name, description, composability, state, vitality, and finally input and output parameters. Extending the* DR *with other attributes, such as QoS attributes is possible. The number of* DR *necessary to define P_i depends on the number of component elements initially identified. The notation of a process DR is a specialization of the generic* DR *notation in (1) for an Entity:*

$$DR(P_i): \langle name, description, composability, state, vitality, \widetilde{in}, \widetilde{out} \rangle \ . \tag{2}$$

Where :

- *name* is the name of the process and *description* is a concise description of the process main functionality;
- *composability* informs about the possibility of decomposing the process into elements. It verifies the condition: $DR(P_i).composability \in \{\texttt{true}, \texttt{false}\}$;
- *state* informs about the execution progress of the process as a whole. The state of P_i is deduced from the state of its different composing elements.
- *vitality* informs about the vitality degree of the process; it is defined according to the vitality degrees of the different component elements appertaining to the process. It verifies: $DR(P_i).vitality \in \{\texttt{vital}, \texttt{non-vital}\}$.
- $\widetilde{in}$ is a tuple composed by the different input parameters of the process. We denote with $\widetilde{in}$ tuples of input parameters; for instance, we may have: $\widetilde{in} =< in_1, in_2 ... in_p >$, for a process with p input parameters. Similarly, $\widetilde{out}$ are the different output parameters of the process. We denote with $\widetilde{out}$ tuples of output parameters.

Definition 2 (Definition Rule of an Element ($DR(E_{i.k})$)**).** *is an ordered tuple that provides relevant information about an element from a process P_i, with k ranging over* $[1..|P_i|]$. *For every element, $DR(E_{i.k})$is defined in the same way we defined it for a process. The same attributes apply for the element also. For every compensating element, we may define $DR(E'_{i.k})$ in the same way we defined it for a process/element.*

3.3 Composability Rules (CR) Determination

Each *CR* specifies the relation between the different *entities* defined by the different *DR* (i.e., how the entities are composed together, how the entities interact together, etc.). We adopt the following tuple-like generic notation to define a *CR* where $CR(Entity_i)$ indicates the relationship that $Entity_i$ defines between $Entity_j$ and $Entity_k$:

$$CR(Entity_i) \rightarrow \langle Entity_j, Entity_k \rangle \ . \tag{3}$$

In our model, the different *CR* identify the relationship of composability between the different elements and processes:

Definition 3 (Composability Rule of a Process ($CR(P_i)$)**).** *Defines the different elements the process is composed of. The CR existence/absence depends on the value of the composability attribute in the process corresponding DR:*

- *If (DR(P_i).composability =* `false`*) then the process is* `atomic` *and no CR are defined. It can be assimilated to only one* `atomic` *element;*
- *If (DR(P_i).composability =* `true`*) then the process is* `composite` *and among the different CR, we must have a predefined rule that says how the process can be decomposed in different elements and what are those elements;*

A CR *of a process is a specialization of the entity's* CR *described in (3) and it defines that the process P_i logic is composed of a set of m elements with $m = |P_i|$:*

$$CR(P_i) \rightarrow \langle E_{i.1}, E_{i.2}, \ldots, E_{i.k}, E_{i.k+1}, E_{i.k+2}, \ldots, E_{i.j}, E_{i.j+1}, E_{i.j+2}, \ldots E_{i.m} \rangle \ . \quad (4)$$

A particular CR *is the first rule that appears at the head of the list of* CR*; it describes the process logic as a whole: the set of the different elements it is composed by. Any* CR *can be recursively defined, that is, a particular element $E_{i.k}$ can be part from a process* CR*, yet, it can also define its own* CR*, if it is not atomic.*

Definition 4 (Composability Rule of an Element ($CR(E_{i.k})$)**).** *The existence of CR depends on the composability attribute. In case it verifies: DR($E_{i.k}$).composability =* `true`*, then, the element can be decomposed and it is defined with* CR*: it is considered as a particular process that has a parent process. For every element, its $CR(E_{i.k})$ is defined in the same way we defined it for a process. For every compensating element $E'_{i.k}$, we may define $CR(E'_{i.k})$ in the same way we defined it for a process/element.*

3.4 Ordering Rules(OR) Determination

Each *OR* defines the condition that the relationship between the different entities defined by the *CR* must verify (i.e., under which condition entities are composed together, under which condition entities interact together, etc.). We adopt the following tuple-like generic notation to define an *OR* in which *op* is the condition that the relation between $Entity_j$ and $Entity_k$ must verify: $OR(Entity_i) \rightarrow \langle Entity_j \ op \ Entity_k \rangle$. In our model, the different *OR* identify the ordering condition that every relationship of composability between elements/process should verify:

Definition 5 (Ordering Rule of a Process ($OR(P_i)$)**).** *It defines the order of execution of the component elements the process defined in its $CR(P_i)$. For a process P_i, if an $OR(P_i)$ is not explicitly defined, then the different elements order is interchangeable. A typical OR of a process is constructed as follows; op is the operator that connects the elements (e.g., sequence, join, etc.); it can be prefixed, post-fixed, or infixed:*

$$\begin{aligned} OR(P_i) \rightarrow \langle & (E_{i.1} \ op \ E_{i.2}) \ldots ((E_{i.k} \ op \ E_{i.k+1}) \ op \ E_{i.k+2}) \ldots \\ & (E_{i.j} \ op \ (E_{i.j+1} \ op \ E_{i.j+2})) \ op \ldots E_{i.m} \rangle \ . \end{aligned} \quad (5)$$

Definition 6 (Ordering Rule of an Element ($OR(E_{i.k})$)**).** *Similarly, it defines the order of execution of the components the element defined in its $CR(E_{i.k})$, if it is not* `atomic`*. For an element $E_{i.k}$. For every compensatable element, if it defines an ($OR(E_{i.k})$), then we should also define the corresponding OR for its compensating element, $OR(E'_{i.k})$, in the same way we defined it for a process/element.*

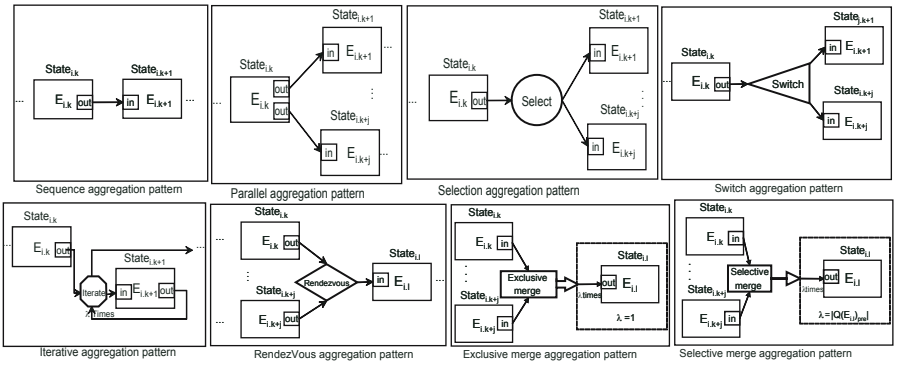


Fig. 2. Aggregation patterns considered by our approach

3.5 Aggregation Patterns

The *op* in the different *OR* varies and depends on the aggregation patterns the process logic follows. To define the different aggregation patterns, we build on existing work about Workflow patterns and on an analysis of existing Workflow languages reported in [14]. The following reasons motivated our choice: *a*) control flow dependencies encountered in Workflow modeling comply well with the WS context, since the situations they capture are also relevant in this domain; *b*) existing languages for WSC like BPEL and BPML were built on the basis of languages for Workflow modeling [15]. By building on the same patterns, we intend to provide a uniform comparison base.

The analysis of existing Workflow languages allowed us to identify the relevant patterns necessary to model the logic of any process, no matter how it is. We identified exactly eight patterns (see Fig.2). In [14], authors introduced twenty patterns, but we limited our study to only eight and we deliberately excluded the others (like the cancellation patterns or the state-based patterns) because, those eight, when combined with the *compensation*, the *state*, and the *vitality degree*, they are enough to express any process in the WS context. Yet, Workflow languages provide only a graphical notation of these patterns, and to date, there is no universal notation.

To define the different aggregation patterns, we propose to describe a syntax in the spirit of Compensating CSP (Hoare's Communicating Sequential Processes))[16][17] defined by a grammar in BNF-like notation. This will also allow formalizing our model and verifying its semantics, when required. We have chosen to build on Compensating CSP process algebra (PA) because it has already supports for compensation and for reasoning about long-running transactions. If the atomic events of CSP are used to model the elements in our model, then, the *op* in the different *OR* can be replaced with the operators provided by the CSP language to support sequence (;), and parallel(||). Besides, in order to support failed transactions, compensation operator (÷) is inherited from Compensating CSP. To support the remaining patterns from all the eight aggregation patterns, we introduce other required operators that CSP does not define.

Syntax. To define the different aggregation patterns, we describe a syntax in the spirit of CSP defined by the following grammar in BNF-like notation:

$$P_1, P_2, P_3 ::= E^{v}_{1.k} \mid E^{\bar{v}}_{1.j} \mid P_1; P_2 \mid P_1 \| P_2 \mid P_1 \star P_2 \mid \lambda P_1 \mid P_1 \div P'_1 \mid P_1 \bigcirc (P_2 \| P_3) \mid P_1 \lhd (P_2 \| P_3) \mid (P_1 \| P_2) \diamondsuit P_3 \mid (P_1 \star P_2) \boxdot\!\!\Rightarrow P_3 \mid (P_1 \star P_2) \boxdot\!\!\rightarrow P_3 \ . \quad (6)$$

- P_1, P_2 and P_3 designate three different process;
- $E^{v}_{1.k}$ and $E^{\bar{v}}_{1.j}$ represent respectively an atomic `vital` element and an atomic `non – vital` element appertaining to the process P_1, with k and j ranging in $[1..m]$ and m is the number of elements composing the process P_1. The vitality degree of P_1 is determined according to the vitality degrees' of its composing elements;
- $P_1; P_2$ represents the sequential construct that combines two processes. In $P_1; P_2$, P_1 is executed first, and only when P_1 terminates successfully can P_2 be executed;
- $P_1 \| P_2$ represents the parallel composition of two processes P_1 and P_2;
- $P_1 \star P_2$ represents the operator for constructing processes where the execution order is arbitrary: it can be in parallel, sequentially, or a conjunction of these two orders;
- $P_1 \bigcirc (P_2 \| P_3)$ represents the selective choice done by P_1 which selects whichever of P_2 and/or P_3 is/are to be enabled. $P_1 \lhd (P_2 \| P_3)$ represents a particular case since at least and at most only one is to be enabled;
- $(P_1 \| P_2) \diamondsuit P_3$ represents the case where the processes P_1 and P_2 are synchronized at a particular rendezvous point; that is to execute a particular process P_3 that comes directly after them, P_1 and P_2 must wait each other;
- $(P_1 \star P_2) \boxdot\!\!\Rightarrow P_3$ represents the case where P_1 and P_2 converge but without synchronization at a particular rendezvous point; that is, P_3 is activated every time a process reaches the rendezvous point. $(P_1 \star P_2) \boxdot\!\!\rightarrow P_3$ is similar since P_1 and P_2 converge but without synchronization at a particular rendezvous point, the difference is that the process that comes directly after P_1 and P_2 is activated only once;
- λP_1 represents iterating a process P_1 a number of times (λ);
- $P_1 \div P'_1$ represents the operator for constructing a compensation pair where P_1 is the forward behavior and P'_1 is its associated compensating process.

4 Illustrative Example

We consider the case of the same above-described process in BPEL of a customer request of a loan. Figure 3 is the graphical notation of the customer loan request process P_1, specified using our specification model. P_1 description as CWS_1 is composed of five elements with their compensating elements: four atomic elements and one composite element: $E^{v}_{1.5}$ is assimilated to an `atomic` element in CWS_1, while it is composed of two elements ($E^{v}_{1.5.1}$ and $E^{v}_{1.5.2}$).

4.1 Rules Determination

A CWS_1 is formed by the triplet combining the list of *DR*,*CR*,*OR* of the process P_1 and of its composing elements/compensating elements:

- Fourteen *DR*: $DR(P_1), DR(E^{v}_{1.1}), DR(E^{v}_{1.2}), DR(E^{v}_{1.3}), DR(E^{v}_{1.4}), DR(E^{v}_{1.5})$ $DR(E^{v}_{1.5.1})$, $DR(E^{v}_{1.5.2})$, $DR(E^{v\prime}_{1.1})$, $DR(E^{v\prime}_{1.2})$, $DR(E^{\bar{v}\prime}_{1.4})$, $DR(E^{\bar{v}\prime}_{1.5})$,$DR(E^{v\prime}_{1.5.1})$, and $DR(E^{v\prime}_{1.5.2})$;
- Two *CR*: $CR(P_1)$ and $CR(E^{v}_{1.5})$; and two *OR*: $OR(P_1)$ and $OR(E^{v}_{1.5})$. For space limitation, in what follow, we only describe some of them, the others can be defined in the same way (see Table 1).

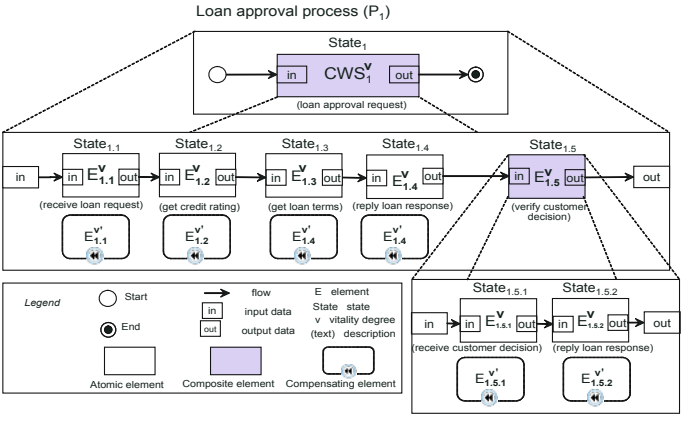


Fig. 3. Composition graphical notation: example of a customer loan request process

Table 1. Determination of the different rules of the loan request process

Definition Rules:	
$DR(P_1) : \langle name = \texttt{P}_1, description = \texttt{loanApprovalRequest}, composability = \texttt{true}, state = \texttt{Waiting}, vitality = \texttt{vital}, \widetilde{in} = \texttt{customerRequest}, \widetilde{out} = \texttt{customerDecision} \rangle$	
$DR(E^v_{1.2}) : \langle name = \texttt{E}_{1.2}, description = \texttt{receiveLoanRequest}, composability = \texttt{false}, state = \texttt{Waiting}, vitality = \texttt{vital}, \widetilde{in} = \texttt{customerRequest}, \widetilde{out} = \texttt{creditRating} \rangle$	
$DR(E^v_{1.5}) : \langle name = \texttt{E}_{1.5}, description = \texttt{verifyDecision}, composability = \texttt{true}, state = \texttt{Waiting}, vitality = \texttt{vital}, \widetilde{in} = \texttt{customerDecision}, \widetilde{out} = \texttt{customerDecision} \rangle$	
Composability Rules:	**Ordering Rules:**
$CR(P_1) \rightarrow \langle E^v_{1.1}, E^v_{1.2}, E^v_{1.3}, E^v_{1.4}, E^v_{1.5} \rangle$	$OR(P_1) \rightarrow \langle E^v_{1.1}; E^v_{1.2}; E^v_{1.3}; E^v_{1.4}; E^v_{1.5} \rangle$
$CR(E^v_{1.5}) \rightarrow \langle E^v_{1.5.1}, E^v_{1.5.2} \rangle$	$OR(E^v_{1.5}) \rightarrow \langle E^v_{1.5.1}; E^v_{1.5.2} \rangle$

4.2 Formalization

The formal notation using the syntax in the spirit of CSP that we defined in (6):

$$P_1 = (E^v_{1.1} \div E^{v\prime}_{1.1}); (E^v_{1.2} \div E^{v\prime}_{1.2}); (E^v_{1.3} \div E^{v\prime}_{1.3}); (E^v_{1.4} \div E^{v\prime}_{1.4}); E^v_{1.5}$$
$$E^v_{1.5} = (E^v_{1.5.1} \div E^{v\prime}_{1.5.1}); (E^v_{1.5.2} \div E^{v\prime}_{1.5.2}) \ .$$

4.3 Scenarios Feasibility Verification

We will show how the different concepts our approach introduces are enough to support a highly flexible and dependable WSC. In particular, we show how the scenarios described in Section 2 can be supported:

In **scenario 1.**, as the process runs, a dynamic WS discovery and mapping process is performed for each of the elements $E^v_{1.2}$ and $E_{1.3}$, realizing respectively the functionalities of `getCreditRating` and of `getLoanTerms` in the process P_1. Assume that the

mapped WS to $E^v_{1.2}$ failed, then, instead of stopping the overall execution, or requiring the developer's intervention to select another WS, a forward recovery is attempted by allocating automatically any of the other candidate WS that satisfy the conditions:

- $DR(E^v_{1.2}).description \equiv DR(WS).description$ *(i.e., element functionalities* $\equiv$ *WS capabilities)*;
- $DR(E^v_{1.2}).\widetilde{in} \equiv DR(WS).\widetilde{in}$ *(i.e., element & WS input parameters are compliant)*;
- $DR(E^v_{1.2}).\widetilde{out} \equiv DR(WS).\widetilde{out}$ *(i.e., element & WS output parameters are compliant)*.

The allocated WS is invoked and the overall process execution can transparently resume without interruption. The other extreme possibility is that all the candidate WS are attempted and none of them was committed. Then, in such a case, a backward recovery is triggered by compensating all the committed elements and by aborting all the still-executing elements. In the case of the process P_1, $OR(P_1)$ indicates that the element $E^v_{1.1}$, is the only element that preceded $E^v_{1.1}$, hence, it will be compensated for by executing its compensating element $E^{v'}_{1.1}$. In ***scenario 2.***, we pointed out the importance of making the process stateful and we proposed to attach to the different elements a *state*. If we add a mechanism —like the one we introduced in [13]— that first, it stores in a particular location (i.e., in the entity responsible of the process execution) all the different *DR*. Second, it keeps track of the execution progress of the process by updating it on every change in any of the elements' *state*. Then, we can have a history of the different executions of all the invoked instances.

Furthermore, if we keep the history of all the invoked instances, we can have a step-by-step execution progress of all the process instances that we can analyze later to have valuable data about the failure reasons/locations [18].

The description of the `bank loan request` process in terms of *DR*, *CR*, and *OR* provides the exact requirements to realize ***scenario 3.*** and ***scenario 4.***. Assume that an analysis of the previous instances execution log has shown that the element $E^v_{1.3}$ is frequently the stage of failure. Assume also that a WS discovery query has shown that there are other newly added WS that satisfy the following conditions:

- $DR(E^v_{1.2}).description \cup DR(E^v_{1.3}).description = DR(WS).description$ *(i.e., functionalities of the two-joined elements*$\equiv$ *WS capabilities)*;
- $DR(E^v_{1.2}).\widetilde{in} = DR(WS).\widetilde{in}$ *(i.e., the input parameters of* 1^{st} *element and of the WS are compliant)*;
- $DR(E^v_{1.3}).\widetilde{out} = DR(WS).\widetilde{out}$ *(i.e., the output parameters of* 2^{nd} *element and of the WS are compliant)*;
- $DR(WS).\widetilde{operation} =< op_1, op_2 >$ *(i.e.,* op_1 *and* op_2 *have same order/semantics as* $E^v_{1.2}$ *and* $E^v_{1.3}$ *in* $OR(P_1)$*)*.

If these conditions are verified, then, we can transparently select one of the newly-discovered candidate WS and allocate it to perform instead of $(E^v_{1.2} \cup E^v_{1.3})$; a new element's *DR* and *OR* can be added. Similarly, to support ***scenario 4.***, we can explode an element transparently and map it to more than one WS, if required. We emphasize here that we are perfectly aware that the above described conditions in ***scenario 1.***, ***scenario 3.***, and ***scenario 4.***, defined for element-WS matchmaking are insufficient. We deliberately gave only simplified conditions that can be developed in a future work.

Table 2. The formal notation of the different aggregation patterns

Pattern	Notation	Pattern	Notation
Sequence	$(E_{i.k}; E_{i.k+1})$	Parallel	$(E_{i.k}; (E_{i.k+1} \| \| ... \| \| E_{i.k+j}))$
Rendezvous	$((E_{i.k} \| \| E_{i.k+1} \| \| \ldots \| \| E_{i.k+j}) \diamond E_{i.l})$	Switch	$(E_{i.k} \lhd (E_{i.k+1} \| \| E_{i.k+2} \| \| \ldots \| \| E_{i.k+j}))$
Selective merge	$((E_{i.k} \star \ldots \star E_{i.k+j}) \boxminus\!\!\rightarrow E_{i.l})$	Selection	$(E_{i.k} \bigcirc (E_{i.k+1} \| \| E_{i.k+2} \| \| ... \| \| E_{i.k+j}))$
Exclusive merge	$((E_{i.k} \star \ldots \star E_{i.k+j}) \boxminus\!\!\Rightarrow E_{i.l})$	Iteration	$(E_{i.k}; \lambda E_{i.k+1})$

Since assessing the similarity of WS to achieve the best matchmaking is an active area of research, we can apply one of the available proposals ranging from keyword-based methods to ontologies and reasoning algorithm enriched methods.

The *vitality degree* concept permits the fulfillment of ***scenario 5.***. If the composite element $E^{v}_{1.5}$ becomes `non-vital`, that is, we have: $DR(E_{1.5}).vitality$ = `non-vital`, $DR(E_{1.5.1}).vitality$ = `non-vital`, and $DR(E_{1.5.2}).vitality$ = `non-vital`, then, as the process runs, even when a customer does not reach the element $E^{\bar{v}}_{1.5}$, the process execution will be committed successfully. With changing transparently the *vitality degree*, we can have different scenarios of the same process. Finally, these scenarios have shown that we can modify more easily and transparently the process logic by only handling its different *DR*, *OR*, and *OR*, exploding some rules, and joining others when required.

5 Discussion

We discuss the contributions in dependability and flexibility enhancement provided by our approach against the Workflow-based composition approaches. For clarity sake, our discussion is based on a confrontation of the BPEL specification of the loan approval process with the specification of the same process we presented in the previous section. Yet, the same arguments are valid for the other solutions that follow the same Workflow-like composition strategy.

Our model is more expressive than BPEL since we define three kind of notations: a textual notation, a graphical notation, and a formal notation. The textual notation can be more easily understood by a business analyst who typically does not have any programming experience and who has to update the business rules embedded in the process, which cannot be said about the syntax of BPEL. Our textual notation can be used as a higher level of abstraction on the top of BPEL. There is no universal graphical notation for a BPEL process but there are several proposals to use UML-like notation for descriptions. However, a standard notation is required to shorten the time necessary for understanding the process.

In [19], authors presented an analysis of the different Workflow aggregation patterns that BPEL can support. To have a common comparison basis, we considered in this work the same set of patterns. The patterns defined by BPEL are a subset of the patterns supported by our specification model, and that we report in (Table 2); a detailed description of these patterns can be found in [20]. Our different notations are rich enough to depict all the different patterns that BPEL supports. This can be particularly helpful since the constructs defined by BPEL tend to be complex and their semantics are not always clear and can be sometimes misleading. For verifying the correctness of the WSC,

BPEL provides no way to verify correctness. Several tentative work have tried so far to formalize BPEL and they employ formalisms based on Pas [21]. However, BPEL is dealing more with implementation than specification. Thus, it is difficult to provide a formalism to verify the correctness of BPEL flows. To allow the analysis of our model, similarly, we introduced the CSP-like notation. We emphasize that the verification of our model is beyond the scope of this paper, and that we intended to provide a solid ground of comparison of our work with others proposal that use similar formalisms.

The vitality degree concept allows introducing a higher flexibility. For instance, consider the listed BPEL process in (appendix A). The following listing shows how the vitality degree can be added:

```
<process name="loanApprovalProcess" state="waiting" vitality="vital"···>
<invoke name="getLoanTerms" state="waiting" vitality="vital"···></invoke>...
<receive name="receiveCustomerDecision" state="waiting" vitality="non−vital"···></receive>
<reply name="replyLoanResponse" state="waiting" vitality="non−vital"···></reply></sequence></process>
```

If we change the vitality degree of the element $E^{v}_{1.5}$ from `vital` to `non-vital`, the loan approval process can be used also as a simulation for loan request process. Since when the element $E^{\bar{v}}_{1.5}$ is `non-vital`, the customer does not have to go up to its execution, and $E^{\bar{v}}_{1.5}$ failure or non invocation cannot affect the overall process commitment.

Similarly, the *state* concept can be attached to the different activities in BPEL and can provide very valuable information about the enactment progress of the different activities and of the process as a whole. Introducing the state concept can contribute to acquiring the information necessary for discovering the faulty WS and discarding them to improve the performances. Further details can be found in our previous work [18] about the state-guided failure analysis approach, we proposed. Finally, the specification of the process in terms of *DR*, *CR*, and *OR* allows a more flexible enactment. If we define in a higher level on the top of BPEL, a specification of the process in terms of *DR*, *CR*, and *OR*, and we provide the enactment environment with the possibility of regenerating dynamically the process definition, then we can achieve better results.

6 Related Work

Making several entities work in tandem to reach a common goal is not a new challenge in itself since it has been widely addressed in several areas. In particular, mechanisms like the flexible process composition and decomposition —we propose in our approach— were also proposed in other areas. The database community introduced the notion of transaction decomposition into steps. A representative work is [22]. In their approach, the authors target was to increase concurrency whereas we target flexibility. In addition, they proposed mechanism based on two-phase locking, not suitable for the WS context. What is interesting about this work is that it directed our attention toward the importance of generating correct process decomposition, and how developers should obtain one; which can constitute a promising future direction for us.

In the area of Workflow systems, massive research efforts were focused toward the need for flexibility at runtime to avoid seeing the Workflow execution stopped due to resources unavailability and to allow the users to deviate from the prescribed sequence of

activities the Workflow definition imposes. The work reported in [23] exemplifies those works. This work presented a mechanism for choosing alternatives to sub-Workflows, thereby allowing the Workflow execution to resume when the predefined sub-Workflow is unavailable; which present some overlaps with the ideas reported in our paper.

We can say the same thing about several formalisms from the area of AI, namely, planning techniques and problem-solving methods (PMS). The work in [24] described a framework that modeled a WSC as a PMS that describes how the WS is decomposed into its components. The concept of task decomposition in HTN (Hierarchical Task Network) planning is very similar to the concept of decomposition in our approach. The work [5] reported how to exploit AI planning techniques for automatic WSC by treating WSC as a planning problem. The SHOP2 system creates plans by task decomposition. This work focused mainly on translating OWL-S WS descriptions to a SHOP2 domain. This particular view oriented our attention toward the similarity between the concept of decomposition in our approach and the concept of composite process in OWL-S ontology. Our approach can complement this work with our different notations and especially with our transactional behavior and flexibility supports.

In the area of WSC, [7] introduced WebTransact. To date, this work is one of the few that introduced a transactional support by defining a WS Transaction Language on top of WSDL enhancing it with functionalities facilitating transactional WSC.

The main differences between our approach and WebTransact are that, first, the WS are statically integrated in WebTransact by a developer who plays the role of WS integrator. Yet, this is not a flexible way. Second, WebTransact is mainly for integrating WS that have their own local transaction support, yet this condition is not always verifiable since not all WS have a transaction support, and not all of them are compliant. Our approach only supports compensatable, retriable, vital, and non-vital behaviors, but it is easily extensible with other transaction behaviors (e.g., pivot). Finally, WebTransact does not address the flexibility issue and it only provided an XML-based notation, hardly understandable and difficult to update. We consider also eFlow [3] and SELF-SERV [4], where WSC are created dynamically. In eFlow, the definition of a service node encloses a service selection rule. When invoking the service node, the rule is executed to select a specific WS. eFlow supports dynamic process change by migration of process instances from a source schema to a destination schema. Concerning SELF-SERV, it exploits the concept of service community, a container of alternative services. At run-time, when a community receives a request, it delegates it to one of its current members. Separating the service description from the actual service provider increases the flexibility. However, SELF-SERV does not support dynamic process changes such as adding new WS type to the WSC. SELF-SERV advocates the use of state-charts instead of formal specifications for easiness to use and modularity whereas eFlow used a proprietary notation.

7 Conclusion

In this paper, we introduced a novel approach for a flexible and dependable composite WS specification. Our approach puts forward the view that WSC failures are not exceptional situations —as exposed by many, but it takes a radically different view by accepting that failures are inevitable for any WSC. In addition, we noticed that the

Workflow-based composition approach is at the core of the majority of the current solutions and we claimed that this line of thinking has yielded solutions that obey to some restrictions that do not meet modern IT environment requirements. We discussed various limitations caused by a Workflow-like composition approach.

In our approach, we enriched the WSC model with several mechanisms that can add flexibility and dependability to the WSC but that are not part from the WS architecture pillars, namely, the state, the transactional behavior, the vitality degree, and the failure recovery mechanisms. We identified a process in terms of *Definition Rules (DR)* to provide the different components description, *Composability Rules (CR)* to inform about the relationship between the components, and *Ordering Rule (OR)* to define the ordering relationship between the components. We also introduced three notations: a graphical, a textual, and a CSP-like notation. Our approach can be seen as a higher layer of abstraction of the solutions proposed so far, since it can extend them to support WSC in more powerful way, with increasing level of flexibility and dependability.

Our notations allow modeling WSC in a formal and concise way, which can then be used to verify the correctness of the WSC and reason on their semantics without hindrance from technical limitations of an XML-based notation. As for the execution environment, a distributed architecture, like the architecture we introduced in [13].

We can achieve the full potential of our approach since it provides the essential concepts to realize the described failure recovery mechanisms, and the dynamic and distributed execution. Currently, we are still working on an implementation of our proposed model. Our next steps will be finalizing the implementation, experimentally measuring its performances, and comparing it with other approaches.

Acknowledgement. Part of this research was supported by CREST of JST (Japan Science and Technology Agency), a Grant-in-Aid for Scientific Research on Priority Areas from MEXT of the Japanese Government (#16016232), and the 21^{st} Century COE Program Framework for Systematization and Application of Large-scale Knowledge Resources.

References

[1] IBM, Systems, B., Microsoft, AG, S., Systems, S.: Bpel4ws business process execution language for web services (2005)
[2] Martin, D., al.: Owl-s: Semantic markup for web services. www.daml-s.org (2004)
[3] Casati, F., Ilnicki, S., Jin, L., Krishnamoorthy, V., Shan, M.: Adaptive and dynamic service composition in eflow. In Wangler, B., Bergman, L., eds.: 12th inter. Conf. on Advanced information Sys. Eng. Volume 1789 of LNCS., London (2000) 13–31
[4] Benatallah, B.and Dumas, M., Sheng, Q.: Facilitating the rapid development and scalable orchestration of composite web services. Dist. and Parallel Databases **17**(1) (2005) 5
[5] Wu, D., Parsia, B., Sirin, W., Hendler, J., Nau, D. In: Automating DAML-S Web Services Composition Using SHOP2. Volume 2870. LNCS, Springer-Verlag (2003) 195 – 210
[6] Sahai, A., Machiraju, V., Sayal, M., van Moorsel, A.P.A., Casati, F.: Automated sla monitoring for web services. In: DSOM '02: Proc. of the 13th IFIP/IEEE Int.Workshop on Distributed Sys., London, UK, Springer-Verlag (2002) 28–41
[7] Pires, P.F.and Mattoso, M., Benevides, M.: Building Reliable Web Services Compositions. In: Web, Web-Services, and Database Systems. Volume 2593. LNCS (2002) 59–72

[8] Elmagarmid, A.: Database transaction models for advanced applications. Morgan Kaufmann, San Mateo, California (1992)
[9] Khalaf, R., Mukhi, N., Weerawarana, S.: Service-oriented composition in bpel4ws. In: WWW (Alternate Paper Tracks). (2003)
[10] Curbera, F., Khalaf, R., Mukhi, N., Tai, S., Weerawarana, S.: The next step in web services. Commun. ACM **46**(10) (2003) 29–34
[11] Cabrera, F., et al.: Specification: Web services transaction (ws-transaction). http://www-106.ibm.com/developerworks/webservices/library/ws-transpec/ (2002)
[12] Fujitsu, IONA, O., Sun, A.T.: Web services composite application framework(ws-caf). http://www.arjuna.com/standards/ws-caf/ (2003)
[13] Benlakhal, N., Kobayashi, T., Yokota, H.: Throws: An architecture for highly available distributed execution of web services compositions. In: IEEE 14th Intl. Workshop on Research Issues on Data Engineering: Web Services for E-Commerce and E-Government Applications (RIDE'04), Boston, USA, IEEE (2004) 103–110
[14] van der Aalst, W., Hofstede, A., Kiepuszewski, B., Barros, A.: Workflow patterns. Distributed and Parallel Databases **14**(1) (2003) 5–51
[15] Aalst, W.: Don't go with the flow: Web services composition standards exposed. IEEE Intelligent Systems **18**(1) (2003) 72–76
[16] C.A.R.Hoare: Communicating Sequential Processes. Prentice Hall (1985)
[17] Butler, M.J., Hoare, C.A.R., Ferreira, C.: A trace semantics for long-running transactions. In: 25 Years Communicating Sequential Processes. (2004) 133–150
[18] Benlakhal, N., Kobayashi, T., Yokota, H.: A failure-aware model for estimating and analyzing the efficiency of web services compositions. In: IEEE 11th Int.Symp.on Pacific Rim Dependable Computing (PRDC2005), Changsha,China, IEEE (2005) pp.114–121
[19] Wohed, P., van der Aalst, W.M., Dumas, M., al.: Pattern based analysis of bpel4ws (2002)
[20] Benlakhal, N., Kobayashi, T., Yokota, H.: A well-defined and failure-aware model for estimating the efficiency of web services compositions. In: Proc. of IEEE Intl. Workshop on Challenges in Web Information Retrieval and Integration(WIRI) In conjunction with ICDE2005, Tokyo, Japan, IEEE (2005) 47–54
[21] Salan, G., Bordeaux, L., Schaerf, M.: Describing and reasoning on web services using process algebra. In: Proc. of Int. Conf. on Web Services, USA, IEEE (2004) 43
[22] Ammann, P., Jajodia, S., Ray, I.: Applying formal methods to semantic-based decomposition of transactions. ACM Trans. Database Syst. **22**(2) (1997) 215–254
[23] Vieira, T.A.S.C., Casanova, M.A., Ferrao, L.G.: An ontology-driven architecture for flexible workflow execution. la-webmedia **00** (2004) 70–77
[24] Gomez-Perez, Gonzalez-Cabero, R., Lama, M.: Ode sws: A framework for designing and composing semantic web services. IEEE Intelligent Systems **19**(4) (2004) 24–31

Appendix A. BPEL Specification of a Loan Request Process

```
<process name="loanApprovalProcess"...>...<partners><containers>
<container name="customerRequest" messageType="loandef:customerRequestMessage"/>
<container name="creditRequest" messageType="creditagency:creditRequestMessage"/>
<container name="creditRating" messageType="creditagency:creditRatingMessage"/>
<container name="loanTerms" messageType="loandef:loanTermsMessage"/>
<container name="customerDecision" messageType="loandef:customerDecisionMessage"/></containers>
<partner name="customer"..." myRole="approver"/>
<partner name="creditAgencyService"..." partnerRole="creditAgency"/>
<partner name="loanTermDeterminationService"..." partnerRole="loanTermDecider"/></partners>
<sequence><receive name="receiveLoanRequest" partner="customer"..." operation="approve"...>
container="customerRequest"</receive>
<invoke name="getCreditRating" partner="creditAgencyService"..." operation="checkCredit"
inputContainer="creditRequest" outputContainer="creditRating"></invoke>
<invoke name="getLoanTerms" partner="loanTermDeterminationService"...
operation="getLoanTerms" inputContainer="creditRating" outputContainer="loanTerms"></invoke>
<reply name="replyLoanResponse" partner="customer"" operation="approve" container="loanTerms"
</reply><receive name="receiveCustomerDecision" partner="customer"..." operation="confirm"
container="customerDecision"></receive><reply name="replyLoanResponse" partner="customer"...
"operation="confirm" container="customerDecision"</reply></sequence>
```

Aspect-Oriented Workflow Languages

Anis Charfi* and Mira Mezini

Software Technology Group
Darmstadt University of Technology
{charfi, mezini}@informatik.tu-darmstadt.de

Abstract. Most available aspect-oriented languages today are extensions to programming languages. However, aspect-orientation, which is a paradigm for decomposition and modularization, is not only applicable in that context. In this paper, we introduce aspect-oriented software development concepts to workflow languages in order to improve the modularity of workflow process specifications with respect to crosscutting concerns and crosscutting changes. In fact, crosscutting concerns such as data validation and security cannot be captured in a modular way when using the constructs provided by current workflow languages. We will propose a concern-based decomposition of workflow process specifications and present the main concepts of aspect-oriented workflow languages using AO4BPEL, which is an aspect-oriented workflow language for Web Service composition.

Keywords: Modularity, Separation of Concerns, Aspect-Oriented Software Development, Workflow Languages.

1 Introduction

A workflow process is a specification of a business process in a form that can be executed by a workflow management system. Typically, a workflow process specification defines a set of *activities*, the order of their execution, the flow of data between them, the participants that perform them, and the applications that support their execution. The specification of a workflow process involves several *workflow aspects* or *workflow perspectives* [12, 27]. The authors of [27] differentiate five perspectives: the functional, the informational, the behavioral, the operational, and the organizational. In the following, we will use the term workflow perspective to avoid confusion with the term *aspect* [20] as used in the terminology of aspect-oriented programming.

In this paper, we focus on the modularity of workflow process specifications. Some few workflow languages such as MOBILE [18] allow for modular workflow process specifications by separating the parts of the specification that correspond to the different workflow perspectives. We will focus on the separation of concerns and crosscutting concern modularity rather than the separation of

* Supported by the German Research Foundation (DFG) in the context of the Phd Program Enabling Technologies for Electronic Commerce.

R. Meersman, Z. Tari et al. (Eds.): OTM 2006, LNCS 4275, pp. 183–200, 2006.

perspectives. We observe that current workflow languages do not allow to properly modularize concerns that cut across process boundaries such as security and data validation. Thus, the process code[1] of those crosscutting concerns is spread across several workflow process specifications (*the scattering problem*) and intertwined with the process code addressing other concerns (*the tangling problem*). The lack of a module concept for crosscutting concerns in workflow languages leads to monolithic and complex workflow process specifications that are hard to understand, maintain, change, and reuse.

In addition, current workflow languages do not support a modular expression of changes (and especially crosscutting changes) as first-class entities. Workflow changes span the different workflow perspectives and there is no module concept for encapsulating all workflow constructs that belong to some change. This makes understanding and managing changes (e.g., undoing changes) a difficult task.

Similar modularity problems have been identified in the context of programming languages, which lack mechanisms to encapsulate the code of a crosscutting concern such as security, logging, persistence, etc. Aspect-Oriented Software Development (AOSD) [9] has emerged as a paradigm that explicitly addresses those modularity problems by introducing some new programmatic constructs.So far, aspect-orientation has been mostly applied to object-oriented and procedural programming languages.

In this paper, we propose using aspect-orientation concepts in the context of workflow languages to solve the modularity problems mentioned above. We introduce aspect-oriented workflow languages, which provide concepts for crosscutting modularity such as aspects, join points, pointcuts, and advice. Aspect-oriented workflow languages support a concern-based decomposition of workflow process specifications instead of the perspective-based decomposition: The business logic, as being the main concern in workflows, can be specified in a modular way within a workflow process module and crosscutting concerns can be specified in a modular way using workflow aspects. Like workflow processes, the specification of a workflow aspect involves the different workflow perspectives.

The concepts that will be presented in this paper are not just theoretical ideas. They have been already validated in AO4BPEL [3, 6], which is an aspect-oriented extension to BPEL. The prototype implementation of AO4BPEL [1] can be considered as a proof-of-concept for aspect-oriented workflow languages. As BPEL is a domain-specific, this paper tries to generalize those ideas to other general-purpose workflow languages.

The remainder of this paper is organized as follows. In Section 2, we give an overview of the WFMC workflow meta model and workflow languages. In Section 3, we illustrate through examples the modularity problems of workflow languages with respect to crosscutting concerns and changes. In Section 4, we present the main concepts of aspect-oriented workflow languages and show how they are incorporated in AO4BPEL. In Section 5, we report on related work and Section 6 concludes the paper.

[1] This includes activities, variables, transitions, participant and application declarations.

2 Workflow Languages

We introduce the WFMC meta-model for process definition to give an overview of the main concepts encountered in workflow languages. Then, we present a graph-based workflow language that we will use throughout this paper.

2.1 The WFMC Workflow Meta-model

The Workflow Management Coalition (WFMC) has defined a workflow process definition meta-model, which provides a common method to access and describe workflow definitions in a vendor independent way. This meta-model covers the different workflow perspectives mentioned earlier. The different entities in this meta-model are explained in the following.

Workflow Process Activity. A process activity comprises a logical, self-contained unit of work that will be processed using a combination of a resource and a software application. An activity could be either atomic or composite (the functional perspective).

Transition Information. Activities are connected to one another via transitions. The transition information describes possible transitions between activities and the conditions which enable/disable them during workflow execution (behavioral perspective).

Workflow Relevant Data. This defines the workflow data, which can be either used to maintain decision data (e.g., to evaluate conditions) or to held the input and output data of the activities. Activities can also access environment and system data (informational perspective).

Workflow Participant Specification. This describes the resources (humans and/or software applications) that perform the process activities. There is a participant assignment attribute of the activity, which associates it to the set of resources that may be allocated to it (organizational perspective).

Workflow Application Declaration. This includes the applications and tools that are invoked by the workflow engine to execute the process activities (the activity implementation). The application assignment attribute of an activity associates it with the application that executes it (operational perspective).

2.2 A Basic Workflow Language

There are several classes of workflow languages: graph-based, Petri-net based, state and activity charts, and script or workflow programming languages [21]. We will use a the simple graph-based workflow language presented in [25] to illustrate the issues of crosscutting concerns in workflow process specifications. We chose this graph-based language because it illustrates well the activity graph, which is the basis for the discussion on crosscutting concerns in Section 3. However, the observations that we made apply also to the other kinds of workflow languages. Moreover, this language allows us to explain the concepts of aspect-oriented workflow languages independently of BPEL.

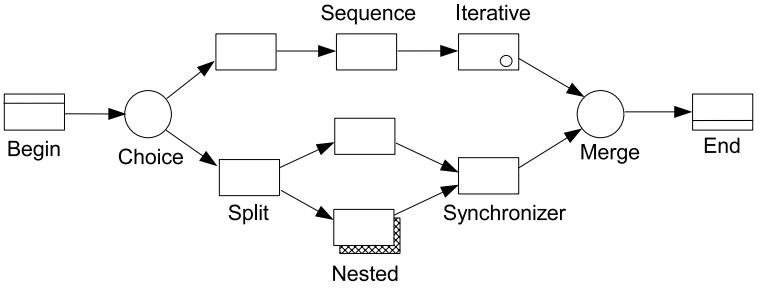


Fig. 1. Process modeling structures

This language provides two kinds of objects: *nodes* and *transitions* to model processes. There are two classes of nodes: *activity* nodes that represent the workflow activities and *choice/merge* nodes. Activity nodes are represented by a rectangle and choice/merge nodes are represented by a circle. A transition links two nodes in the graph and is represented by a directed edge. A transition shows the execution order of its source node and destination node. A workflow process can be modeled by connecting nodes with transitions into a directed acyclic graph using the modeling structures shown in Figure 1. This language supports the basic workflow control patterns [30]. In addition, it provides other generic workflow modeling constructs such as *iteration* and *nesting*.

3 Crosscutting Concerns in Workflows

We consider two workflow processes: one for order processing and one for call for bids (CFB for short) processing. Both processes are modeled in Figure 2 using the workflow language presented in Section 2. The order processing workflow (Wf1) is executed whenever a customer places an order. It first checks if the requested product is available. If not, the order is rejected and the process terminates. Otherwise, the *price calculation* activity executes; and after its successful completion the workflow proceeds with the *request confirmation* activity.

The second workflow process (Wf2) is executed whenever a call for bids is received. It also checks for product availability. If the product is available, an offer is made and sent to the party that published the CFB. Otherwise, the call for bids data is stored and the process terminates. The activity *make an offer* is a composite activity, which includes the price calculation activity.

We will elaborate on *data validation* and *security* as examples of crosscutting concerns in workflow specifications. The point that we intend to convey here is that workflow designers need in many situations to add data validation and security activities, which cut across several workflow processes and lead to modularity problems.

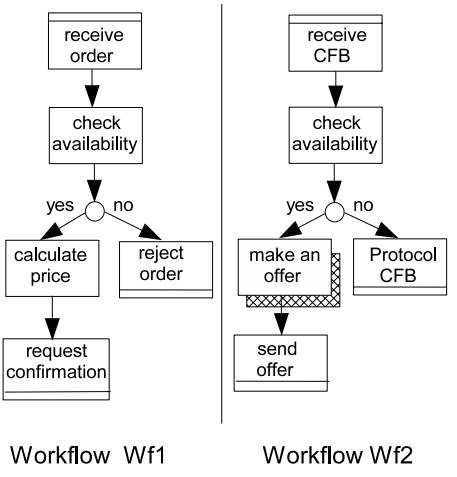


Fig. 2. Two Workflow processes

3.1 Data Validation

According to the WFMC, workflow data is divided in three kinds: *workflow control data*, *workflow relevant data*, and *workflow application data*. In the following, we focus on workflow application data because the other two types of workflow data are generally not accessible to ordinary users. Hence, when we speak of data, workflow application data is meant.

Business processes involve several business partners such as customers, suppliers, financial institutions, etc. Consequently, the organization that deploys a workflow process receives data from external sources outside its domain of control. Such data might be invalid and requires *semantic* or *syntactic* validation.

The order data is semantically invalid if e.g., the customer specifies a non-existing *product ID*. The call for bids data is syntactically invalid if it does not match some XML type specification (e.g., when this data is received from an external broker).

Some commercial workflow products such IBM MQ Workflow [15] and BizTalk Server [22] can be used with tools that support data validation. The *WebSphere Data Interchange* tool supports several data validation and mapping operations. The BizTalk Server provides data validation by verifying each instance of a document against a specification. With those tools data validation activities are added to the process without showing up in the workflow process. However, even with those tools there are still cases where adding data validation activities to the process specifications is necessary, e.g., for semantic data validation.

In order to insure the validity of order data and CFB data in the order and CFB processing workflows, the programmer has to add data validation activities to those workflows as shown in Figure 3, where those activities are represented by small rectangles dashed with vertical lines. The data validation activities are composite activities, which validate the order or the CFB and if it is invalid they send a fault and terminate the process because there is no need to proceed with the other activities in *Wf1* and *Wf2* when the order or CFB is invalid.

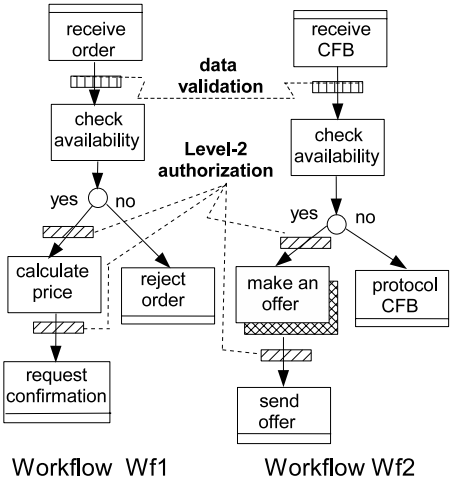


Fig. 3. Crosscutting concerns in workflow processes

We observe that the data validation activities are spread across the workflow processes Wf1 and Wf2. If the activity *receive order* for instance occurs several times in one process or in many processes then the same activity for validating the order data would have to be added after all those occurrences. In addition, the resulting workflow processes address not only the business logic concern but also the data validation concern. When we look at the workflow process specifications of Wf1 and Wf2 (cf. left side of Figure 4) these observations are confirmed. In fact, the workflow constructs (activities, variables, transitions, participant and application declarations) that implement the data validation concern are *scattered* across many workflow processes and are not localized in a separate module although they all address the data validation concern. Moreover, the resulting workflow process specifications are *tangled*, i.e., business logic activities are intertwined with activities that address other concerns. This leads to complex workflow specifications, which are hard to understand, to maintain, and to change.

3.2 Security

In [23], the authors differentiate three levels of security in workflows. *Level 1 security (database-level)* insures that each activity is performed by an authorized subject i.e., the subject is granted access to the underlying data objects of the activity. *Level 2 security (workflow-level)* insures that access to that data is granted only during the execution of the respective activity. *Level 3 security (application-level)* focuses on application-specific security requirements.

For the purpose of this discussion, we concentrate on level 2 and level 3 security. In fact, some security requirements in workflows could be supported in layers below the workflow level, e.g., in the database, in the messaging infrastructure, or in the application that implements some activity. However, workflow-level and

application-level security requirements have to be addressed in the workflow process because they require knowledge about the workflow execution state and the application semantics that is unavailable in the underlying layers. In that case, it becomes necessary to have security activities in the workflow process.

Some commercial workflow management systems can be used with tools that provide advanced security support. For example, the *WebSphere MQ Extended Security Edition* is an add-on to the IBM MQ Workflow [15] that such workflow data protection by signing and/or encrypting messages. In such case, security activities are added to the workflow without showing up in the workflow specification. However, this tool does not support workflow-level and application-level security requirements, which can only be supported by adding security activities explicitly to the workflow process. So, there are always cases, where adding security activities the workflow process is unavoidable.

For example, it might be necessary to insure the integrity and confidentiality of application data at the workflow level if this feature is not provided by the underlying layers. Assume that the price calculation activity in the order processing workflow is performed in an external subsidiary by calling some application via CORBA. Then, the subsidiary replaces that application with a Web Service that uses WS-Security. If the workflow management system hosting the order process does not support secure Web Service calls it becomes necessary to add activities to the workflow to secure the data before and also after the price calculation activity because the response of the price calculation Web Service might be secured. In this case, one would have activities in the workflow processes for e.g., encrypting and signing data.

Another well-known example of workflow-level security requirements is that authorized subjects should gain access to the required data only during the activity execution (and not before or after its execution). This means that the security system should authorize the subjects in synchronization with the progress of the workflow. Several workflow management systems do not support such temporal authorization [31]. To support this authorization model in those systems, the workflow designer has to add activities, which notify the authorization system before and after the execution of the process activities as illustrated by the small rectangle dashed with diagonal lines in Fig. 3.

Like for data validation, the security activities are *scattered* across *Wf1* and *Wf2* and the resulting workflow process specifications suffer from *tangling*.

When the workflow designer needs to add support for crosscutting concerns such as data validation and security, he/she not only has to modify the activity graph by adding new activities as shown in Fig. 3. There are other workflow constructs that should be added to the respective workflow process specifications such as variables, participant declarations, application declarations, etc. In lack of a module concept for modularizing the workflow constructs that belong to a crosscutting concern, the scattering problem spans all those constructs and is not only restricted to activities.

One might argue that the data validation and security could be implemented as part of the activity implementation in separate modules; and thus would

not be referenced at the level of workflow process model. This means that the activity implementation would consist of a core business object that is supported by well-modularized help objects for data validation and security. Such solution assumes a green-field engineering scenario and it is also not feasible if the activity implementation is unavailable (e.g., only binary code) or unaccessible (a remote application e.g., an external Web Service).

3.3 Crosscutting Changes and Change Modularity

When the workflow programmer adds support for security or data validation to the workflows Wf1 and Wf2, he/she has to do with a crosscutting change that spans several processes. In the following, we focus on the modularity of expressing workflow changes and crosscutting workflow changes in particular. To study how changes are expressed, we take as example the adaptive workflow management systems ADEPT [24] and WASA [32].

We observe that ADEPT and WASA do not provide a module concept to express a change as a first-class entity. They just provide a set of dynamic change operations e.g., for adding or deleting activities and edges to/from the activity graph. Therefore, understanding what the change is about can only be done implicitly by looking at the change operations and the API calls that were used to introduce it, or by comparing two workflow schemes.

In addition, we argue that a workflow change does not only encompass the insertion or removal of workflow activities and control edges. Workflow changes span and affect the different workflow perspectives. They may also require new participant declarations, data declarations, and application declarations to be added to the workflow schema.

The previous discussion motivates the need for a module that encapsulates all workflow constructs that belong to a given change. In presence of such a module, changes will be expressed as first-class entities. Thus, they can be understood and managed more easily. In particular, to undo temporary changes one only has to deactivate the change module. The need for modular change expression is even stronger in the case of crosscutting changes, which affect several places in various workflows.

4 Aspect-Oriented Workflow Languages

Aspect-oriented workflow languages are workflow languages that provide constructs typically found in aspect-oriented programming languages such as *aspect*, *pointcut*, and *advice*.

4.1 Aspect-Oriented Programming

Aspect-Oriented Programming [20] introduces a new unit of modularity called aspect aimed at modularizing crosscutting concerns in complex systems. In this paper, we use the terminology of AspectJ [19], the most mature AOP language today. It introduces three key concepts: *join points*, *pointcuts* and *advice*. Join

points are points in the execution of a program. In object-oriented programs, examples of join points are method calls, constructor calls, field read/write, etc. A pointcut is a means to identify related join points. It can be thought of as a predicate on attributes of join points. One can select related method execution points, e.g., by the type of their parameters or return values, by pattern matching, etc. The crosscutting functionality at join points identified by a pointcut is specified in an advice. The advice code is executed when a join point in the set identified by a pointcut is reached. It may be executed before, after, or instead of, the join point at hand, corresponding to *before*, *after* and *around* advice. With the around advice the aspect can control the execution of the original join point: It can integrate the further execution of the intercepted join point in the middle of some other code to be executed around it.

An aspect module consists, in general, of several pointcut definitions and advice associated to them. In addition, it may define state and methods which in turn can be used within the advice code. Advice code also has access to the execution context at join points that trigger its execution.

4.2 Concern-Based Decomposition of Workflow Specifications

The aspect in aspect-oriented workflow languages is a module that encapsulates a non-functional crosscutting concern. This breaks with the *tyranny of the dominant decomposition* [28] and enables a multi-dimensional and concern-based decomposition along the process dimension and the aspect dimension. Hence, the workflow constructs that belong to some concern are specified in one module: The workflow logic is encapsulated in a *process module* and non-functional concerns are encapsulated in *aspect modules* as shown in Fig. 4. The left hand side shows how workflow specification is done in current workflow languages. The right hand side of the figure shows how workflow specification can be done in a more modular way by using workflow aspects to encapsulate crosscutting concerns such as security and data validation.

AO4BPEL [3,6] is an aspect-oriented extension to (BPEL) [10], which provides a solution to the lack of appropriate means for the modularization of crosscutting concerns and for supporting dynamic changes in BPEL. AO4BPEL can be considered as the proof-of-concept for aspect-oriented workflow languages. We used AO4BPEL aspects to modularize various concerns such as measurement of activity execution time, auditing data collection, security, and reliable messaging [3,6,5,7]. In the following, we will present the main concepts of aspect-oriented workflow languages and will use AO4BPEL to illustrate them.

Aspects. An aspect contains several pointcut and advice declarations. In addition, it defines the activities, transitions, data, participants, and application declarations that belong to a crosscutting concern as shown in Fig. 4. This Figure illustrates how security and data validation are modularized by using aspects. Each aspect defines appropriate pointcuts to select join points in the base workflow processes *Wf1* and *Wf2* where the data validation and security advices will be executed. The left side of this figure shows how workflow specification is

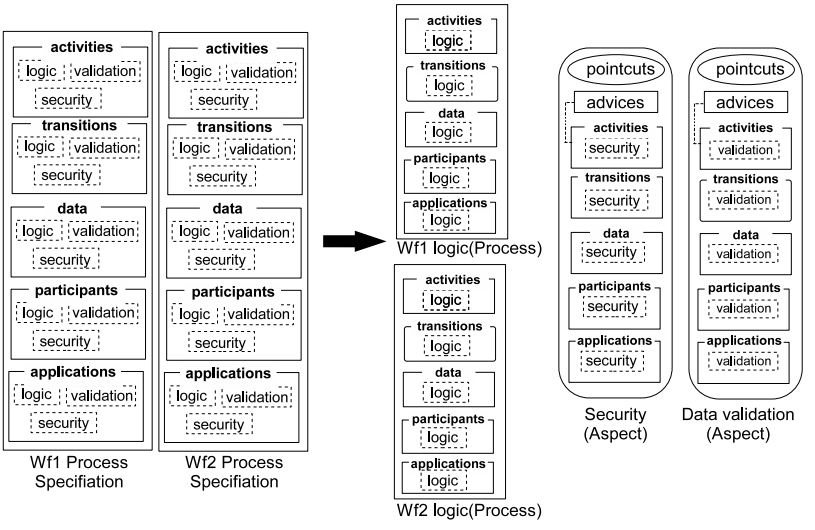


Fig. 4. Concern-based decomposition of workflow specifications

done in current workflow languages and the right side shows how it is done in aspect-oriented workflow languages.

An aspect-oriented workflow-based application consists of *workflow processes* capturing the core logic of the business processes and *aspects* capturing the other crosscutting concerns. The management staff would typically look only at the workflow processes (BPM view). The IT staff, which deals with the technical concerns would look at the workflow processes and the aspects (WFM view).

Aspects provide a cross-process view on how a certain concern is handled in several workflows. In this way, if the workflow programmer needs to understand or modify for example a security policy then he/she has to change one module only (namely the security aspect). As shown in Fig. 4, there is one security aspect that captures the security code of the two workflow processes Wf1 and Wf2.

Aspects can also be used for expressing changes and crosscutting changes as first-class entities in a modular way. The pointcut of an aspect specifies all places that are affected by the change. The advice and the declarations for data, participants, and applications explain what the change is about. Moreover, with aspects a change could be switched on/off more flexibly by deploying/undeploying the respective aspect.

In AO4BPEL, aspects are written in XML syntax in separate files. In addition to pointcut and advice declarations, an aspect can also define the partner links, the variables, and the handlers (fault handlers, event handlers, compensation handlers) that are necessary to implement a crosscutting concern.

Join points and pointcuts. In our approach, aspect activities are defined separately from process activities. Therefore, some means are needed to specify when aspect activities should be executed with respect to the workflow process.

The *join point* concept refers to a point in the execution of a workflow process. Join points could be coarse-grained e.g., the execution of an activity [3] or fine-grained e.g., internal points in the execution of an activity [5]. To simplify things, consider the *execution of a workflow activity as a join point.*

The pointcut language provides means to select a set of related join points, which could span different processes. A powerful pointcut language should provide constructs that enable a flexible join point selection according to the various workflow perspectives for two reasons. First, this would increase the expressiveness of the pointcut language and allow a pointcut to select all join points where some activity a that is reused in several workflows is executed (functional perspective), where some activity a is executed after another activity b (behavioral perspective), where an activity is executed by a certain participant (organizational perspective), where an activity modifies a certain workflow variable (informational perspective), where an activity is executed with help of an external application (operational perspective), etc. Second, some crosscutting concerns could be captured in a more natural and easy way when the pointcut language supports certain workflow perspectives. For instance, the join points where *data persistence* is needed could be selected in more natural manner when using a pointcut language that supports the informational perspective. Join points where security is needed could be expressed more easily with a pointcut language that support the organizational and the operational perspectives.

The pointcut language of an aspect-oriented workflow language can operate either on a text-based workflow specification (e.g., BPEL or XPDL) or on a graphical representation of the workflow (we call the latter *visual pointcuts*).

AO4BPEL supports *workflow-level join points*, which correspond to the execution of a BPEL activity as well as *interpretation-level join points* [5,4], which correspond to internal steps in the interpretation of an activity capturing for instance the point where a SOAP message of a messaging activity has been generated. As BPEL process specifications are XML documents, we use XPath [8] as pointcut language in AO4BPEL. For example, one could define a pointcut selecting all interactions with a certain partner Web Service by using the attribute *partnerLink* of the activities *invoke*, *reply*, and *receive*.

Advice. After we selected a set of related join points with a pointcut, we specify with the *advice* what should happen at those join points. *The advice is basically an activity that implements some crosscutting concern.* The advice activity is executed whenever the respective pointcut matches an activity during the execution of the workflow process.

The semantics of the advice is to replace each join point activity by a composite activity that may contain the join point activity in addition to activities belonging to the crosscutting concern. If the join point activity is source or target of transitions, the composite activity becomes the source or target of those transitions. The advice is a self-contained activity, i.e., no transitions are allowed between the advice and the other activities of the workflow process except the join point. In the around advice, it is possible to integrate the execution of the

join point in the middle of the advice using the *proceed* activity. The latter acts as a place holder for the current join point.

The advice activity can be executed before, after, or instead of the join points selected by the pointcut depending on the advice type. Other orders of execution between the join point activity and the advice are also possible according to the variety of workflow control patterns [30]. E.g., it is possible to define a parallel advice, which executes concurrently to the join point.

The advice activity needs in many cases to access the context of the join point activity. For example, a data validation advice for the workflow processes of Fig. 3 needs to access the output data of the join point activities *receive order* and *receive CFB*. The pointcut and advice languages should provide constructs to access the join point context in a generic way because a pointcut could select more than one activity, i.e., the advice cannot refer to the output variable of a join point by its name. The advice language should also provide *context collection* and *reflection* constructs to get the data and the other attributes of the current join point activity.

According to the *aspect deployment* strategy, the advice is either applied to all instances of the workflow process (process-level deployment) or to some workflow instances only (instance-level deployment). The pointcut language should provide appropriate constructs to support instance-level aspect deployment.

In AO4BPEL, the advice is a BPEL activity that can be executed before, after, or around of the selected join points. Some special constructs can be used in the advice such as the *proceed* activity and the *reflection* and *context collection* variables. For example, the reflective variable *ThisJPActivity* provides information about the current join point such as activity name, activity type, etc. The reflective variables *ThisJPInVariable* and *ThisJPOutVariable* refer to the input and output variables of the join point activity. AO4BPEL currently supports both process-level deployment and instance-level deployment. It also provides constructs to specify the execution order of aspects that are triggered at the same join point.

Aspect/Process Composition. A mechanism is required for the integration of aspects with the workflow processes. In aspect-oriented programming languages, this mechanism is called *weaving*. The aspects and the base application are integrated at compile time in *static weaving* approaches and at runtime in *dynamic weaving* approaches.

In aspect-oriented workflow languages, the composition of aspects with the workflow processes can be considered as a transformation of the workflow activity graphs [13], whereby activities that are matched by a pointcut are replaced by a new composite activity that contains the advice and the original join point with appropriate transitions. This transformation can be performed physically or logically, which results in two possible aspect/process composition approaches:

Process transformation: A tool is needed to merge the worfklow processes and the aspects (physical transformation) before deploying the resulting process. This tool performs the reverse work of what is shown in Fig. 4. The benefit of

this approach is that the workflow engine does not need to be modified, which makes the aspect-oriented extension independent of any specific engine. Moreover, one would have two versions of the workflow process (one before and one after weaving). The version before weaving is useful to understand the business process by abstracting away from technical details. The version after weaving is required to exactly understand and predict the workflow process that will be executed, which is necessary for workflow auditing and workflow log mining as well as for debugging.

Aspect-aware engine: The workflow engine should be modified to check for aspects before and after executing each activity (logical transformation). In this approach, the workflow process is not modified. The implementation of an aspect-aware engine is more complex than a process transformation tool and it makes our proposal dependent on a particular engine. However, it supports the dynamic composition of aspects and workflows, which could be very useful to enable flexible and adaptable workflows [14]. It makes it possible to deploy aspects that change running workflow processes. These aspects could be switched on and off flexibly without need to edit the workflow process definition and redeploy the process.

In addition to transforming the activity graph, there are common tasks that should be performed by the composition mechanism independently of the process/aspect composition approach. For example, aspect-local declarations for data, participants, and applications should be handled similarly to the declarations in workflow processes e.g., by adding those declarations to the workflow processes that are affected by the aspect. Another task is *pointcut matching* i.e., to decide whether some activity is matched by a pointcut. In addition, the composition mechanism has to resolve all context collection and reflection constructs (e.g., to get the input data of the join point) and special constructs (e.g., using the proceed activity) used in the advice.

Our implementation of AO4BPEL [1] is based on the aspect-aware engine approach, which supports in a better manner the dynamic composition of workflow aspects and workflow processes. Hence, it makes BPEL processes more flexible and dynamically adaptable [3]. Moreover, pointcuts that depend on runtime data and instance-based aspect deployment could be supported without adding hook activities to the workflow process to evaluate such runtime conditions.

We extended the IBM BPWS4J engine [16] with support for aspects and dynamic aspect/process composition. The activity life cycle [11] of BPWS4J was modified to check before and after each activity if there is an aspect with a pointcut matching the current activity. If an aspect is found, the engine executes the respective advice. The overhead of these local checks is negligible when compared with the cost of Web Service interactions via the network as confirmed by the performance measurements that we presented in [6].

4.3 Examples of Workflow Aspects

In the following, we show how the crosscutting concerns mentioned in Section 3 can be modularized using workflow aspects.

The data validation aspect defines two pointcuts selecting respectively all occurrences of the activity *receive order* and the activity *receive call for bids*. The data validation aspect defines two advice activities respectively for validating an order and a call for bids.

Fig. 5. A data validation aspect in AO4BPEL

Fig. 5 shows an excerpt of the data validation aspect in AO4BPEL with the assumption that the workflow processes Wf1 and Wf2 are specified in BPEL and deployed on our AO4BPEL engine. The *data validation* aspect defines a partner link to the validation Web Service and four variables for the input and output of the *invoke* activities that call the operations *validateOrder* and *validateCFB* on that Web Service. The pointcut of this aspect is an XPath expression that selects all *receive* activities matching the operations *placeOrder* and *placeCFB* in any deployed BPEL process. The after advice, which is associated with this pointcut is a *sequence* activity that contains calls the data validation Web Service and if the data of an order or a CFB is invalid (*switch* activity) the process terminates (using the *terminate* activity).

In this data validation aspect, the pointcut selects two activities; each of them could be reused in many workflows e.g., a third workflow *Wf3* for processing orders that are placed after an accepted bid could contain another occurrence of the activity *receive order*.

The level-2 authorization concern could be modularized using an aspect that defines a pointcut selecting the activities *price calculation* and *make an offer*. This pointcut would be associated with an around advice, which calls the

authorization system before and after each of the selected activities. The join point activities are executed between the two calls to the authorization system by using the *proceed* activity. The advice needs to pass reflective information about the current join point activity to the authorization system. In AO4BPEL, this is achieved by using the special AO4BPEL variable *ThisJPActivity*.

```
<aspect name="Level2Authorization">
 <partnerLinks>
   <partnerLink name="AuthorizationWS" partnerLinkType="AuthorizationPLT"
                myRole="Process" partnerRole="AuthorizationSystem"/>
 </partnerLinks>
 <variables>
   <variable  name="grantAccessRequest" messageType="grantAccessInput"/>
   <variable  name="revokeAccessRequest" messageType="revokeAccessInput"/>
 </variables>
 <pointcutandadvice>
  <pointcut name="selected activities">
     //invoke[@operation="processPayment"] | //invoke[@operation="makeOffer"]
  </pointcut>
  <advice type="around">
   <sequence>
     <assign><copy>
               <from variable="ThisJPActivity" part="name"/>
               <to variable="grantAccessRequest" part="activityName"/>
             </copy><copy>
              <from variable="ThisJPActivity" part="process"/>
               <to variable="grantAccessRequest" part="processName"/>
             </copy></assign>
    <invoke partnerLink="AuthorizationWS" portType="AuthorizationPT"
            operation="grantAccess"  inputVariable="grantAccessRequest"/>
    <proceed>
    <assign>...</assign>
    <invoke partnerLink="AuthorizationWS" portType="AuthorizationPT"
            operation="revokeAccess"  inputVariable="revokeAccessRequest"/>
   </sequence>
  </advice>
 </pointcutandadvice>
</aspect>
```

Listing 1. A level-2 authorization aspect in AO4BPEL

This level-2 authorization concern is a typical crosscutting workflow concern in general-purpose workflow languages. However, it does not make sense in the context of BPEL because BPEL does not support human tasks, role resolution, and task-list management. Nevertheless, there are already some proposals for extending BPEL with human tasks such as BPEL4People [17]. In listing 1, we show the level-2 authorization aspect using the syntax of AO4BPEL, the only available aspect-oriented workflow language to date.

5 Related Work

To our best knowledge, the idea of introducing aspect-oriented software development concepts to workflow languages is a novel one. AO4BPEL is the first aspect-oriented workflow language. We have presented the AO4BPEL language

in [3,6]. In [6], we discuss several issues with regard to our approach such understandability, debugging, maintenance, static analysis, etc. In the current paper, we abstract away from BPEL and generalize the ideas of AO4BPEL to other workflow languages.

There are some other works such as [2], [26], and [29], which have used aspects in the context of workflow management.

In [2], the authors use AspectJ for the dynamic evolution of workflow instances. They use aspects in the object-oriented implementation of the workflow management system. They show different kinds of control flow adaptations such as insertion of a new activity to the process, replacement of an activity by another, etc. Unlike our proposal, the authors of [2] apply AOP at the implementation level and not at the workflow specification level.

The work presented in [26] calls for the specification of workflows according to the various workflow perspectives. In that paper, Schmidt and Assmann argue that the different perspectives can be merged together using an aspect weaver, but they do not present any implementation and do not give any details on how an executable workflow process could be generated from the different specifications of those perspectives. In our approach, we merge the specifications of the different concerns that are modularized in *worklfow aspects* (and not the different perspectives).

In [29], the authors propose combining business process management and AOP to enable flexible and dynamic business processes. They weave a generalized process with participant process aspects, which are process steps that can be included in a business process to customize it for execution by a particular resource. Their implementation uses AspectJ and Java.

The papers mentioned so far do not introduce any aspect-oriented concepts (pointcut, join points, advice) to the workflow language. They use aspect-oriented languages at the workflow implementation level only, unlike our proposal, which makes aspect-oriented extensions accessible to the workflow designer. Thus, we provide language means to modularize crosscutting concerns.

In [13], the authors present a set of transformation operations on workflow graphs that range from basic transformations such as encapsulation in a sequence and moving join and splits to complex transformations. This work provides a good basis for aspect/process composition using a process transformation tool.

6 Conclusion

In this paper, we illustrated the modularity problems in workflow languages with respect to crosscutting concerns. To solve those problems, we proposed a concern-based decomposition of workflow process specifications. We also introduced aspect-oriented workflow languages, which support a more modular workflow process specification and a better separation of concerns by providing new language constructs such as aspect, pointcut, and advice. The idea of aspect-oriented workflow languages is not just theoretical but it was already implemented in AO4BPEL. As BPEL is a domain-specific workflow language,

this paper tries to trigger thoughts and research efforts on the application of aspect-oriented concepts to other more typical workflow languages.

Acknowledgements. We thank Prof. Frank Leymann and Michael Haupt for the helpful comments on an early draft of this paper.

References

1. Anis Charfi and Mira Mezini. Aspect-Oriented Web Service Composition in AO4BPEL, Demo at the International Conference on Aspect-Oriented Software Development (AOSD). http://aosd.net/2006/demos/index.php, March 2006.
2. Boris Bachmendo and Rainer Unland. Aspect-based workflow evolution. In *Workshop on Aspect-Oriented Programming and Separation of Concerns*, August 2001.
3. Anis Charfi and Mira Mezini. Aspect-Oriented Web Service Composition with AO4BPEL. In *Proceedings of the European Conference on Web Services (ECOWS)*, volume 3250 of *LNCS*, pages 168–182. Springer, September 2004.
4. Anis Charfi and Mira Mezini. An Aspect-based Process Container for BPEL. In *Proceedings of the 1st Workshop on Aspect-Oriented Middleware Development (AOMD)*, November 2005.
5. Anis Charfi and Mira Mezini. Using Aspects for Security Engineering of Web Service Compositions. In *Proceedings of the IEEE International Conference on Web Services (ICWS), Volume I*, pages 59–66. IEEE Computer Society, July 2005.
6. Anis Charfi and Mira Mezini. AO4BPEL: An Aspect-Oriented Extension to BPEL. *World Wide Web Journal: Recent Advances on Web Services (special issue), to appear*, 2006.
7. Anis Charfi, Benjamin Schmeling, and Mira Mezini. Reliable messaging in bpel processes. In *Proceedings of the 3rd IEEE International Conference on Web Services (ICWS)*, September 2006.
8. J. Clark and S. DeRose. XML Path Language (XPath)Version 1.0. W3C Recommendation 16 November 1999.
9. AOSD Community. Aspect-Oriented Software Development Community and Conference. http://www.aosd.net.
10. F. Curbera, Y. Goland, J. Klein, et al. Business Process Execution Language for Web Services (BPEL4WS) Version 1.1, May 2003.
11. Francisco Curbera, Rania Khalaf, William Nagy, and Sanjiva Weerawarana. Implementing BPEL4WS: The Architecture of a BPEL4WS Implementation. In *GGF 10 Workshop on Workflow in Grid Systems*, Berlin, Germany, March 2004.
12. Bill Curtis, Marc I. Kellner, and Jim Over. Process Modeling. *Commun. ACM*, 35(9):75–90, 1992.
13. Johann Eder, Wolfgang Gruber, and Horst Pichler. Transforming Workflow Graphs. In *Proceedings of the 1st International Conference on Interoperability of Enterprise Software and Applications (INTEROP-ESA)*, February 2005.
14. Yanbo Han, Amit Sheth, and Christoph Bussler. A Taxonomy of Adaptive Workflow Management. In *Workshop "Towards Adaptive Workflow Systems" in conjunction with CSCW*, November 1998.
15. IBM. Websphere MQ Workflow. http://www-306.ibm.com/software/integration/wmqwf/.
16. IBM. The BPEL4WS Java Run Time, August 2002.
17. IBM and SAP. WS-BPEL Extension for People - BPEL4People, July 2005.

18. Stefan Jablonski. MOBILE: A Modular Workflow Model and Architecture. In *Proceedings of the Fourth International Working Conference on Dynamic Modelling and Information Systems*, September 1994.
19. G. Kiczales, E. Hilsdale, J. Hugunin, et al. An Overview of AspectJ. In *Proceedings of the 15th European Conference on Object-Oriented Programming (ECOOP)*, volume 2072 of *LNCS*, pages 327–353. Springer, 2001.
20. G. Kiczales, J. Lamping, A. Mendhekar, et al. Aspect-Oriented Programming. In *Proceedings of the 11th European Conference on Object-Oriented Programming (ECOOP)*, volume 1241 of *LNCS*, pages 220–242. Springer, 1997.
21. Mathias Weske and Gottfried Vossen. *Handbook on Architectures of Information Systems*, chapter Workflow Languages, pages 359–379. Springer , Berlin, 1998.
22. Microsoft. BizTalk Server 2004 Architecture White Paper. december 2004.
23. Martin S. Olivier, Reind P. van de Riet, and Ehud Gudes. Specifying Application-Level Security in Workflow Systems. In *Proceedings of the 9th Workshop on Database and Expert Systems Applications (DEXA)*, pages 346–351, August 1998.
24. Manfred Reichert and Peter Dadam. ADEPT flex -Supporting Dynamic Changes of Workflows Without Losing Control. *Journal of Intelligent Information Systems*, 10(2):93–129, 1998.
25. Wasim Sadiq and Maria E. Orlowska. On Business Process Model Transformations. In *The 19th International Conference on Conceptual Modeling (ER)*, volume 1920 of *LNCS*, pages 267–280. Springer, October 2000.
26. R. Schmidt and Uwe Assmann. Extending Aspect-Oriented-Programming in order to flexibly support Workflows. In *Proceedings of the Aspect-Oriented Programming Workshop in conjunction with ICSE 98*, April 1998.
27. Stefan Jablonski and Christoph Bussler. *Workflow Management: Modeling Concepts, Architecture and Implementation.* International Thomson Computer Press, London, UK, 1996.
28. P. Tarr, H. Ossher, W. Harrison, and S. M. Sutton. N Degrees of Separation: Multi-dimensional Separation of Concerns. In *Proceedings of the 21st International Conference on Software Engineering (ICSE)*, pages 107–119. ACM Press, 1999.
29. Simon Thompson and Brian Odgers. Aspect-Oriented Process Engineering. In *Proceedings of the Workshop on Object-Oriented Technology in conjunction with ECOOP 99*, June 1999.
30. Wil M. P. van der Aalst, Arthur H. M. ter Hofstede, Bartek Kiepuszewski, and Alistair P. Barros. Workflow Patterns. *Distributed and Parallel Databases*, 14(1):5–51, 2003.
31. Vijay Atluri. Security for Workflow Systems. *Information Security Technical Report*, 6(2):59–68, 2001.
32. Mathias Weske. Flexible Modeling and Execution of Workflow Activities. In *Proc. of the 31st Hawaii International Conference on System Sciences (HICSS)-Volume 7*, pages 713–723. IEEE Computer Society, 1998.

A Portable Approach to Exception Handling in Workflow Management Systems

Carlo Combi[1], Florian Daniel[2], and Giuseppe Pozzi[2]

[1] Università degli Studi di Verona, Strada le Grazie 15, 37134 Verona, Italy
carlo.combi@univr.it
[2] Politecnico di Milano, P.za Leonardo da Vinci 32, 20133 Milano, Italy
{florian.daniel, giuseppe.pozzi}@polimi.it

Abstract. Although the efforts from the Workflow Management Coalition (WfMC) led to the definition of a standard process definition language (XPDL), there is still no standard for the definition of expected exceptions in workflows. Yet, the very few Workflow Management Systems (WfMS) capable of managing exceptions, provide a proprietary exception handling unit, preventing workflow exception definitions from being portable from one system to another one.

In this paper, we show how generic process definitions based on XPDL can be seamlessly enriched with standard-conform exception handling constructs, starting from a high-level event-condition-action language. We further introduce a suitable rule compiler, enabling to yield portable process and exception definitions in a fully automated way.

1 Introduction

Workflows are activities involving the coordinated execution of atomic activities (*tasks*) performed by different processing entities (*agents*). The process model (*schema*) of a workflow includes the description of its component tasks, of the flow of execution, of *routing nodes* (*connectors*) to activate parallel (*AND split*) and conditional executions (*OR split*), and to resynchronize executions (*AND join*, *OR join*). Instances of schemas are known as *cases*. Workflow management systems (WfMS) support the automatic execution of workflows [1,2].

The schema of a workflow considers the *normal* flow of execution, where cases evolve through predefined execution paths according to the variables of every case. Occasionally, during the execution of a case, some exceptional situations may occur, possibly deviating the execution path from the set of normal execution paths. The semantics of these exceptions is not negligible and must be foreseen at workflow design time: we call these exceptions *expected exceptions* [3]. Other exceptions (e.g. power fail, network disconnection...) are not modelled at workflow design time and will no longer be considered throughout the paper.

The *Workflow Management Coalition* (WfMC [1]), whose recommendations and standards influence WfMS vendors, issues the XPDL language, which is an XML-based process definition language, aimed at obtaining portability of the process definitions among different WfMSs. While several efforts have been

R. Meersman, Z. Tari et al. (Eds.): OTM 2006, LNCS 4275, pp. 201–218, 2006.

done with respect to the normal process flow definition, and several WfMSs may share the same schema based on XPDL, very few WfMSs come with an expected exception handling unit and no standard has been achieved about exceptions.

In the following, we propose a technique to map expected exceptions inside the XPDL schema, and: i) provide WfMSs with a primordial exception management unit; ii) maintain the portability of the process definition among WfMSs, capable of processing XPDL descriptions. As reference language to define, capture, and manage exceptions we consider the Chimera-Exception language [4].

In this paper, Section 2 considers related work; Section 3 and Section 4 consider process and exception specification, introducing XPDL and Chimera-Exception. Section 5 shows how to model ECA rules in XPDL. Section 6 describes how to support expression evaluation in XPDL, used in Section 7 to map ECA rules onto process specifications. Section 8 describes some practical experiences gained. Finally, Section 9 draws some conclusions and anticipates future work.

2 Related Work

The literature covers several facets of exception handling in WfMS. Mourão et al. [5] define a suitable environment to orchestrate ad-hoc human interventions with a minimum impact on system integrity. Golani et al. [6,7] consider how to increase the flexibility of a WfMS to deal with unexpected exceptions, and propose a dynamic mechanism that allows backtracking and forward stepping at the instance level. Luo et al. [8] focus on exception-handling schemes for conflict resolution in the delivery of e-business transactions, with particular attention to support conflict resolution in cross-organizational settings. Schuschel et al. [9] exploit the functionalities of triggers defined within a WfMS to automate the creation of process definitions by planning algorithms: by replanning and dynamically adapting the process, the newly defined process is capable of reacting to unexpected events. Finally, van der Aalst et al. [10] propose a mixed approach between workgroup management systems and WfMSs,where the normal evolution of the process is managed by predefined process control structures, while exceptions are manually managed by a human agent and the case handling system assists rather than guiding him/her in doing so.

Our particular attention is focused on expected exceptions as defined by Eder et al. in [3]. To this regard, very few WfMSs include a fully fledged exception handling unit to deal with expected exceptions. An interesting approach to exception handling is provided by the OPERA prototype [11], where exceptions can be triggered by data events or by notifications from external applications: tasks and control flows are used as reactions to captured exceptions. In OPERA, as soon as an exception is detected, the process is suspended and the control is transferred to the exception handler. COSA [12] comes with the notion of trigger, defined as an event-action rule capturing events or deadline expirations, and reacting by activating a task or a (sub)process instance. InConcert [13] includes event-action triggers in its workflow model: triggering events can be process state changes (e.g., a task becomes ready for execution), external (user

defined) events, or temporal events; actions include notification of messages to agents, activation of a new process, or invocation of a user-supplied procedure. Staffware [14] and HP's Chang*e*ngine allow the definition of a special kind of task called event node (or event step), which can suspend the case execution on a given path until a defined (exceptional) event occurs, and can then activate an event-handling path in the workflow.

The WIDE project has a powerful exception handling unit [4], where exceptions are specified by means of Event-Condition-Action (ECA) rules: workflow, temporal, external, and data events may trigger actions as suspending a case or a task, starting a new case, changing database instances, or notifying messages to agents. In this paper, we adopt the formal Chimera-Exception language developed within the WIDE project.

3 Business Process Modelling and XPDL

The XPDL (*XML Process Definition Language*) is an XML file format from the WfMC: it has the same expressive power as the visual process specification language Business Process Management Notation [15] (BPMN). XPDL can be used as file format of BPMN. In this paper we shall use BPMN modelling constructs for expressing XPDL processes and activities, instead of providing unexpressive snippets of XPDL code. *Tasks* are represented as boxes, whose names describe the performed activity. *Transitions* are represented as directed arcs to connect process nodes, which, for routing purposes, may have associated a *condition* over workflow-relevant, application, or system data. Each process definition has one *start* and one *end* task, depicted by thin or thick circles, respectively. *Gateways* are represented as diamonds; condition gateways also have an associated textual condition, while `AND` gateways are diamonds that contain a + symbol.

An XPDL document is structured hierarchically, represents a *package* made of a set of process definitions, and includes package-wide workflow *participants*, *applications* and *relevant data fields*. *Process definitions* may contain references to separately defined *sub-flows*, making up part of the overall process definition, and inherit globally defined entities. Process definitions are articulated in *pools* and *swim lanes*. For instance, when modelling interactions in B2B scenarios, pools represent private business processes, while swim lanes represent the different participants performing each of the private processes. Process *activities* represent tasks and are associated to pools and lanes. Activities are connected by means of *transitions* (intra-pool flows) or *message flows* (inter-pool flows).

An Example. As process model example, we consider (Figure 1) a company trading in books with very limited editions and accepting orders by its customers via telephone; payments are by credit card only. After receiving an order, the *Sales Office* immediately checks the credit status of the customer's credit card. In case of overdraft, the order is declined. Otherwise, the *Sales Office* proceeds and checks the stock for the requested books. If all the products are available, the customer is notified of the immediate shipment, and the *Production* department

provides for shipment. If, instead, not all the requested products are available, the *Sales Office* informs the user and asks for the approval of a delayed shipment. To accelerate the overall process, the *Production* department plans the production of the missing items in parallel, regardless of the user's decision, produces them and, if no cancellation is received, ships the complete order.

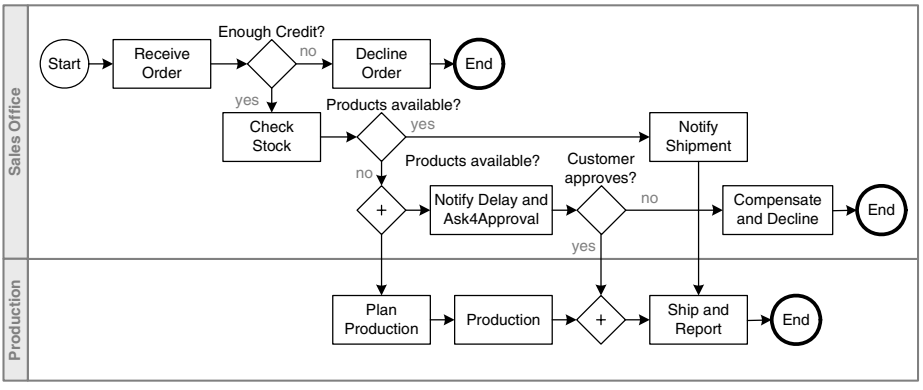


Fig. 1. Example `OrderManagement` process definition

4 Handling Workflow Exceptions

Although "anomalous" with respect to the "normal" process flow, expected exceptions are part of the semantics of the overall process and can be classified [4,16] according to *synchronicity*, *scope*, and *triggering event*.

Synchronicity: synchronous exceptions are bound to specific workflow events (i.e., task start, task completion), while asynchronous ones may occur at any time during process execution. Synchronous exceptions may be further subdivided into *localized* exceptions, caused only by the execution of one or few tasks, and *sparse* exceptions, caused at several stages during process execution.
Scope: *process-specific* exceptions may occur during the execution of one unique process; *cross-process* exceptions may occur during the execution of more processes; *global* exceptions may occur during the execution of any process.
Triggering event: events can be *data*, *temporal*, *external* or *workflow* events.

These characteristics strongly influence the way expected exceptions can be mapped onto XPDL, and thus onto process definitions. Mapping exceptions moves from Chimera-Exception definitions, as specified in the following.

4.1 The Chimera-Exception Language

Chimera-Exception builds on an object-oriented formalism. Classes are *workflow-independent* (e.g., role, agent, task) or *workflow-dependent* (e.g., workflow-relevant data fields). Rules are specified by event-condition-action constructs.

Events. Rules can be triggered by the following events: *data events* (changes of the content of workflow-relevant data or within the underlying data source, by `create`, `modify` or `delete` statements); *temporal events*, which can be (i) *instant events* expressed through the `@`-symbol (e.g. `@timestamp ‘‘December 15th, 2005’’`), (ii) *periodic events* defined by the `during` keyword (`1/days during weeks` denotes the first day of every week [17]), (iii) *interval events* expressed as `elapsed` *duration* `since` *instant*, (e.g. `elapsed (interval 1 day) since caseStart`); *external events* (recognized by means of the `raise` primitive, which – when an external event occurs – provides the name of the triggering event and suitable parameters); or *workflow events* (start and completion of tasks and cases by the primitives `caseStart`, `caseEnd`, `taskStart`, and `taskEnd`).

Conditions. A condition consists of predicates that inspect the content of the database. The predicate `occurred` enables to refer to objects or tasks that were affected by the triggering event. The predicate `old` enables to access the database state at the time of the triggering.

Actions. The actions of Chimera-Exception focus on exception management within the workflow environment and can assign a task or a case to an agent, cancel a task or a case, start a task or a new case, suspend a task...

The trigger `ProductionNotification` valid for the schema `OrderManagement` is an example of a Chimera-Exception trigger. It monitors – as event – the completion of the task `Production`: after the completion of the task, the trigger notifies the `Sales Office` about the terminated production.

```
define trigger ProductionNotification for OrderManagement
  events    taskEnd("Production")
  condition Agent(A), A.Name = "Sales Office"
  actions   notify(A,"Production done.")
end
```

The `CustomerCancel` trigger reacts to cancellations of the whole process. The process is informed of the cancellation by changes to the workflow-relevant variable `CustomerCalledToCancel`. During case execution, should the parameter change to "Yes", the task `CompensateAndDecline` is started.

```
define trigger CustomerCancel for OrderManagement
  events    modify(CustomerCalledToCancel)
  condition OrderManagement(O),
            occurred(modify(CustomerCalledToCancel),O),
            O.CustomerCalledToCancel="Yes"
  actions   startTask("CompensateAndDecline")
end
```

5 Modelling ECA Rules in XPDL

In the next sections, we show how the textual specification of high-level Chimera-Exception rules can be translated into XPDL by a mapping mechanism based on

XPDL *macros*. Macro modules represent fragments of the process graph and provide flexible interconnections of workflow fragments and macro modules.

Macros consider *events*, *conditions*, and *actions*. High-level macros are made of lower-level macros, such as connector macros, and of a set of predefined sub-processes, which serve for evaluating expressions.

5.1 The Rationale for Macros

A generic macro element has at least one input node, prefixed by "`In-`", and one output node prefixed by "`Out-`". The suffix specifies the kind of graph element that can be connected to the interface. Table 1 shows the possible suffixes for interface nodes. Triggers may have a void condition primitive (if set to `none`), while the event and the action part can not be null; for simplicity, conditions will be considered as part of the action. Therefore, the `Ac` suffix can connect action macros as well as condition macros.

Table 1. Possible interface suffixes for specifying the kind of graph element to connect to a given macro element

Interface	Suffix	Graph element
In	Wf	Primary workflow
	Ev	Event macro
	Ac	Action or condition macro
Out	Wf	Primary workflow
	Ac	Action or condition macro
	True/False	Action macro

Connecting two macros requires connecting the output interface of the first macro to the input interface of the second macro. In order to be connectable, the two interfaces must present the same suffix. Interfaces are modelled by means of *task* nodes, because task nodes enable more flexible connection configurations than *transitions*. Transitions, in fact, would allow only one connection through an interface, while the auxiliary tasks allow more than one incoming transition to be connected to a specific interface.

5.2 ECA Macro Elements

We now consider the generic, minimal structure of the three high-level macros representing events, conditions and actions.

Events. Figure 2(a) depicts a generic macro for events with one input interface and one output interface. Actually, events should respect the macro structure of Figure 2(b), as generally events are not activated by incoming transitions. However, since we are tackling the problem of mapping events to XPDL process definitions/graphs, events require an input interface for connecting them to the primary process flow, too. Indeed, as shown in Section 7.1, event macros could also contain the event-generating logic, rather than representing the event itself.

(a) Macro with `In-Ev` and `Out-Ac` interfaces.

(b) Macro with `Out-Ac` interface.

Fig. 2. Generic macro for Chimera-Exception events

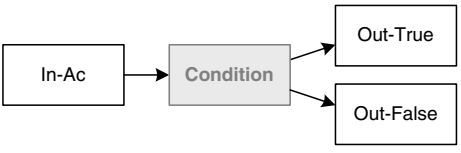


Fig. 3. Generic condition macro

Conditions. Figure 3 shows a generic condition macro with one input and two output interfaces. This macro is the only one using the suffixes `True` and `False` to denote the interface activated if the condition evaluates *true* or *false*.

The use of condition macros requires the introduction of an exception to the general rule for connecting interfaces (see Section 5.1), as there is no input interface with `True` or `False` suffixes. Therefore, if condition macros are used, their output interfaces can only be connected to interfaces with `Ac` suffixes.

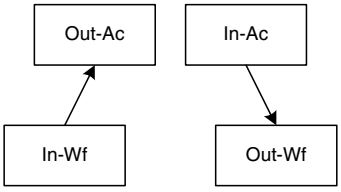
Fig. 4. Generic description of Chimera-Exception actions

Actions. Figure 4 describes the generic macro for actions. It has one input interface and one output interface, both to be connected to other `Ac` interfaces.

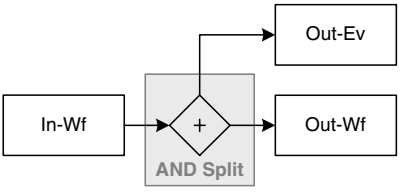
5.3 Basic Connector Macros

In order to switch between the "process environment" and the "exception environment" within an enriched process definition, proper connector macros are introduced. Figure 5 depicts the three basic connectors adopted in this paper.

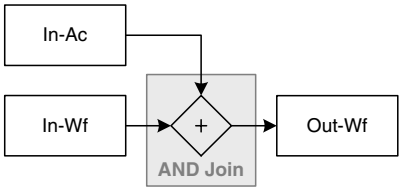
Serial macro. This macro (Figure 5(a)) allows splitting the normal process flow and inserting an exception handling sub-flow. The normal process is connected to the two interfaces with suffix `Wf`, while the two other interfaces connect to conditions or actions. The serial macro is used only to map exceptions triggered by workflow events, which are *synchronous* and *localized* (see Section 4). The serial connector may therefore act as both connector and triggering event.



(a) Serial XPDL macro element



(b) AND Split macro element



(c) AND Join macro element

Fig. 5. Basic connector macros

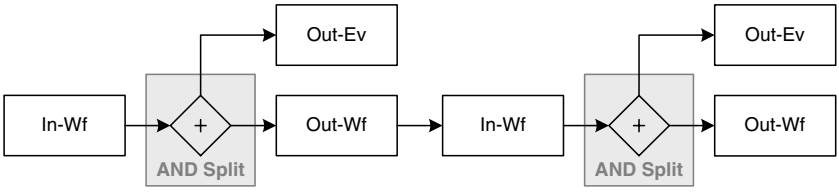
AND split. The AND split (Figure 5(b)) starts parallel flows, one for the normal process execution, one for the exception handling. The output interface with suffix `Ev` is arranged for attaching proper event macros; hence, the AND split does not represent an event. The AND split is particularly suited for mapping *asynchronous* or *synchronous, sparse* exceptions (see Section 4), when used in combination with event macros that monitor the actual triggering event (i.e., data events, temporal events, external events).

AND join The AND join (Figure 5(c)) aims at joining a process flow and a parallel exception handling flow. The join represents the termination of a triggered rule and allows connecting in input an action or a condition macro.

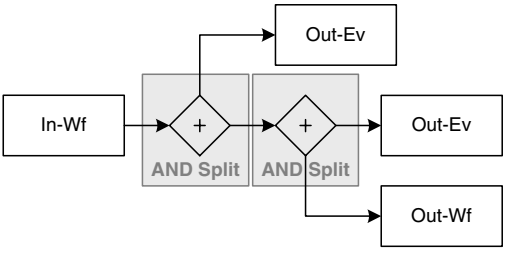
Process Graph Reduction. The use of task nodes for interfaces leads to a high number of auxiliary tasks within the process graph. Input and output interfaces as well as suffixes are particularly useful for the consistent enrichment of the process graph with exception handling constructs. After the mapping process is terminated, the additionally introduced interface nodes are no longer needed and the enriched process graph can be optimized. Figure 6 exemplifies the graph reduction or optimization process relative to two connected AND splits: whenever the process graph contains an output interface connected to an input interface having the same suffix, the two nodes can be removed.

6 Supporting Chimera-Exception Expressions

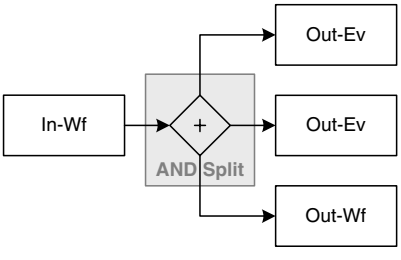
Chimera-Exception rules may contain complex expressions: we need some XPDL constructs capable of interpreting object-oriented primitives. We propose a



(a) Two connected AND splits



(b) Elimination of redundant In/Out tasks



(c) Reduced graph after optimization

Fig. 6. Process graph reduction

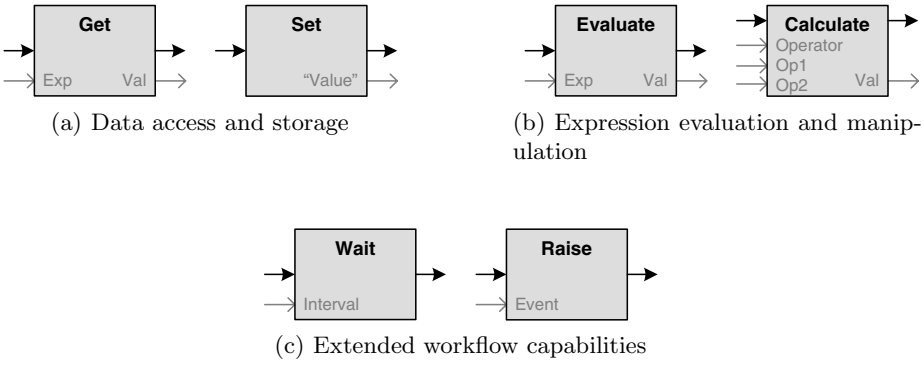
Fig. 7. Notation for modelling sub-processes and workflow-relevant data

solution based on sub-processes to provide basic and atomic functionalities for expression evaluation and data access (*basic sub-processes*). We also define *expression patterns* referring to particular configurations of workflow primitives and sub-processes, implementing higher-level functionalities with respect to those supported by sub-processes. Figure 7 depicts the notation for *sub-processes*, *data flows* and *workflow-relevant data fields*. Sub-processes exhibit the names of their input data, positioned at the left hand side, and of output data, at the right

hand side. A workflow-relevant data field connected by an arrow to an input variable represents *consumed* values, while a field connected to an output variable gathers *produced* values. An appropriate use of workflow-relevant data fields therefore allows combining sub-processes and propagating parameter values.

6.1 Basic Sub-processes for Expression Evaluation

Figure 8 depicts some basic sub-processes. *Get(ExpRef)* accesses persistent data by object-oriented expressions (i.e., `Task1.Status`): *ExpRef* is a string variable referencing the variable to be read. *Set(Value,DataRef)* assigns values to workflow-relevant data fields: `Value` is the constant value to be assigned to variable referenced by *DataRef*. *Evaluate(PredRef)* evaluates Chimera-Exception condition predicates and returns the evaluation result that can be stored within a workflow-relevant data field: *PredRef* references the predicate to be evaluated. *Calculate(OperatorRef,Op1Ref,Op2Ref)* computes the 4 basic arithmetic operations, referenced by *OperatorRef*, over the operands referenced by *Op1Ref*, *Op2Ref*. *Wait(IntRef)* implements a temporal delay, where the termination of the sub-process signals the expiration of the delay referenced by *IntRef*. *Raise(Event)* implements the WfMS-specific logic required to detect external events, described by a unique name; the termination of the sub-process implies the occurrence of the specified external event.



(a) Data access and storage

(b) Expression evaluation and manipulation

(c) Extended workflow capabilities

Fig. 8. Basic subprocesses for evaluating Chimera-Exception expressions

Also proper sub-processes for Chimera-Exception *actions* are required. Actions strongly depend on the adopted WfMS: each action requires a customized sub-process, implementing the respective functionality on top of the chosen WfMS.

6.2 Expression Patterns

Complex *expression patterns* can now be defined. Expression patterns and basic sub-processes are the starting point for the automatic translation of Chimera-Exception triggers into XPDL.

Figure 9 shows an example of a workflow pattern `GetDataObject` mapping to XPDL Chimera-Exception expressions like `Task1.Status`. As shown, the pattern is composed of one `Set` operation and one `Get` operation. The former assigns the constant value `''Task1.Status''` to a workflow-relevant data field (`Exp`); the latter evaluates the expression and accesses the respective data. The retrieved value is then stored within the data field `Val`.

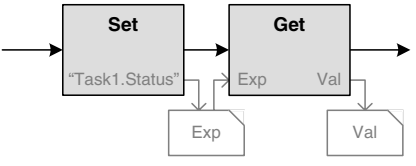


Fig. 9. Expression pattern for accessing persistent workflow data and status information

Besides data access patterns, other patterns manipulate strings, correctly instantiate `Wait` sub-processes, evaluate Chimera-Exception predicates...

By combining and concatenating basic sub-processes, according to such expression patterns, we build the specification and translation of complex expressions, as exemplified by Figure 10, where the expression `''Status of Process = '' + C.Task1.Status` is expressed by proper sub-processes. For this purpose, two expression patterns are used, one for accessing persistent data within the underlying data source, and one for the concatenation of generic strings.

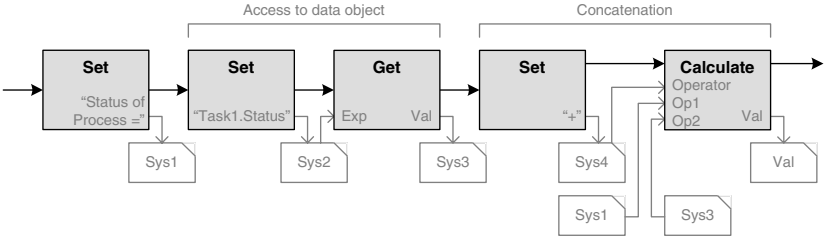


Fig. 10. Mapping of the expression `''Status of Process = '' + C.Task1.Status`

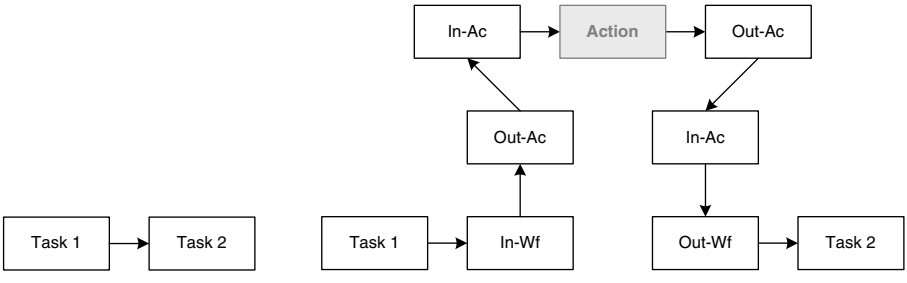
The example in Figure 10 further shows how sub-processes are configured and connected automatically. The constant values used by the `Set` operations within the depicted workflow fragment can be directly taken from the expression to be translated, while the passing of parameters from one sub-process to another can be achieved by means of system-generated data fields (i.e., `Sys1` or `Sys2`). The result of the described chain of sub-processes is stored within the data field `Val`. System-generated data fields can be derived at compilation time in a completely automated way, starting from the expression to be translated and the set of known expression patterns.

7 Mapping Exceptions to XPDL

The mapping of expressions can be achieved in a completely *process-independent* way, while the mapping of the high-level ECA constructs can only be accomplished in a *process-dependent* way. Sub-processes can directly be translated into proper XPDL constructs for building up *event*, *condition* and *action* macros, which then are connected to the process graph by means of the already mentioned interface nodes. In this section, we describe in more detail how ECA macros can be built, starting from basic sub-processes, and how they can be connected to the process graph.

7.1 Mapping Events

Events are the starting point for the evaluation of ECA rules and directly depend on the kind of exception. We have workflow, data, temporal, and external events.



(a) Start configuration: two subsequent tasks

(b) After insertion of the `Serial` macro

Fig. 11. Mapping of the `taskEnd(Task1)` workflow event

Workflow Events. Workflow events are *synchronous*, *localized*, and are mapped by the *serial* connector macro described in Section 5.3. Figure 11(b) shows the fragment of Figure 11(a) after the mapping of the event `taskEnd(Task1)`. The serial connector macro plays the twofold role of connector and event. The two interface nodes available after the mapping process directly allow the connection of action or condition macros. Events `taskEnd()`, `caseStart()` and `caseEnd()` map analogously.

Data Events. Data events are *asynchronous*, not localized, and mapped by an *AND split* connector, which splits the process flow immediately after the *start node* into two parallel flows, one for the primary process, one for the handling of the asynchronous exception.

Figure 12 exemplifies the mapping of the data event `modify(Object)` and its connection to the process graph by means of the AND split connector (inside the grey-shaded box at the bottom of the figure). The actual event macro is connected to the `Out-Ev` interface of the connector by means of an `In-Ev` interface.

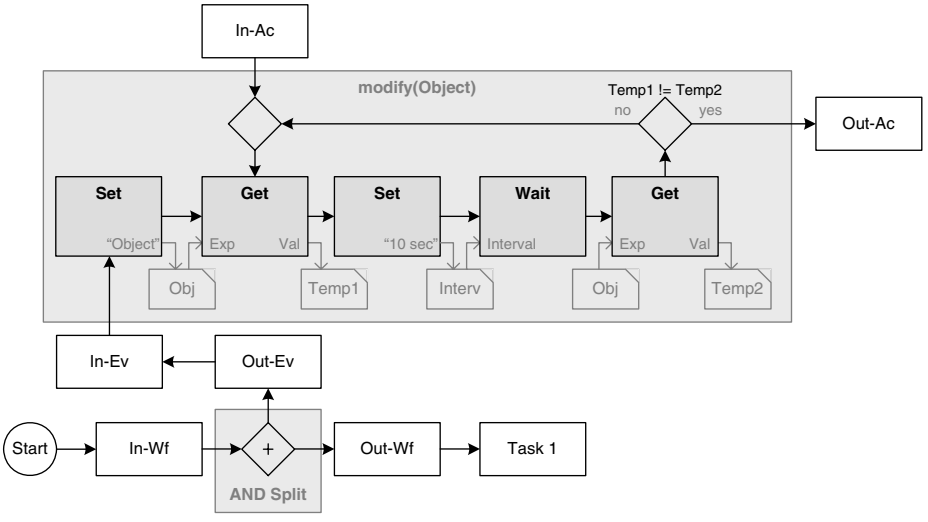


Fig. 12. Mapping of the `modify(Object)` data event

The `modify(Object)` event further presents two other interfaces towards condition or action macros (`Out-Ac` and `In-Ac`). The former enables the execution of the rule's actions in case of modifications of the data object `Object`, the latter re-connects the output interface of the action macro to the event macro.

The event macro in itself, rather than representing a real event, contains the runtime logic required for "generating" actual data events. This is achieved by cyclic monitoring (polling) of the respective data attribute within the underlying data source and transferring control to the condition and action macros, in case a modification is detected. In such a case, re-connecting the output interface of the action macro to the `In-Ac` interface of the event macro closes the polling cycle, which is suspended for the whole time taken to execute the rule's actions.

Internally, the `modify(Object)` event consists in the following steps: the current value of the data object to be monitored is retrieved and stored in a workflow-relevant data field (`Temp1`). After the expiration of a predefined time interval, the (possibly changed) value of the monitored data object is stored in a second data field (`Temp2`). The two stored values are compared and, if they differ, the `Out-Ac` interface is enabled and the rule's condition and action parts are evaluated. Otherwise, the macro continues polling the monitored data object. The `delete` and `create` events map analogously.

Temporal Events. Temporal events require different mapping techniques, mainly based on their synchronicity features.

- *Instant events* are asynchronous with respect to the normal process flow. Their implementation is based on the use of suitable `Wait` operations;
- *Periodic events* are asynchronous, thus mapping similarly to instant events;

– *Interval events* are synchronous events as their evaluation depends on workflow events. The event `elapsed(interval 1 day) since taskStart(Production)`, for instance, is bound to the *anchor* event start of task `Production`. These events are mapped by branching the normal process flow according to the anchor event and by connecting the event handling constructs to the parallel control flow. After the expiration of the time interval, the actual event occurs, and the respective ECA rule is evaluated.

External Events. External events are asynchronous. The implementation of the appropriate detection mechanism strongly depends on the particular used WfMS. The detection of external events requires a proper sub-process (Section 6.1), which is in charge of waiting for the incoming event. After receiving the notification of the occurrence of the external event, the sub-process terminates, thus signalling the occurrence of the event. The respective macro for raising asynchronous, external events is based on cyclic activations of this sub-process, and is attached in parallel to the process graph after the start task. A more detailed treatment of external events exceeds the scope of this paper.

7.2 Mapping Conditions

The mapping of a condition involves as many instances of the `Evaluate` subprocess (introduced in Section 6.1) as predicates in the condition expression. The single intermediate evaluations are stored as workflow-relevant data, while a final condition primitive determines the overall evaluation.

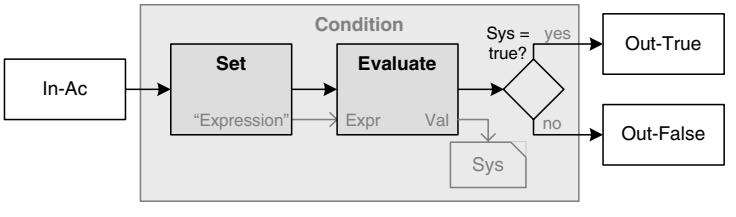


Fig. 13. XPDL macro element for expressing conditions

Figure 13 exemplifies the described mapping of condition expressions: the `Set` operation stores the predicate to be evaluated (a string constant) in the data field `Expr`, which is then consumed as input by the `Evaluate` operation. This operation parses the predicate and stores the result in a system-generated data field (`Sys`). The actual result of the condition macro is computed by the final condition primitive.

7.3 Mapping Actions

Action implementation must be provided by basic sub-processes prior to the mapping process. In order to form proper action macros and to connect such

sub-processes to the process graph, action sub-processes are wrapped by means of an `In-Ac` interface and an `Out-Ac` interface. Possible parameters are then specified by means of complex Chimera-Exception expressions, interpreted, and made available to the execution environment by workflow-relevant data fields.

In this way, the actions described in Section 4.1 can be seamlessly integrated into the enriched process definition. While the translation of exception triggers from Chimera-Exception to XPDL only requires the definition of the *interface* of the actions to be mapped, their execution on top of a particular WfMS requires their *implementation* according to the WfMS's extension mechanisms.

8 Compiling Chimera-Exception into XPDL

As a proof of concept, we developed a Java prototype of a *Compiler* for the automatic mapping into XPDL of exceptions defined in the Chimera-Exception language. An additional *Optimizer* module is in charge of reducing the enriched process graph by eliminating useless interface nodes.

The prototype demonstrates both the feasibility of the envisioned automatic translation of Chimera-Exception triggers into XPDL at the process definition level, and that the enriched process definitions are effectively executable by real workflow engines, if proper WfMS-specific sub-processes are provided.

8.1 XPDL Process Modelling and Rule Compilation

To validate the compilation process, the open-source, graphical workflow process editor JaWE [18] has been adopted. JaWE enables the visual specification of workflow processes and their storage in XPDL as native file format. The process definition example of Figure 1 is now modelled in JaWE by the swim lanes *Sales Office* and *Production* of Figure 14: white nodes represent *routing nodes* (automatically performed), while the gray shaded nodes represent the actual tasks to be executed by the resource/role associated to the respective swim lane.

Several trigger definitions have been compiled to test the compilation process: as an example, we consider here the `CustomerCancel` trigger described in Section 4.1 (Figure 14). The triggering event is an asynchronous data event, occurring after changes to the workflow-relevant data field `CustomerCalledToCancel`. If a modification to this parameter occurs, the case is aborted by activating the `CompensateAndDecline` task.

The enabling event `modify(CustomerCalledToCancel)` has been connected in parallel to the primary process flow by branching the primary flow immediately after the start node. According to Figure 12, the data event is implemented by a polling mechanism, monitoring the value of the workflow-relevant data field `CustomerCalledToCancel` and represented in figure by the tasks `Get`, `Wait` and the first `Cond` task. The second `Cond` task maps the conditional predicate `OrderManagement(O)`, `occurred(modify(CustomerCalledToCancel),O)`, and `O.CustomerCalledToCancel = ''Yes''`, while the specified action `startTask(''CompensateAndDecline'')` is achieved by connecting one of the outgoing transitions of the `Route` node to the respective task.

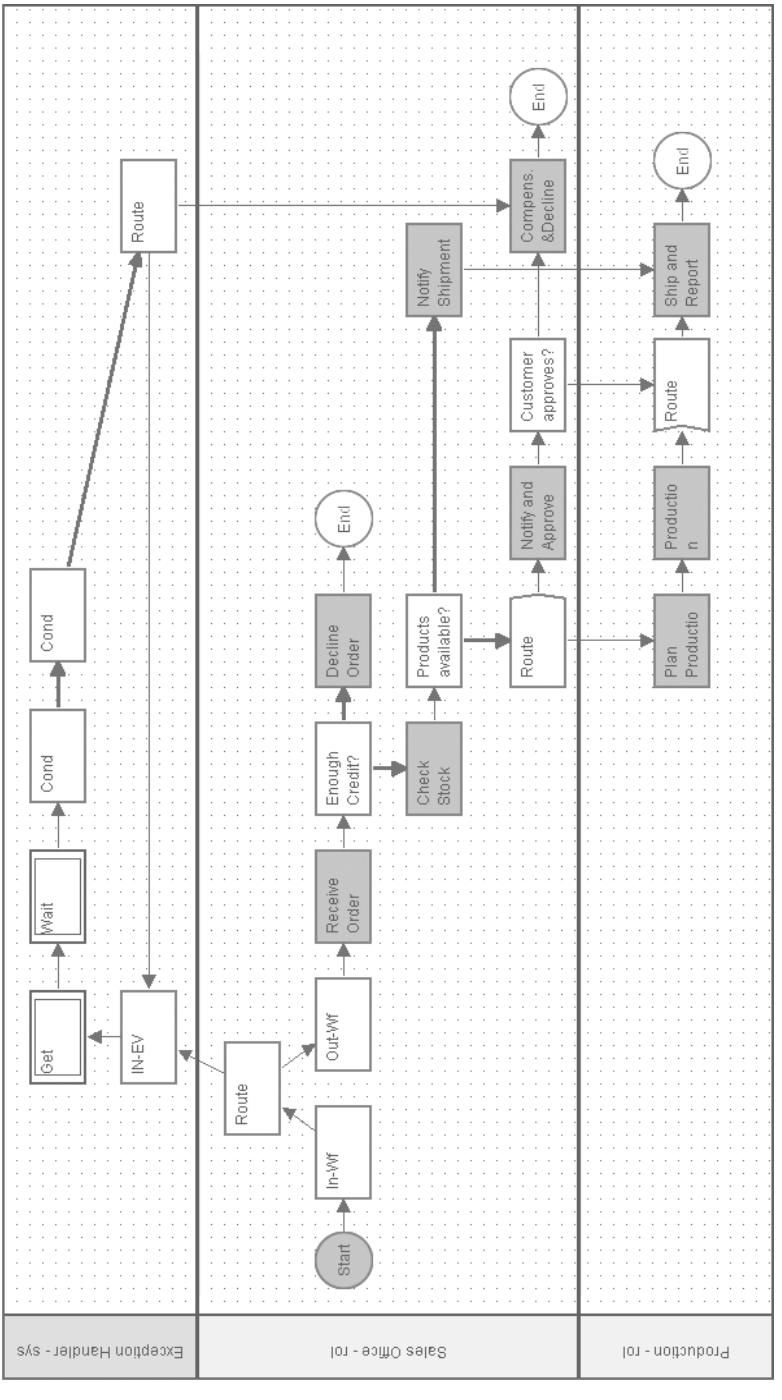


Fig. 14. Process model after compilation of the `CustomerCancel` trigger

The evaluation of the execution of enriched process definitions has been tested by means of the open-source Java workflow engine Shark [19], which is completely based on XPDL as native process definition format. In order to execute enriched process definitions, the basic sub-processes introduced throughout this paper have been implemented as "Java applications" to be associated to the so-called `application tasks` (performed by means of predefined applications). The performed tests allow us to cover successfully both compilation and execution of enriched process definitions.

9 Conclusions and Future Work

In this paper we considered a relevant aspect of modern workflow management systems, i.e., exception handling. We leveraged the use of a high-level specification language for the definition of workflow exceptions, such as Chimera-Exception, and proposed a fully automated translation technique for rule definitions. According to our approach, rules are translated into XPDL, the standard process definition language proposed by the Workflow Management Coalition, which hence fosters portable process and exception handling specifications. As a proof-of-concept, the implemented rule compiler prototype allowed us to test all the stages of the compilation process. The main contribution of this paper is that it keeps the definition of exceptions at a conceptual level, by assuring – at the same time – portability.

As future research directions, we plan a formal analysis of the *complexity* of the outlined mapping approach, which however we expect being *linear* since there are no interdependencies between the mappings of two triggers. We are also confident about the *termination* of enriched processes, assuming a correct termination of the input processes and of the Chimera-Exception triggers to be mapped.

References

1. The Workflow Management Coalition. http://www.wfmc.org (2005)
2. Combi, C., Pozzi, G.: Architectures for a Temporal Workflow Management System. In: SAC '04: Proceedings of the 2004 ACM symposium on Applied computing, New York, NY, USA, ACM Press (2004) 659–666
3. Eder, J., Liebhart, W.: The Workflow Activity Model WAMO. In: CoopIS. (1995) 87–98
4. Casati, F., Ceri, S., Paraboschi, S., Pozzi, G.: Specification and Implementation of Exceptions in Workflow Management Systems. ACM Transactions on Database Systems **24** (1999) 405–451
5. Mourão, H., Antunes, P.: Exception handling through a workflow. In Meersman, R., Tari, Z., eds.: CoopIS/DOA/ODBASE (1). Volume 3290 of Lecture Notes in Computer Science., Springer (2004) 37–54
6. Golani, M., Gal, A.: Flexible business process management using forward stepping and alternative paths. In van der Aalst, W.M.P., Benatallah, B., Casati, F., Curbera, F., eds.: Business Process Management. Volume 3649. (2005) 48–63

7. Reichert, M., Dadam, P.: Adept$_{flex}$-supporting dynamic changes of workflows without losing control. J. Intell. Inf. Syst. **10** (1998) 93–129
8. Luo, Z., Sheth, A.P., Kochut, K., Arpinar, I.B.: Exception handling for conflict resolution in cross-organizational workflows. Distributed and Parallel Databases **13** (2003) 271–306
9. Schuschel, H., Weske, M.: Triggering replanning in an integrated workflow planning and enactment system. In Gottlob, G., Benczúr, A.A., Demetrovics, J., eds.: ADBIS. Volume 3255 of Lecture Notes in Computer Science., Springer (2004) 322–335
10. van der Aalst, W.M.P., Weske, M., Grünbauer, D.: Case handling: a new paradigm for business process support. Data Knowl. Eng. **53** (2005) 129–162
11. Hagen, C., Alonso, G.: Flexible Exception Handling in the OPERA Process Support System. In: ICDCS. (1998) 526–533
12. Baan Company N.V. - COSA Soultions: COSA Reference Manual (1998)
13. McCarthy, D., Sarin, S.: Workflow and transactions in In-Concert. IEEE Data Engineering, 16(2):5356 (1993)
14. Staffware Corporation: Staffware for Intranet based Workflow Automation. http://www.staffware.com/home/whitepapers/data/globalwp.htm (1997)
15. (BPMI.org), B.P.M.I.: Business Process Modeling Notation - Version 1.0. www.bpmi.org (2004)
16. Casati, F., Pozzi, G.: Modeling Exceptional Behaviors in Commercial Workflow Management Systems. In: CoopIS. (1999) 127–138
17. Leban, B., McDonald, D.D., Forster, D.R.: A Representation for Collections of Temporal Intervals. In: Proceedings of the Conference on AAA-I, (AAAI86, Philadelphia, PA) (1986) 367–371
18. ObjectWeb Consortium: Enhydra JaWE (Java Workflow Editor). http://jawe.objectweb.org/ (2005)
19. ObjectWeb Consortium: Enhydra Shark. `http://shark.objectweb.org/` (2005)

Methods for Enabling Recovery Actions in Ws-BPEL

Stefano Modafferi and Eugenio Conforti

Politecnico di Milano
Dipartimento di Elettronica e Informazione
P. zza L. da Vinci 32,
20133 Milano, Italy
modafferi@elet.polimi.it, conforti_eugenio@bah.com

Abstract. Self-Healing is an emerging exigency for Information Systems where processes are everyday more complicated and where many autonomous actors are involved. Roughly, self-healing mechanisms can be viewed as a set of automatic recovery actions fired at run-time according to the detected fault. These actions can be at infrastructure level, i.e. transparently to the process, or they can be defined in the workflow model and executed by the workflow engine. In the Service Oriented Computing world Ws-BPEL is the most used language for web-service orchestration, but standard recovery mechanisms provided by Ws-BPEL are not enough to implement, with reasonable effort, lots of suitable recovery actions.

This paper presents an approach where a designer defines a Ws-BPEL process annotated with some information about recovery actions and then a preprocessing phase, starting from this "annotated" Ws-BPEL, generates a "standard" Ws-BPEL, that is a file understandable for a standard Ws-BPEL engine. This approach has the advantage of avoiding any change in the engine using the standard capabilities to define specific behaviors that will realize recovery actions, but at the end are still a set of Ws-BPEL basic and structured activities.

1 Introduction

In Service Oriented Architectures languages and framework for managing processes are becoming mature and quite stable. The de-facto standard for defining Web-Service based process is Ws-BPEL [7]. Several engines [1,2,3] are available for implementing Ws-BPEL processes.

The next step for research in this field is an advanced fault management to realize Self-Healing Information Systems able to choose and implement appropriate recovery actions. Thus a Self Healing Information System will implement a monitoring part to capture faults, a diagnosis part to detect what and where is the source of the problem and to decide which recovery action has to be fired, and a recovery part that actually implements the recovery action.

These actions can be at infrastructure level, i.e. transparently to the process (e.g. the approach of [17]), or they can be defined in the workflow model and

R. Meersman, Z. Tari et al. (Eds.): OTM 2006, LNCS 4275, pp. 219–236, 2006.

executed by the workflow engine. This paper focuses on the recovery part of a Self Healing Information System following the latter approach and considering recovery mechanisms that can be defined during the process design phase.

A designer who wants to use a language like Ws-BPEL as is, has very few and basic mechanisms to implement recovery actions. It is possible to realize more complex recovery behavior, but the effort is too much.

Our approach tries to extend Ws-BPEL strategies suggesting some composite constructs that mix handlers and activities (standard and structured), to drive the designer in his work, enabling standard, and therefore general, mechanisms which perform repair actions at run time level.

During the design phase, the designer has to decide which mechanism must be used, specializing it according to the given schema. That is, for instance, inserting in the general mechanism the actual recovery actions.

The proposed mechanisms cover a wide range of possibility: i) the ability of modifying the value of process variables by means of external messages, ii) the specification of a time deadline associated with a task, iii) the ability of redoing a single task or an entire scope, iv) the possibility of specifying alternative paths to be followed, in the prosecution of the execution, after the reception of an enabling message, v) the possibility of going back in the process to a point defined as safe for redoing the same set of tasks or for performing an alternative path.

The work is so structured: Section 2 is devoted to the analysis of existing Ws-BPEL constructs for recovery actions, then in Section 3 we define our general approach to improve power expression in Ws-BPEL without modifying existing engines, mechanisms are presented in Section 4, then in Section 5 we compare our work with related literature and finally in Section 6 we draw our conclusion.

2 Ws-BPEL Recovery Mechanisms

In the Service Oriented Computing world Ws-BPEL [7] is the most used language to specify Web-Service Orchestration. It defines a language to design workflow processes by specifying the process tasks and their interaction with the external world. Ws-BPEL has standard activities (*invoke*, *receive*, *assign*, *wait*, *reply*, *terminate* etc.) called simply tasks, and structured activities (*loop*, *pick*, *sequence* etc.) that can be combined to define the process.

An important concept is the *Scope* that defines a portion of workflow upon which variables and handlers can be defined. Each *Scope* has a primary activity that defines its normal behavior. The primary activity can be a complex structured activity, with many nested activities with an arbitrary depth. The *Scope* is shared by all the nested activities. Each *Scope* has unique entry and exit points.

Ws-BPEL provides standard mechanisms for managing exceptions. Specific handlers (fault, compensation and event) are associated with a single action or with a *Scope*, that is, a set of actions. The major problems that handlers create are about their lack of flexibility. Those handlers are enabled at different time during execution: Fault and Event Handlers are enabled only during running time (of a Task or *Scope*); Compensation Handler is enabled when the status is

"completed" and therefore the execution point is ahead of the *Scope*. The fault and compensation can be defined both at Task or *Scope* level, but Event Handler exists only at *Scope* level. In the following a brief description of each handler is provided:

- Fault Handler. Its aim is to undo the partial and unsuccessful work of a *Scope*. The first thing it does is to terminate all the activities contained within the *Scope*, then it performs its tasks defined at design phase. Sources of faults can be: i) invoke activity responding with a fault WSDL message, ii) a programmatic throw activity iii) standard fault that pertains to the engine iv) "CatchAll" for anything else. If a fault is not consumed in the current *Scope* it is recursively forwarded to the enclosed ones. If termination of instance has not been invoked, after consuming the exception the normal flow restarts at the end of the *Scope* associated with the fault.
- Compensation Handler. This is basically a wrapper for compensation activities. A Compensation Handler is available only after the related *Scope* has been completed correctly. It represents a compensation process of already executed activities. Activities see a snapshot of all the variables of the *Scope* the Compensation Handler is attached to and they cannot update live data. It can be used from within fault-handlers or Compensation Handlers. Calling a compensation on a *Scope* without Compensation Handler, the default handler is executed, that is all Compensation Handlers for the immediately enclosed scopes in reverse order are invoked.
- Event Handler. The whole process as well as each *Scope* can be associated with a set of Event Handlers that are invoked concurrently if the corresponding event occurs. The actions taken within an Event Handler can be any type of activity, such as sequence or flow. Event Handler is considered a part of the normal processing of the *Scope*, i.e., active Event Handlers are concurrent activities within the *Scope*. Events can be: i) incoming messages; ii)temporal alarms.

Using these three handlers as they are, it is possible to realize very basic "recovery" mechanism. In fact Ws-BPEL provides the three handles as standard mechanisms and leaves to designer any other specification about the tasks actually executed when an handler is fired. Therefore more powerful and flexible instruments could be built, but this effort is currently fully in charge of the designer. For instance, no native method is available for redoing an action, designer could use the Compensation Handler to do this, but it is not so easy and a discussion about how to implement a redo action in Ws-BPEL is given in Section 4.3.

Moreover in developing advanced recovery mechanisms, the designer has to consider the side effect of each specific handler. For instance when *Fault Handler* is invoked, it terminates all the activities in the *Scope* upon which it is defined, therefore if the designer wants to provide a repair action on the process without killing a part of it (or all), he needs to catch the fault with the *Event Handler* that is executed concurrently to the normal flow. It does not kill any activity, but, at the same time, it does not stop the process flow.

3 Enhancing Design Capability

To overcome Ws-BPEL limitation in supporting recovery action three different approaches can be followed: to define a totally new workflow language and workflow engine, to define an extended Ws-BPEL and the corresponding extended engine or to use the concepts of annotation and preprocessing for enhancing Ws-BPEL at design time without modifying the workflow engine.

Starting from the assumption that Ws-BPEL is currently the de facto standard for orchestrating Web Service based workflow, changing the model will be useless, hardly limit the diffusion of the new model. Thus the followed direction is to give to the designer several advanced recovery mechanisms asking him very few changes of his current instruments.

For this reason the presented approach follows the third way. The payed price is the necessity of interpreting an annotated Ws-BPEL and some limitations (related to the nature of the language) that cannot be overcome with any mechanism. The advantages is to avoid any change in the engine supporting any commercial engine complaint with Ws-BPEL 1.1 standard without imposing any choice to the company interested in these mechanisms and moreover without interfering with autonomous developments carried out by the actual owner of each engine.

Our approach exploits Ws-BPEL annotation based on the XML nature of Ws-BPEL, that is adding new tags in the XML during the design phase and removing them during the preprocessing phase. Each Ws-BPEL process can be invoked like a Web Service having a WSDL interface. The final WSDL, output of pre-processing phase, is modified according to the new (recovery) operations that the process supports. To realize a distinction between the operations related to the Business and those related to Management our approach follows the direction proposed in WSDL-S [15] to specify the semantic associated with this (recovery). This language is an extension of WSDL and supports the use of extension attributes, namely *modelReference*, that specify the association between a WSDL entity and a concept in a semantic model.

As stated before, the enhancement of design capability is not related with specific process schema. It is provided like the basic Ws-BPEL handlers and thus each mechanism requires a design phase that is devoted to decide where the mechanism is active and which are the actual recovery actions that have to be performed once the mechanism has been fired.

4 Proposed Mechanisms

In this section the five mechanisms enabling specific recovery actions are described. They are the focus of the paper and cover a wide range of possibility: i) the ability of modifying the process variables value by external messages, ii) the specification of a time deadline associated with a task, iii) the ability of redoing a single Task or an entire *Scope*, iv) the possibility of specifying alternative paths to be followed, in the prosecution of the execution, after the reception of

an enabling message, v) the possibility of going back in the process to a point defined as safe for redoing the same set of tasks or for performing an alternative path. All the mechanisms, if not differently specified, can be associated both at task level and at *Scope* level.

4.1 External Variable Setting

A common typology of errors in Business Process executions is related to data. Actual recovery actions in this field are often performed outside the process by human actions. Even if it is performed out of the process, this kind of recovery usually produces the need for an update of several process variables. This mechanism allows the designer to simply identify which variables can be set from incoming message, leaving to the preprocessor the task of producing the corresponding set of Event Handlers.

In fact the basic idea behind this mechanism is to use several Event Handlers with a simple activity each one modifying the corresponding variable.

Abstract Extended Model. Even if the "annotated" Ws-BPEL is not executable on standard workflow engines, it can be still viewed as a workflow description language. Step by step we will show what has to be added to the original model for enabling advanced management of recovery actions. In the following the term "attribute" has general meaning and does not imply that in the corresponding XML structure it is actual an attribute, in fact in the XML it could also be an element.

For this feature the model has to consider also an External Modifiable attribute $ExtVar$ optionally associated with each Ws-BPEL variable definition.

The Transformation Algorithm. This is the algorithm performed by the preprocessor to generate the "standard" Ws-BPEL:

1. Scan the "annotated" Ws-BPEL generating an Event Handler for each variable having the attribute $ExtVar$ true.
2. Put an "assign" operation for modifying corresponding variable in each Event Handler generated in the previous step.

4.2 Timeout

In the communication between two web services A problem is the time that one actor can wait before the message arrival. We need a mechanism to manage, at process level, timeout in the communication. In Ws-BPEL the possibility of using timeout is associated to *Event Handler* and to *Pick* activity. In this section a simple way to extend this possibility associating a timeout to a Receive activity is presented.

The designer specifies a timeout for each selected *Receive* activity and the corresponding recovery actions, that is the set of activities performed if the timeout happens.

With simple extension involving an Event Handler it is possible to associate a timeout with activities different from *Receive*. This extension exploit the possibility of using a timeout for firing an Event Handler. For terminating the current Activity/Scope is used a Fault Handler.

Abstract Extended Model. For Timeout feature the model has to consider:

- Timeout attribute $Time$ optionally associated with "Receive" activity definition.
- Set of basic and structured Ws-BPEL activities $Rec_{Actions}$. With respect to the default execution flow $Rec_{Actions}$ identifies a semantically different behavior because it is performed as a recovery action, but syntactically it is composed of standard Ws-BPEL code.

The Transformation Algorithm. As shown in Fig. 1 and Fig. 2, to enable this feature each *Receive* activity associated with a timeout is transformed in a *Pick* activity with two branches, one associated with the original message and the other associated with an alarm, that is the standard mechanism provided by Ws-BPEL for managing timeout.

$< receive >$ ..... $< Timeout = "duration - expression" >?$ $\underline{RecActivity}$ $< /Timeout >$ $< /receive >$	$< pick >$ $< onMessage >$ $< Empty/ >$ $< /OnMessage$ $< onAlarm(for = Timeout) >$ $\underline{RecActivity}$ $< /OnAlarm >$ $< /pick >$

Fig. 1. Transformation of a receive with timeout in a pick

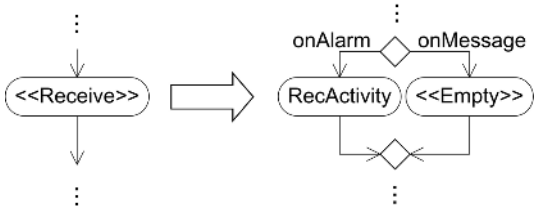


Fig. 2. UML representation of transformation

This is the algorithm performed by the preprocessor to generate the "standard" Ws-BPEL:

1. Scan all the "annotated" Ws-BPEL transforming every receive action associated with a timeout in a pick with two branches:

- A branch characterized by *onMessage* property and associated with the same message originally devoted to corresponding receive.
- A branch characterized by *onAlarm* property and associated with the corresponding defined timeout.

2. Associate the "standard behavior" with the *onMessage* branch.
3. Associate the "recovery behavior" with the *onAlarm* branch.

4.3 Redo Mechanism

Redoing an activity is an action often useful repairing a process. In Ws-BPEL there is not A mechanism to manage this type of situations. Redo means to execute again a Task or a *Scope* (a set of activities) that is already completed.

The action of Redo is not related to the concept of rolling back a process or part of it. According to [14] we are assuming that concurrently with a running process, at a time, the system can ask for redoing a Task or a *Scope* without any relationship with the current point in the execution flow.

Compensation Handler is enabled only when the related Task/*Scope* is completed. Event Handler and Fault Handler are enabled only during the running phase of the Activity/*Scope*.

To redo an activity the mechanism is based on the Compensation Handler.

```
< CompensationHandler >?
        < switch >
            < casecondition = "bool − expr" > +
                Redoactivity
            < /case >
            < otherwise >?
                Compensateactivity
            < /otherwise >
        < /switch >
< /CompensationHandler >
```

Fig. 3. New structure of Compensation Handler

In this mechanism redo is viewed as a compensation activity. The basic idea is to define a behavior that syntactically is inserted in a Compensation Handler, but actually is the repetition of "normal" behavior. An Event Handler fired by the specific redo message is used to activate this compensation.

It is possible to use the same construct also at single task level, because the *Invoke* activity has an internal Compensation Handler.

It is important to remark that for each Task/*Scope*, only one Compensation Handler can be defined and it cannot be invoked by an external message. For this reason the message rising a redo action will syntactically be an event message. The aim of the Event Handler is to set the variable that will drive the choice between redo and compensation and then to call the Compensation Handler.

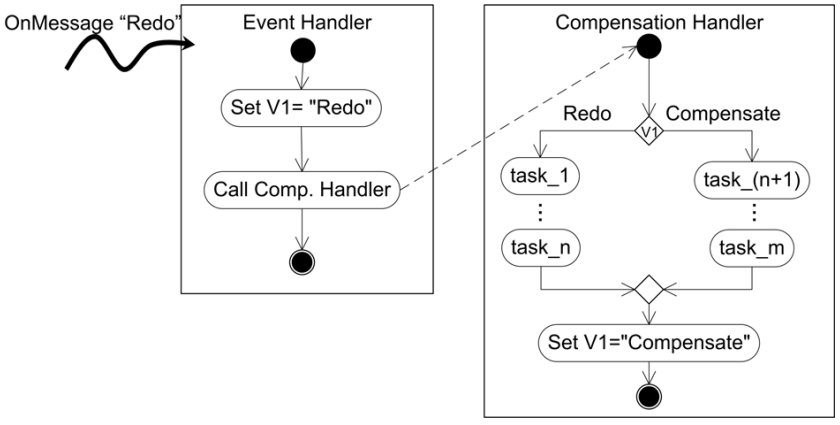


Fig. 4. Overview of Redo mechanism at run-time

Using this approach it is possible to have two different messages for raising compensation or redo action. The default behavior performed in the Compensation Handler is the compensation path.

The designer has to only specify the Tasks/Scopes that could be redone.

Abstract Extended Model. For this feature the model has only to consider the Redo attribute *Redo* optionally associated with *Invoke* activity definition or with *Scope*. From the enhanced model perspective no other information is necessary. Fig. 3 shows how the information can be provided.

The Transformation Algorithm. The following transformation algorithm is performed for each Task/Scope upon which the *redo* attribute is defined:

1. A global variable v_i is defined by the preprocessor. This variable will drive the choice between compensation and redo action.
2. A new event message and the corresponding handler is created.
3. In the Event Handler an operation for setting v_i to "redo action" value is defined.
4. In the Event Handler an operation for calling the Compensation Handler associated with the activity is inserted.
5. a *Switch* construct is put on the top of Compensation Handler by the preprocessor.
6. The Task or the *Scope* is copied in a branch of the switch in the handler and the normal compensation behavior in the other branch by the preprocessor.
7. This switch is driven by a variable v_i that indicate if the wanted behavior is compensate or redo.
8. An operation for setting the value of v_i to "Compensation", default value, is inserted.

In Fig. 4 a basic functional schema is depicted. Notice that the compensation branch could be a shortcut if compensation is not defined upon the corresponding Task/Scope.

4.4 Future Alternative Behavior

This mechanism provides a way for specifying that as consequence of an event, in the future the process, in specific sections, will follow an alternative behavior instead of the default one.

The typical example is when a given (and not vital) service, during the process execution becomes not available and an incoming message carries this information. Each operation related to this service will be skipped until the situation does not change.

The designer has to define the following items:

1. The scopes of workflow sensible to one (or more) possible alternative behaviors. To do this he defines a *Scope* exactly covering every region that could be substituted.
2. The set of "Alternative Behaviors" associated with the workflow, that is the set of scopes available as alternative paths.
3. The relationship among default scopes and alternative scopes. Each region can be associated with more than one different Alternative Behaviors.

High-level Overview. The idea behind this mechanism is to have some alternative behaviors available along the process and related to several portions of it. The preprocessing phase uses a *Switch* activity to store all the possible alternative behaviors in the corresponding places, specific variable drives one and only one of this kind of *Switch*. Each alternative behavior is fired (or killed) by a specific message. To manage this situation the Ws-BPEL Event Handler is used. The incoming message carries a boolean information (Alternative/Default). When a message arrives the Event Handler reacts and all the variables driving the *Switch* activities containing the alternative behavior indicated in the message are set to the indicated value.

This approach assumes that default and alternative behaviors are mutually exclusive and that the incoming message fires the associated behavior along all the process. If for a single *Scope* more than one alternative behavior has been defined, the last incoming message will decide the actual behavior. This is quite reasonable because the semantic associated with the message can be summarized in: "From now perform the associate alternative behavior wherever it is defined".

Figure 5 shows an high level picture of this mechanism. It is worth noticing that in the message content it is possible to specify if the alternative pattern has to be activated or if the default behavior has to be restored.

Even if not yet implemented, an important evolution of this mechanism is the possibility of specifying the alternative behavior not as an absolute set of tasks with a predefined topology defined at design-time, but as a set of rules defined upon the default behavior (e.g. after each operation related to partner a perform an *Invoke* operation to partner b informing about the content of the previous communication).

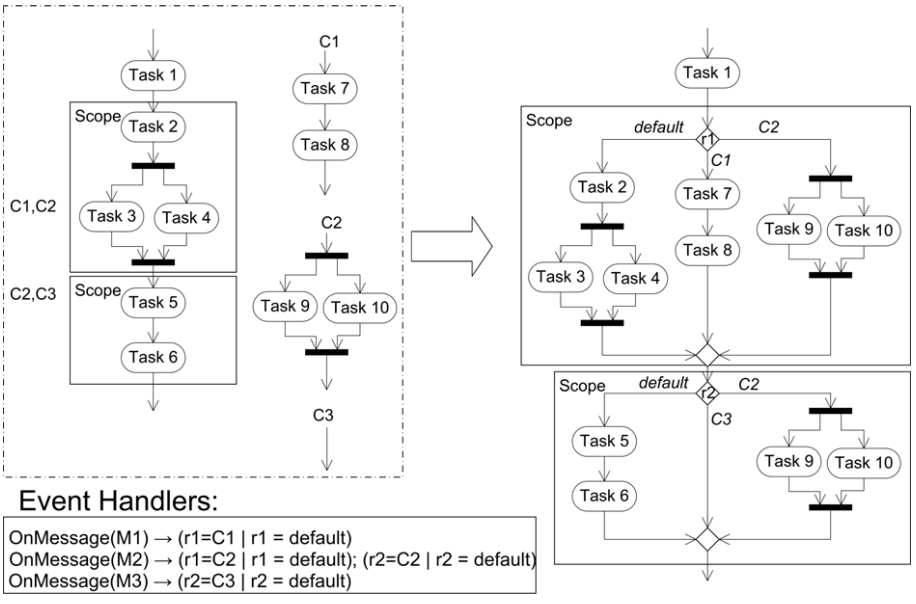


Fig. 5. Overview of future alternative behavior mechanism

Abstract Extended Model. The extended model for this feature has also to consider:

- A *Conditional Scope* CS_i which is a *Scope* with his alternative behaviours:
  - Starting Point CS_iP_s of the *Scope*
  - Ending Point CS_iP_e of a *Scope*.
  - Set of Alternative Arcs AA_{CS_i} linking the *Scope* with all the corresponding available *Alternative Scopes*. Each arc is defined as $AA_{CS_i-AltScope_j}$.
- The set of *Alternative Scope AltScope* available for the workflow. Each $AltScope_i$ is defined as a set of basic and structured Ws-BPEL activities. $AltScope_{CS_i}$ identifies the set of alternative behaviors associated with a *Conditional Scope* CS_i. With respect to the default execution flow $AltScope_i$ identifies a semantically different behavior because it is performed as a recovery action, but syntactically it is composed of standard Ws-BPEL code.

The Transformation Algorithm. The Pre-processing algorithm for each *Conditional Scope* is defined as follow:

1. For each *Conditional Scope* CS_i:
   - In correspondence of each *Conditional scope* a *Switch* activity starting from CS_iP_s and ending in CS_iP_e is placed. A variable r_{CS_i} drives the switch construct.
   - The associated $AltScope_{CS_i}$ is identified. It represents the set of alternative behaviors available for CS_i.

- The *Switch* structure have $| AltScope_{CS_i} | +1$ branches. A branch is filled with the default behavior and the others with an available *Alternative Scope*.

2. For each *Alternative Scope* $AltScope_i$:
   - An Event Handler $EH_{AltScope_i}$ associated with message $MSG_{AltScope_i}$ enabling the *Alternative Scope* $AltScope_i$ is defined.
   - All the *Scope* (and the set of corresponding r_{CS_i}) where the $AltScope_i$ behavior is defined are identified.
   - The body of $EH_{AltScope_i}$ is filled with a set of *Assign* activities devoted to switch each involved r_{CS_i} from current value to $AltScope_i$ or to the original CS_i according to message content.

4.5 Rollback and Conditional Re-execution of the Flow

A common recovery action in workflow exception management is to compensate a part of the executed process, rolling back the state to a "safe point". it is possible to compensate a *Scope* using the simple Compensation Handler provided by standard Ws-BPEL, but the execution flow can only proceed ahead and no "jump" or "go to" construct are provided. The only way to go back is to use a loop in a proper way. In the following we will refer to a more general mechanism that allows to rollback the process until a safe point and then to execute the same or a possibly different behavior.

The concept of safe points is derived from [11] and their identification is in charge of the designer. In the following they will be addressed as "migration points" because each point can be the "starting point" for migrating to an alternative behavior.

The designer has to specify:

- The point considered safe, that is the point suitable for ending a rollback process and starting migration process. Migration process can be also "self-migration", that is starting and ending configurations/behaviors are the same.
- The compensation process associated with the rollback action
- The arcs linking a "starting migration point" with related "ending migration point" of different behaviors and the optional transformation process associated with each arc.

The use of both "Starting" and "Ending" Migration Point allows general migrations without enforcing a symmetry among different behaviors.

High Level Overview. The solution is derived from the one presented in [16] where the authors propose a way for specifying run-time change of configuration as reaction to a context change. It uses the Fault Handler property that terminates all the activities inside a *Scope* and restarts the flow after the end of that *Scope*. This *Scope* is inserted in a loop and the Fault Handler modifies the variable driving the loop for performing another iteration. Two different switch

constructs drive the choice of the configuration that will be performed after the rollback phase and the restarting from the correct point in the configuration.

In this way, by combining Fault Handler properties, specific code performed in the Fault Handler and an appropriate main flow structure it is possible to have a behavior that realize the rollback using the Ws-BPEL language.

Abstract Extended Model. The extended model will consider:

- A Context-sensitive region (CSR) that is a workflow subprocess that supports rollback and may have several configurations exporting different behavior according to specific conditions. A CSR will be defined exactly upon a Ws-BPEL *Scope*.

  A region is composed of alternative configurations linked with particular arcs called *migration arcs*. A migration arc is associated with instructions to migrate a workflow instance from one configuration to another.
- A Configuration that is always a workflow subprocess, but with different characteristic with respect to CSR. A Configuration $Conf_i$ is composed by: i) an entry condition EC; ii) basic and structured activities; iii) a set of Starting Migration Points MPs; iv) a set of Ending Migration Points MPe; v) a set of directed Configuration Arcs FC.

  The entry condition EC is an expression used to define when the configuration has to be entered. In typical workflow execution the default behavior satisfies the EC, but it could be that in the past the EC has been varied for enabling specific behavior.

The Transformation Algorithm. The transformation algorithm for translating from an annotated Ws-BPEL, see figure Fig. 6, to a standard one, see Fig. 7, is shown in the following:

1. For each configuration (each Context-Sensitive Region has some alternative configurations) the preprocessor builds $n + 1$ sub-configurations (where n is the number of migration points, the other sub-configuration is for the default behavior), each one starting from a different migration point (or from the start of the configuration) and ending at the end of the configuration.
2. All these sub-configurations become branch of a switch construct. This switch will be driven from $Ending\_mig\_point$ variable.
3. At this point there is a *Switch* construct for each configuration. Each configuration becomes a branch of another *Switch* construct. This latter switch will be driven from the behavior variable.
4. The *Switch* construct is put inside the *Scope* where the exception is managed. This fact ensures that at the end of exception management the flow is just outside the switch construct.
5. The *Scope*, that is the *Switch*, is put inside a *Loop*. This *Loop* is driven by $pass\_through$ variable, that is "if no exceptions have been raised the flow go away", otherwise "the flow go back to the start of the loop".
6. In the place of each starting migration point put an activity called $Update\ MPs$, necessary to maintain updated the $last\_MPs$ variable.

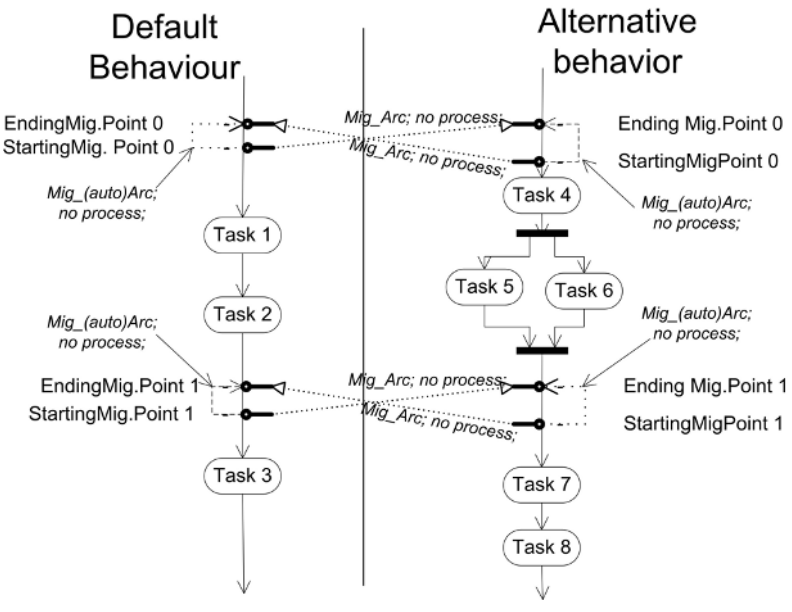


Fig. 6. Example of designer specification for a *Scope* supporting rollback execution

7. As first task of the *Loop*, before the beginning of the *Scope*, an activity called $UpdatePass\_through$ is put. The aim of this activity is to set $Pass\_through = true$, to allow the flow to go out of the loop if an exception has been raised previously.

To better understand the run-time behavior let us suppose that an exception is raised. The handler starts and the actions performed are:

1. Compensate until last migration point.
2. Set $Pass\_through = false$.
3. Kill the current *Scope* and start again the flow at his end.

Now let us suppose that during the execution of Task 3 (see Fig. 7), a recovery action that requires rollback has to be performed. This information is carried in a fault message and then when it arrives all the activities in the *Scope* are terminated and the Fault Handler is called. The Fault Handler sets the variable *pass_through* to false, then performs the recovery actions associated with the rollback process, determines the new configurations that will be followed (again "Default" or "Alternative")and the $Ending\_Migration\_point$ corresponding to the last $Starting\_Migration\_Point$ (that is the Safe Point) upon which the flow is passed. Eventually, in case, perform the actions associated with the migration process.

The main flow will restart immediately after the end of the *Scope* when the Fault Handler ends. The variable that drives the loop construct has been set for performing another iteration and then the main flow will go back to the start

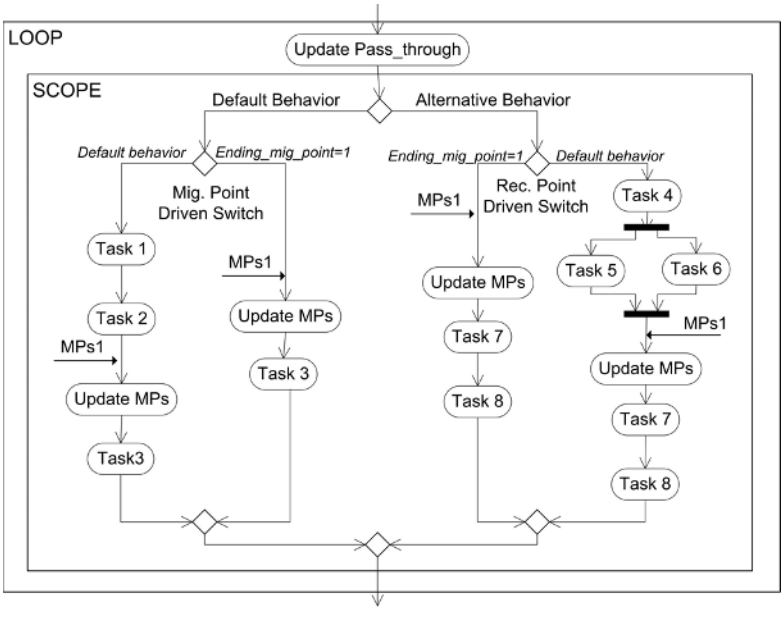


Fig. 7. Standard Ws-BPEL implementing specification of Fig. 6

of the loop. The first activity in the loop (Update *pass_through*) is necessary to allow the flow exiting the loop. Then the first switch will drive the choice of the right configuration and the second, driven from the *Ending_migration_point*,will determine which is the right point where the flow has to start in the new configurations.

If the new configuration is the same of the previous one, the system is performing a traditional rollback operation.

4.6 Harmonizations of Proposed Mechanisms

The changes of semantic applied to many constructs using our mechanisms are very profitable, but attention has to be put in considering possible conflicts among mechanisms. In fact final Ws-BPEL code is much more complicated then the original one.

Some basic policies has been followed in defining mechanisms:

- Each mechanism is raised by a different message for avoiding non-determinism.
- Each mechanism is independent and self-contained, that is different mechanisms do not communicate and therefore, given that each mechanism terminates, deadlock introduced by cyclic and reciprocal calls of different mechanisms is prevented.
- The definition of region upon which mechanisms are defined is the same of Ws-BPEL *Scope*. This ensures that partial overlapping among them are not allowed.

This three policies and general considerations about mechanism behaviors lead us to be sure that conflicts can be avoided.

Moreover we assume the hypothesis that the engine support a recovery mechanism at once. This choice definitively improves robustness and can be realized blocking, and in case buffering, incoming message using an apposite firewall.

5 Related Work

Recovery actions for workflow systems have been wide studied in the past. The works [6,8,18] present specific workflow models that widely support recovery actions; in [12] the authors focus on the analysis, prediction, and prevention of exceptions in order to reduce their occurrences. The model presented in [13] focuses on the handling of expected exceptions and the integration of exception handling in the execution environment, while in [5] the authors propose the use of "worklets", a repertoire of self-contained subprocesses and associated selection and exception handling rules to support the modelling, analysis and enactment of business processes. The work in [9] presents the requirements of a Web Service Management framework which also includes the typical functionalities addressed in self-healing systems. The authors analyze and compare multiple alternative architectures for the implementation of Web Service Management systems proposing Web service substitution and complex service re-compositions as repair actions.

In addition, an extensive amount of work on flexible recovery in the context of advanced transaction models has been done, e.g., in [10,19]. They particularly show how some of the concepts used in transaction management can be applied to workflow environments.

A "minimal" approach to recovery can be built with BPML [4] that uses a Petri nets based model focusing on flexibility. For this reason recovery actions are viewed as a transition from the actual (faulty) state to a new (correct) one, but constraints in state transitions for guarantying the correctness of recovery action have to be defined by the designer. High flexibility is ensured but the effort required becomes too cumbersome.

A specific comparison will be carried out with the systematic approach to recovery actions presented in [14]. In this work the authors consider a set of recovery policies both on task and region of a workflow. They use an extend Petri Net approach to change the normal behaviour when an expected but unusual situation or failure occurs. As in our approach the recovery policies are set at design time.

Table 1 shows how the mechanisms presented in Hamadi's work can be realized using Ws-BPEL. Several solutions use standard Ws-BPEL handlers, others are realized exploiting the mechanisms proposed in this paper.

When Hamadi uses the term "after", he defines the moment after finishing the execution of task/region and before initiating any subsequent task/region. The terms region in his work is the same of *Scope* in our approach.

Table 1. Comparison between recovery actions presented in [14] and the proposed mechanisms

Recovery Policy in [14]	**Proposed solution Task level**	**Proposed solution Scope level**
Redo	Redo mechanism	Redo policy
RedoAfter	Redo mechanism	Redo mechanism
Compensate	Compensation Handler	Compensation Handler
CompesateAfter	Compensation Handler	Compensation Handler
AlternativeActivity	Catch fault	rollback + Fault Handler
AlternativeProvider	Dynamic Binding	———
Skip	Catch fault empty	Fault Handler
SkipTo	Not supported	Not supported
Timeout	TimeOut mechanism	TimeOut mechanism

The meaning of "redo" actions are analogous in our approach and in the Hamadi's one. With the mechanism presented in Section 4.3 it is possible to have the same behavior.

The "compensation" is realized in Ws-BPEL by Compensation Handler.

According to Hamadi, the "Alternative recovery policy" allows another task/region T' to be executed in place of a running task/region T in case the later fails. The proposed mapping with Ws-BPEL distinguishes between task and region. In the first case the solution is simple and it is exploited by a Fault Handler opportunely filled with the alternative behavior. In the second case an analogous behavior can be realized using again a Fault Handler filled with the alternative behavior or it can be better exploited using the mechanism of Rollback and conditional re-execution of the flow. This mechanism allows the designer to define more than one alternative behavior linking the choice of this behavior to the current point during the workflow execution where the fault happens.

"Alternative Provider" for the single tasks can be exploited in Ws-BPEL using some mechanisms for dynamic binding provided by several Ws-BPEL engines however this is not supported by standard Ws-BPEL. It is still not possible to implement an analogous behavior to the pattern of Alternative Provider at *Scope* level.

The "skip" is mapped in Ws-BPEL with an empty Fault Handler defined upon the single Task or the *Scope*.

The "skip to" is not mapped on Ws-BPEL because it does not support any way to realize a free goto operation.

The "timeout" is implemented in Ws-BPEL using the corresponding method presented in Section 4.2.

The other mechanisms presented in our work have not a direct mapping, this is compliant with the idea of the authors of [14] stating that recovery patterns should be extended and improved to cope with more complex situations.

6 Conclusion and Future Work

In this paper some mechanisms for enabling recovery actions using standard Ws-BPEL language and engine has been presented.

Ws-BPEL is the de-facto standard language for Web-based process orchestration an the possibility of overcoming several limitations about recovery without modifying it is really a need because Self-Healing are an emerging exigence for Information Systems where processes are everyday more complicated and where many autonomous actors are involved.

The proposed mechanisms cover a wide range of possibility: i) the specification of a time deadline associated with a task, ii) the ability of redoing a single Task or an entire *Scope*, iii) the possibility of specifying alternative paths to follow in the prosecution after the reception of an enabling message, iv) the possibility of going back in the process to a point defined as safe for redoing the same set of tasks or for performing an alternative path.

Ongoing research is spread in several direction: the improvement of "Future alternative behavior" mechanism for defining a suitable set of rules allowing to design the alternative behavior in a parametric way with respect to the default behavior, the implementation of an efficient preprocessor, the demonstration of absence of conflict among mechanisms.

Future research will cover the development of a graphical tool for Ws-BPEL annotation and the study of new and more flexible mechanism that should allow the freezing, and even killing, of a Ws-BPEL instance for creating a new one that should inherit the state of the dead one. An orthogonal aspect in the future work will be the enrichment of the simple and prototypical version of the used ontology of faults and recovery actions.

Finally, interesting use of presented mechanisms can also be envisaged in developing of Self-Healing workflow engine in an advanced Self-Healing Information System. In this scenario they can be leveraged in more complex recovery strategies, decided somewhere in the environment, and composed of a part in charge of the Ws-BPEL engine (i.e. these mechanisms) and of a part hidden to it and managed outside.

Acknowledgement

This work is partially funded by by EU Commission within the FET-STREP Project WS-Diamond. The authors are grateful to Prof. Fugini for the fruitful discussions.

References

1. http://www.alphaworks.ibm.com/tech/bpws4j, 2002.
2. http://www.oracle.com/technology/products/ias/bpel/index.html, 2003.
3. http://www.activebpel.org, 2004.
4. A. Arkin et al. Business process modeling language BPML 1.0, 2002.

5. M. Adams, A.H.M. ter Hofstede, D. Edmond, and W.M.P. van der Aalst. Facilitating flexibility and dynamic exception handling in workflows through worklets. In *Short Paper Proceedings at (CAiSE)*, volume 161 of *CEUR Workshop Proceedings*, Porto, Portugal, 2005.
6. F. Casati, S. Ceri, B. Pernici, and G. Pozzi. Workflow evolution. *Data Knowl. Eng.*, 24(3):211–238, 1998.
7. F. Curbera, Y. Goland, J. Klein, F. Leymann, D. Roller, S. Thatte, and S. Weerawarana. *Business Process Execution Language for Web Services, version 1.0*, 2002. http://www.ibm.com/developerworks/library/ws-bpel/.
8. J. Eder and W. Liebhart. Workflow recovery. In *Proc. of IFCIS Int. Conf. on Cooperative Information Systems (CoopIS)*, pages 124 – 134, Brussels, Belgium, 1996. IEEE.
9. E. Esfandiari and V. Tosic. Towards a web service composition management framework. In *Proc. of Int. Conf. on Web Services (ICWS)*, Orlando FL, USA, 2005.
10. D. Georgakopoulos, M.F. Hornick, and F. Manola. Customizing transaction models and mechanisms in a programmable environment supporting reliable workflow automation. *IEEE Trans. Knowl. Data Eng.*, 8(4):630–649, 1996.
11. P. Grefen, B. Pernici, and G. Sanchez (Eds). *Database Support for Workflow Management - The WIDE Project*. Kluwer Academic Publishers, 1999.
12. D. Grigori, F. Casati, U. Dayal, and M.C. Shan. Improving business process quality through exception understanding, prediction, and prevention. In *Proc. of Proceedings of Int. Conf. on Very Large Data Bases (VLDB)*, Roma, Italy, 2001.
13. C. Hagen and G. Alonso. Exception handling in workflow management systems. *IEEE Trans. Software Eng.*, 26(10):943–958, 2000.
14. R. Hamadi and B. Benatallah. Recovery nets: Towards self-adaptive workflow systems. In *Proc. of Int. Conf. on Web Information Systems Engineering (WISE)*, pages 439–453, Brisbane, Australia, 2004.
15. J. Miller, K. Verma, P. Rajasekaran, A. Sheth, R. Aggarwal, and K. Sivashanmugam. Adding semantics to wsdl. *White paper*, 2004. http://lsdis.cs.uga.edu/library/download/wsdl-s.pdf.
16. S. Modafferi, B. Benatallah, F. Casati, and B. Pernici. A methodology for designing and managing context-aware workflows. In *Proc. of IFIP TC 8 Working Conference on Mobile Information Systems (MOBIS)*, Leeds, UK, 2005.
17. B. Pernici (Ed). *Mobile Information Systems Infrastructure and Design for Adaptivity and Flexibility*. Springer, 2006.
18. M. Reichert, S. Rinderle, U. Kreher, and P. Dadam. Adaptive process management with ADEPT2. In *Proc. of Int. Conf. on Data Engineering ICDE*, pages 1113–1114, Tokyo, Japan, 2005.
19. H. Wächter and A. Reuter. The ConTract model. In A.K. Elmagarmid, editor, *Database Transaction Models for Advanced Applications*, pages 219–263. Morgan Kaufmann, 1992.

BPEL Processes Matchmaking for Service Discovery

Juan Carlos Corrales, Daniela Grigori, and Mokrane Bouzeghoub

Prism, Universite de Versailles Saint-Quentin en Yvelines
45 avenue des Etats-Unis, 78035 Versailles Cedex, France
Juan-Carlos.Corrales-Munoz@prism.uvsq.fr,
Daniela.Grigori@prism.uvsq.fr,
Mokrane.Bouzeghoub@prism.uvsq.fr

Abstract. The capability to easily find useful services (software applications, software components, scientific computations) becomes increasingly critical in several fields. Current approaches for services retrieval are mostly limited to the matching of their inputs/outputs. Recent works have demonstrated that this approach is not sufficient to discover relevant components. In this paper we argue that, in many situations, the service discovery should be based on the specification of service behavior. The idea behind is to develop matching techniques that operate on behavior models and allow delivery of partial matches and evaluation of semantic distance between these matches and the user requirements. Consequently, even if a service satisfying exactly the user requirements does not exist, the most similar ones will be retrieved and proposed for reuse by extension or modification. To do so, we reduce the problem of behavioral matching to a graph matching problem and we adapt existing algorithms for this purpose. A prototype is presented which takes as input two BPEL models and evaluates the semantic distance between them; the prototype provides also the script of edit operations that can be used to alter the first model to render it identical with the second one.

Keywords: web services, services retrieval, behavioral matching.

1 Introduction

The capability to easily find useful services (software applications, software components, scientific computations) becomes increasingly critical in several fields. Examples of such services are numerous:

- Software applications as web services which can be invoked remotely by users or programs. One of the problems arising from the model of web services is the need to put in correspondence service requesters with service suppliers, especially for services which are not yet discovered or which are new, taking into account the dynamic nature of the Web where services are frequently published, removed or released.
- Programs and scientific computations which are important resources in the context of the Grid, sometimes even more important than data [1]. In such a system, data and procedures are first rank classes which can be published, searched and handled. Thus, the scientists need to retrieve procedures with desired characteristics, to determine if a required calculation was already carried out and whether it is more advantageous to carry it out again or to retrieve data generated previously.

R. Meersman, Z. Tari et al. (Eds.): OTM 2006, LNCS 4275, pp. 237–254, 2006.

– Software components which can be downloaded to create a new application. To reduce the development, test and maintenance costs, a fast solution is to re-use existing components.

In all these cases, users are interested in finding suitable components in a library or collection of models. User formulates a requirement as a process model; his goal is to use this model as a query to retrieve all components whose process models match with a whole or part of this query. If models that match exactly do not exist, those which are most similar must be retrieved. For a given task, the models that require minimal modifications are the most suitable ones. Even if the retrieved models have to be tailored to the specific needs of the task, the effort for the tailoring will be minimal.

In this paper we argue that, in many situations, the service discovery process requires a matchmaking phase based on the specification of the component behavior. The idea behind is to develop matching techniques that operate on behavior models and allow delivery of partial matches and evaluation of semantic distance between these matches and the user requirements. Consequently, even if a service satisfying exactly the user requirements does not exist, the most similar ones will be retrieved and proposed for reuse by extension or modification. To do so, we reduce the problem of service behavioral matching to a graph matching problem and we adapt existing algorithms for this purpose.

In the next section we present several motivating scenarios. Section 3 presents existing approaches for service retrieval and shows their drawbacks for the presented scenarios. In section 4 we show how the behavioral matching is reduced to a graph matching problem; a similarity measure is defined based on graph edit distance for which two new graph edit operations are introduced. We show also how to combine two graph models in order to satisfy user requirements. Section 5 shows how the graph matching algorithm can be used for BPEL process matchmaking. In section 6 we present an experimental study of the matchmaking algorithm. Finally section 7 present ongoing work and conclusions.

2 Motivating Scenarios

In this section we present two scenarios requiring behavioral matchmaking. The first example situates in the context of web services integration and consists in retrieving services having compatible behavior. The second example is delta analysis which consists of finding differences between two models.

Web services integration. Consider a company that uses service S to order office suppliers. Suppose that the company wants to find retailers (say WalMart or Target) having compatible web services (a new retailer or replacing the current partner). The allowed message exchange sequences are called business protocols and can be expressed for example using BPEL abstract processes, WSCL, or other protocol languages (see, e.g., [2]). The specification of the business protocol is important, as it rarely happens that service operations can be invoked independently from one another. Thus the company will search for a service having a compatible business protocol. Among retailer services, the most compatible one has to be found. If the service is not fully compatible,

the company will adapt its service or will develop an adaptor in order to interact with the retrieved service. In both situations, the differences between the business protocols have to be automatically identified. In the former case, finding the most similar service allows to minimize the development cost. In the latter case, identifying automatically the differences between protocols is the first stage in the process of semi-automatically developing adapters (see [3]).

Delta-analysis. Delta analysis consists in finding the differences between two models. For example, the first one is the model specified by a standard and the second one is the model as it is implemented in an enterprise. Business definitions can be specified by industry specific standards groups in the same way that, for example, RosettaNet PIPs are specified by RosettaNet and used by participating enterprises. Enterprises need to verify if their services follow the guidelines prescribed by the standards. Thus, they need to compare the business model of their existing service with that prescribed by the standards. Ideally a tool should identify all the differences between the two models. Based on these differences the cost of reengineering of the existing service could be evaluated.

3 Related Work

Currently, the algorithms for Web services discovery in registers like UDDI or ebXML are based on a search by key words or tables of correspondence of couples (key-value). Within the framework of the semantic Web, description logics were proposed for a richer and precise formal description of services. These languages allow the definition of ontologies, such as for example OWL-S, which are used as a basis for semantic matching between a declarative description of the required service and descriptions of the services offered ([4,5,6]). In [4,6], a published service is matched with a required service when the inputs and outputs of the required service match the inputs and outputs of the published service (i.e., they have the same type or one is a generalization of the other). In [7], independent filters are defined for service retrieval: the name space, textual description, the domain of ontology that is used, types of inputs/outputs and constraints. The approach presented in [8] takes into account the operational properties like execution time, cost and reliability. The authors of [9] provide a lightweight semantic comparison of interfaces based on similarity assement methods (lexical, attribute, interface and QoS similarity).

In the context of the Grid [1], the search of procedures is based on a high-level language which expresses the relationships among procedures and their input/output data.

Service retrieval based of key words or some semantic attributes is not satisfactory for a great number of applications. The tendency of recent work is to exploit more and more knowledge on service components and behavior. The need to take into account the behavior of the service described by a process model was underlined by several researchers [10,11,5,12,13,14]. In [5], in order to improve precision of web service discovery, the process model is used to capture the salient behavior of a service. A query language for services is defined which allows to find services by specifying conditions

on the activities which compose them, the exceptions treated, the flow of the data between the activities.

In [11], authors argue that the matchmaking based on service input and output is not sufficient as some output data may be produced only under certain internal conditions. Thus, they propose an algorithm that matches output data taking into account the process structure, for instance conditional branching.

In [10], authors underline the importance of including behavior aspects in matchmaking process in the B2B environment and mention it as a future work. The authors of [13], which propose a model for dynamic service aggregation, stress also the capability to automatically verify the behavioral compatibility of various processes as a requirement in electronic marketplaces.

In [12], authors deal with the equivalence of two processes modelled using Petri nets. It is supposed that partners discover each other by searching in business registry, and then agree on a common protocol. Their work verifies the compatibility between the agreed protocol and the process existing in the enterprise. We take a different approach, by allowing to find a partner that is fully or partially compatible to an existing enterprise process.

Very recently, authors in the academic world have published papers that discuss similarity and compatibility at different levels of abstractions of a service description (e.g., [15,16,17,14]). In terms of protocols specification and analysis, existing approaches provide models (e.g., based on pi-calculus or state machines) and mechanisms to compare specifications (e.g., protocols compatibility checking).

In [14], authors give a formal semantics to business process matchmaking based on finite state automata extended by logical expressions associated to states. Computing the intersection is computationally expensive, and thus does not scale for large service repositories. To solve this problem, the authors of [18] present an indexing approach for querying cyclic business processes using traditional database systems; they introduce an abstraction function that removes cycles and transforms a potentially infinite set of message sequences into a finite representation, which can be handled by existing database systems. The choice of finite state automata as a modelling formalism limits the expressiveness of the models, for instance representing parallel execution capabilities can lead to very large models.

A new behavior model for web services is presented in [19] which associates messages exchanged between participants with activities performed within the service. Activity profiles are described using OWL-S (Web Services Ontology Language). Web services are modelled like non-deterministic finite automatons. A new query language is developed that expresses temporal and semantic properties on services behaviors.

To summarize, the need to take into account the service behavior in the retrieval process was underlined by several authors and some very recent proposals exist ([19],[14]). The few approaches that exist give a negative answer to the user if a model satisfying exactly his requirements does not exist in the registries, even if a model that requires a small modification exists. Our objective is to propose an approach for service retrieval based on behavioral specification allowing an approximate match. To the best of our knowledge, there is not another approach allowing to retrieve services having similar behavior and defining a behavior-based similarity measure.

4 A Graph-Based Approach to Behavior Matchmaking

In this section we show how the behavioral matching is reduced to a graph matching problem. Section 4.1 recalls the principles of the graph matching method that we use, the error correcting subgraph isomorphism, which is based on the idea of graph edit operations. Next sections show how we adapt it to our problem: we extend the set of graph edit operations, we define a similarity measure for behavior matchmaking and we show how to compose two library graphs to satisfy user requirements.

A business protocol describes the observable behavior of a web service. It complements the web service interface definition by imposing constraints on the order of exchanged messages. Most of existing proposals (standards and research models) are graph based. For this reason, we choose to use a graph representation of business protocols in order to compare two models.

Using graphs as a representation formalism for both user requirements and service models, the service matching problem turns into a graph matching problem. We want to compare the process graph representing user requirements with the model graphs in library. The matching process can be formulated as a search for graph or subgraph isomorphism. However, it is possible that there does not exist a process model such that an exact graph or subgraph isomorphism can be defined. Thus, we are interested in finding process models that have similar structure if models that have identical structure do not exist. The error-correcting graph matching integrates the concept of error correction (or inexact matching) into the matching process ([20,21]). To make the paper self-contained, in the next section we briefly recall the principle of this graph matching method and the basic definitions as given in [22].

4.1 Background and Basic Definitions

In order to compare the model graphs to an input graph and decide which of the models is most similar to the input, it is necessary to define a distance measure for graphs. Similar to the string matching problem where edit operations are used to define the string edit distance, the subraph edit distance is based on the idea of edit operations that are applied to the model graph. Edit operations are used to alter the model graphs until there exist subgraph isomorphism to the input graph. For each edit operation, a certain cost is assigned. The costs are application dependent and reflect the likelihood of graph distortions. The more likely a certain distortion is to occur the smaller is its cost. The subgraph edit distance from a model to an input graph is then defined to be the minimum cost taken over all sequences of edit operations that are necessary to obtain a subgraph isomorphism. It can be concluded that the smaller the subgraph distance between a model and an input graph, the more similar they are.

In the following we give the definitions of error correcting graph matching as given in [22].

A directed labelled graph is defined by a quadruple $G = (V, E, \alpha, \beta)$ where V is the set of vertices, $E \subset V \times V$ is the set of edges, $\alpha : V \rightarrow L_V$ is the vertex labelling function and $\beta : E \rightarrow L_E$ is the edge labelling function.

Definition 1. Graph isomorphism. *Let g and g' be graphs. A graph isomorphism between g and g' is a bijective mapping $f : V \rightarrow V'$ such that*

- $\alpha(v) = \alpha'(f(v))$ *for all* $v \in V$
- *for any edge* $e = (u, v) \in E$ *there exists an edge* $e' = (f(u), f(v)) \in E'$ *such that* $\beta(e) = \beta'(e')$ *and for any edge* $e' = (u', v') \in E'$ *there exists an edge* $e = (f^{-1}(u'), f^{-1}(v')) \in E$ *such that* $\beta(e) = \beta'(e')$.

If $f : V \rightarrow V'$ is a graph isomorphism between graphs g and g', and g' is a subgraph of another graph g", i.e. $g' \subset g$", then f is called a subgraph isomorphism from g to g".

Given a graph G, a graph edit operation δ on G is any of the following:

∘ substituting the label $\alpha(v)$ of vertex v by l
∘ substituting the label $\beta(e)$ of edge e by l'
∘ deleting the vertex v from G (for the correction of missing vertices). Note that all edges that are incident with the vertex v are deleted too.
∘ deleting the edge e from G (for the correction of missing edges).
∘ inserting an edge between two existing vertices (for the correction of extraneous edges).

Definition 2. Edited graph. *Given a graph and an edit operation* δ *, the edited graph* $\delta(G)$ *is a graph in which the operation* δ *was applied. Given a graph* G *and a sequence of edit operations* $\Delta = (\delta_1, \delta_2, \cdots \delta_k)$*, the edited graph* $\Delta(G)$ *is a graph* $\Delta(G) = \delta_k(\cdots \delta_2(\delta_1(G)))..)$.

Definition 3. Ec-subgraph isomorphism. *Given two graphs* G *and* G'*, an error correcting (ec) subgraph isomorphism* f *from* G *to* G' *is a 2-tuple* $f = (\Delta, f_\Delta)$ *where* Δ *is a sequence of edit operations and* f_Δ *is a subgraph isomorphism from* $\Delta(G)$ *to* G'.

For each edit operation δ, a certain cost is assigned $C(\delta)$. The cost of an ec-subgraph isomorphism $f = (\Delta, f_\Delta)$ is the cost of the edit operations Δ, i.e., $C(\Delta) = \sum_{i=1}^{k} C(\delta_i)$. Usually, there is more than one sequence of edit operations such that a subgraph isomorphism from $\Delta(G)$ to G' exists and, consequently, there is more than one ec-subgraph isomorphism from G to G'. We are interested in the ec-subgraph isomorphism with minimum cost.

Definition 4. Subgraph edit distance. *Let* G *and* G' *be two graphs. The subgraph distance from* G *to* G'*,* $ed(G, G')$ *is given by the minimum cost taken over all error-correcting subgraph isomorphism* f *from* G *to* G'.

4.2 Extension of the Sub-graph Edit Distance

The models to be compared can have different granularity levels for achieving the same functionality. For example, the first service has a single operation (activity) to achieve certain functionality, while in the second service the same behavior is achieved by composing several operations. Thus, new edit operations are required. Given a graph G, we extend the definition of edit operation δ on G by adding two operations:

∘ decomposing a vertex v into two vertices v_1, v_2
∘ joining two vertices v_1, v_2 into a vertex v.

We limit ourselves to a simple case of decomposition, when a vertex is decomposed into a sequence of two vertices. This simple type of decomposition is sufficient for

applications that we analyzed. A more general decomposition operation would be to decompose a vertex into a connected subgraph, this is subject of future work.

The operation of decomposing a vertex v into two vertices v_1, v_2 is executed in the following way :

- all the edges having as destination the vertex v will have as destination the vertex v_1;
- all edges having as source the vertex v, will have as source the vertex v_2;
- an edge between the vertex v_1 and v_2 will be added.

The joining operation is executed in a similar way. These two new edit operations allow to model one-to-many dependencies among vertices of two graphs (i.e., a vertex in one graph corresponds to two vertices in the second graph). The classical edit operations take into account only one-to-one mappings between vertices of the two graphs. For example, if a vertex v in the first graph corresponds to the composition of two vertices in the second graph ($v1$ followed by $v2$), a matching algorithm based on the classical edit distance would map v to $v1$ and suppress $v2$. It would not be possible to discover that v is mapped to a composition of $v1$ and $v2$.

4.3 Similarity Measure for Behavioral Matching

The subgraph edit distance defined previously expresses the cost of transformation needed to adapt the model graph in order to cover a subgraph in the input model. This distance is asymmetric, it represents the distance from the model graph to the input graph. In order to rank the model graphs, the similarity measure has to take into account the number of vertices in the input graph that were covered by the model graph. If two model graphs have the same subgraph distance to the input graph but are matched to subgraphs with different number of nodes, the one that matches a subgraph with more nodes will be preferred.

Depending on the application, the similarity measure can be defined in different ways. The total distance between the two graphs can be defined as the sum of the subgraph edit distance and the cost of adding the nodes of the input graph not covered by the ec-subgraph isomorphism. The second possibility is to define it as the subgraph edit distance (ed) divided by the number of nodes of model graph (N_M): $D = ed/N_M$. The similarity measure is the inverse of this distance ($S = 1/D$).

4.4 Composing Fragments to Match the Input Graph

If two models are matched into 2 subgraphs of the input model that are disjoint (the set of nodes are disjoint), then it is possible to combine them to form a graph that will be matched to a larger subgraph of the input graph.

Suppose two model graphs G_1 and G_2 that are disjoint. Let $f_1 = (\Delta_1, f_{\Delta_1})$ and $f_2 = (\Delta_2, f_{\Delta_2})$ be two ec-subgraph isomorphism from G_1 and G_2 to G_I, respectively. The problem is to find an ec-subgraph isomorphism from $G = G_1 \cup G_2$ to G_I that is based on f_1 and f_2. f_1 and f_2 can be combined if no two vertices in $\Delta(G_1)$ and $\Delta(G_2)$ are mapped into the same input vertex. More precisely, the intersection of the images of f_{Δ_1} and f_{Δ_2} must be empty, i.e., $f_{\Delta_1}(V_1) \cap f_{\Delta_2}(V_2) = \emptyset$.

The construction of an ec-subgraph isomorphism $f = (\Delta, f_\Delta)$ from f_1 and f_2 requires that a set of edit operations Δ and a subgraph isomorphism f_Δ are generated on the basis of Δ_1, Δ_2, and f_{Δ_1}, f_{Δ_2} respectively such that Δ is a subgraph isomorphism from $(G_1 \cup G_2)$ to G_I. Let $f_1 = (\Delta_1, f_{\Delta_1})$ and $f_2 = (\Delta_2, f_{\Delta_2})$, $\Delta_1(G_1) = (V_{\Delta_1}, E_{\Delta_1}, \alpha_{\Delta_1}, \beta_{\Delta_1})$. Then :

$$f_\Delta(v) = \begin{cases} f_{\Delta_1}(v)\ if\ \ v \in V_{\Delta_1} \\ f_{\Delta_2}(v)\ if\ \ v \in V_{\Delta_2} \end{cases}$$

and $\Delta = \Delta_1 + \Delta_2 + \Delta_E$. Δ_E is constructed as follows. For each pair $(v_i, w_j), v_i \in V_1, w_j \in V_2$, if there is an edge $e_I = (f_{\Delta_1}(v_i), f_{\Delta_2}(w_j)) \in G_I$, then an edge (v_i, w_j) must be inserted in Δ_E.

If C_1 and C_2 are the cost of the ec-subgraph isomorphism f_1 and f_2 respectively, then the cost of the ec-subgraph isomorphism f is : $C(f) = C_1 + C_2 + C(\Delta_E)$.

To summarize, if two models G_1 and G_2 cover disjoint subgraphs in the input graph, it is possible to construct a model that is their composition in order to find a better match for the input graph.

5 BPEL Processes Matchmaking

In this section we illustrate the use of the error-correcting graph matching algorithm for BPEL processes matchmaking. We first give an overview of the matchmaking process and then we discuss each step in detail; finally, we illustrate it using an example.

We choose to exemplify our approach for business protocol matchmaking by using the BPEL model. The same approach can be applied for other models, as long as the business protocol can be transformed to a graph in a unique way (equivalent representations of a business protocol are reduced to the same process graph).

BPEL [23] has emerged as a standard for specifying and executing web services-based processes. It supports the modelling of two types of processes: executable and abstract processes . An *abstract process* is a business protocol, specifying the message exchanges between different parties from the perspective of a single organization (or composite service), whitout revealing the internal behavior. An *executable process*, in contrast, specifies the actual behavior of a participant. A BPEL process is composed of a set of activities, that can be either primitive or structured. Primitive activities include operations involving web services like the *invoke*, the *receive* and the *reply* activity. There are further activities for assigning data values for variables (*assign*) or *wait* to halt the process for a certain time interval. Structured activities are used for defining the control flow, e.g. to specify concurrency of activities using *flow*, alternative branches (*switch*) or sequential execution (*sequence*). These structured activities can be nested. Beyond that, *links* can be used to specify order constraints between activities composing a *flow*, similar to control flow arcs.

BPEL builds on IBM's WSFL and Microsoft's XLANG and combines thus the features of a block structured language (XLANG) with those for directed graphs (WSFL). As a result there are sometimes two equivalent ways to implement a desired behavior. For exemple, a sequence can be realised using a *sequence* or a *flow* with a link between

activities, a choice based on certain data values can be realised using the switch or flow elements, etc.

For this reason, we transform a BPEL process to a process graph and thus equivalent constructs that are syntactically different will be transformed to the same graph fragment. In the following we concentrate on the matchmaking of BPEL abstract processes, but the approach can be adapted to matching executable processes.

The BPEL matchmaking process is composed of the following steps. First, the BPEL processes to be compared are transformed to graphs. Next, the graph matching algorithm is applied taking into account the comparison rules and possibly the nodes decomposition. Finally the similarity result is shown.

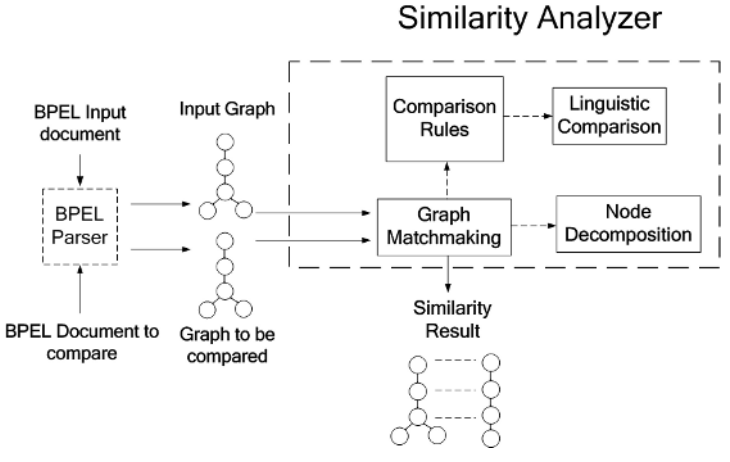


Fig. 1. Architecture

The architecture of the behavior matchmaking system is presented in Figure 1. The system is composed of a parser that transforms a BPEL document into a graph and a similarity analyzer module that evaluates the similarity between the graphs. The similarity analyzer is composed of the following elements (Figure 1):

- *Graph matchmaking*, that takes as inputs the graphs produced by the BPEL parser and finds out the error correcting sub-graph isomorphism with minimal cost.
- *Comparison rules module*, that groups the cost functions for the graph edit operations.
- *Decomposition module*, that applies the decomposition operation if it is necessary in order to have the same level of granularity in both models.
- *Linguistic comparison*, that implements different algorithms useful to find the similarity between two strings.

In the next sections we present in detail the functionalities of each module.

5.1 Transforming BPEL to Graph

The parser function transforms a behavior model into a process graph. A process graph has at least one start nodes and can have multiple end nodes. The graph has two kind of nodes : regular nodes representing the activities and connectors representing split and join rules of type XOR or AND. Nodes are connected via arcs which may have an optional guard. Guards are conditions that can evaluate to true or false.

We implemented the flettening strategy presented in [24] to transform a BPEL to a process graph. The general idea is to map structured activities to respective process graph fragments. The algorithm traverses the nested structure of BPEL control flow in a top-down manner and applies recursively a transformation procedure specific to each type of structured activity.

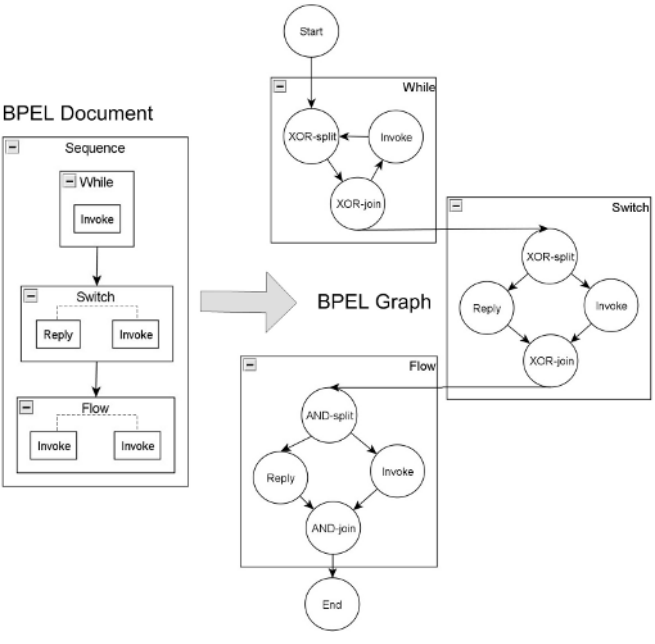


Fig. 2. Correspondences between BPEL elements and graph elements

A *BPEL basic activity* is transformed to a node. The *sequence* is transformed by connecting all nested activities with graph arcs; each sub-activity is then transformed recursively. For the *while* activity a loop is created between an XOR join and an XOR split, the condition is added to the edge. The graph representation of *switch* consists of a block of alternative branches between an XOR split and an XOR join. The branching conditions are associated to edges. The *flow* is transformed to a block of parallel branches starting with an AND split and synchronized with an AND join.

The nodes that represent the activities have the following attributes: Operation and PortType. The connector nodes have two attributes: ConnectorType (AND-split,

AND-join, XOR-split, XOR-join) and ActivityType (the BPEL structured activity from which it was transformed). Figure 2 shows the correspondence between BPEL constructs and graph elements.

5.2 Graph Matchmaking

This module implements the algorithm for error-correcting sub-graph isomorphism detection ([22]). The sub-graph isomorphism detection is based on a state-space searching using an algorithm similar to A* [20]. The basic idea of a state-space search is to have states representing partial solutions of the given problem and to define transitions from one state to another, thus, the latter state represents a more complete solution than the previous state. For each state s there is an evaluation function $f(s)$ which describes the quality of the represented solution. The states are expanding themselves according to the value of f. In the case of each sub-graph isomorphism detection, given a model graph G and an input graph G_I, a state s in the search space represents a partial matching from G to G_I. Each partial matching implies a number of edit operations and their cost can be used to define the evaluation function $f(s)$. In other words, the algorithm starts by mapping the first node of G with all the nodes of G_I and chooses the best mapping (with minimal cost)(Algorithm1, line 1). This represents a partial mapping that will be extended by adding one node at a time (line 7). The process terminates when either a state representing an optimal ec-subgraph isomorphism from G to G_I has been reached or all states in the search space have edit costs that exceed a given acceptance threshold.

Algorithm 1. Error-correcting sub-graph isomorphism detection $(G(V), G_I(V_I))$

1: Initialize OPEN: map the activity node in V onto each activity node in V_I (call Activity-Match), i.e. create a mapping p. Calculate the cost of this mapping C(p) and add p to OPEN.
2: IF OPEN is empty THEN Exit.
3: Select p of OPEN such that C(p) is minimal and remove p from OPEN
4: IF C(p)> Accept threshold THEN Exit.
5: IF p represents a complete mapping from G to G_I THEN output p. Set accept threshold = C(p). Goto 2.
6: Let p be the current mapping that maps k nodes from G.
7: FOR *each activity node w in V_I that has not yet mapped to a corresponding node in V*
  7.1: extends the current mapping p to p' by mapping the k+1 node of V to w and calculate the cost of this mapping C(p')
  7.2: add p' to OPEN
8: Goto 2

The cost of the mapping $C(p')$ (line 7.1) represents the cost of extending the current mapping p with the next node in the model graph. Extending the mapping by mapping a vertex v (in the input graph that has not yet mapped) to a vertex w in the model graph (that does not belong to the current mapping) implies node edit operation and edge edit operations. First, the label and attributes of v must be substituted by label attributes of

w, and secondly, for each mapping $(v', w') \in p$ it must be ensured that any edge (v', v) in the model graph can be mapped to an edge (w', w) in the input graph by means of edge edit operations.

A process graph has two kind of nodes: activities and connectors. In contrast with activities, connectors do not represent business functions, they express control flow constraints. For this reason, in the matching process, we compare them in a manner similar to edges. That is, when mapping edges between two activity nodes, we map also the possible connectors binding directly to the nodes and calculate the corresponding edit cost.

5.3 Comparison Rules

The *Comparison rules module* contains all the application-dependent functions allowing to calculate the cost of graph edit operations. These functions are used by the graph matching module for calculating the distance between the graphs. In order to support applications with specialized cost function, user-defined cost function can be registered in this module. In the following we explain the cost functions used for BPEL protocol matchmaking. The cost for inserting, suppressing edges and vertices can be set to a constant. The cost for editing vertices (basic activities and connectors) are presented below.

Algorithm 2. Function BasicActivityMatch

INPUTS: (Nodei,Nodej)
Nodei: Struct (Opi, PTi), Nodej: Struct (Opj,PTj)
OUTPUT: $DistanceNode$

Calculate Operation Similarity $SimOperation = LS(Opi, Opj)$
if $SimOperation = 0$ (different Operations) **then**
  Return $DistanceNode = 1$
else
  Calculate PortType Similarity $SimPortType = LS(PTi, PTj)$
  Calculate $DistanceNode$

$$DistanceNode = 1 - \frac{w_{op} * SimOperation + w_{pt} * SimPortType}{w_{op} + w_{pt}}$$

end if

Matching basic activites. The cost for editing a basic activity vertex (*receive, invoke, reply*) is calculated by function BasicActivityMatch (see Algorithm 2). This cost expresses the distance between two BPEL basic activities. Each activity has two attributes: the Operation name (Op) and the PortType (PT). The matchmaking gives priority to operation comparison, and if two operations are similar ($SimOperation > 0$), it compares the similarity of the $PortType$ and calculates the distance between activities ($DistanceNode$).

Weights w_{op} and w_{pt} indicate the contribution of Op (similarity of Operations) and PT (similarity of PortTypes) respectively in the total $DistanceNode$ score ($0 \leq w_{op} \leq 1$ and $0 \leq w_{pt} \leq 1$).

Matching connectors. The connectors represent the control flow of process. The cost for editing a connector vertex is calculated by function ConnectorMatch (see Algorithm 3). This cost verifies if two nodes depict the same connectors. Each connector has two attributes: the Connector Type (CT) and the Activity Type (AT) that the connector represents.

Algorithm 3. Function ConnectorMatch

INPUTS: (Nodei,Nodej)
Nodei: Struct (CTi, ATi), Nodej: Struct (CTj, ATj)
OUTPUT: $DistanceNode$

if $CTi \neq CTj$ (different Connector Type) AND $ATi \neq ATj$ (different Activity Type) **then**
  Return $DistanceNode = 1$
else
  $DistanceNode = 0$
end if

Matching wait activities. This function (see Algorithm 4) calculates the cost for editing a vertex which represents a *wait* activity. Each wait vertex has two attributes: a delay for a certain period of time (F) or until a certain deadline is reached (U). The function checks if two $ForExpressions$ or two $UntilExpressions$ are similar, and gives a result for $DistanceNode$ respectively. The time similarity function (TS) calculates the resemblance between the time expressions.

Algorithm 4. Function WaitMatch

INPUTS: (Nodei,Nodej)
Nodei: Struct (Fi, Ui), Nodej: Struct (Fj, Uj)
OUTPUT: $DistanceNode$

if $ForExpression$ there exist **then**
  Calculate ForExpression Similarity $SimFor = TS(Fi, Fj)$
  Calculate $DistanceNode = 1 - SimFor$
else
  Calculate UntilExpression Similarity $SimUntil = TS(Ui, Uj)$
  Calculate $DistanceNode = 1 - SimUntil$
end if

5.4 Linguistic Comparison

The *Linguistic comparison module* calculates the linguistic similarity between two labels based on their names [25]. The labels are often formed by a combination of words and can contain abbreviations. To obtain a linguistic distance between two strings, we use existing algorithms: *NGram*, *Check synonym*, *Check abbreviation*, tokenization, etc. The *NGram* algorithm estimates the similarity according to a number of common *qgrams* between labels names [26]. The *Check synonym* algorithm uses a linguistic dictionary (e.g. Wordnet [27] in our implementation) to find synonyms while *Check abbreviation* uses a custom abbreviation dictionary.

If all algorithms return 1, there is an exact matching. On the other hand, if all the algorithms return 0, it means that there is no matching between labels. If the *NGram* value and the *Check abbreviation* value are equal to 0, and *Check Synonym* is between 0 and 1, the total linguistic similarity value will be equal to the *Check Synonym* one. Finally, if the three algorithms values are between 0 and 1, the similarity LS ([25]) is the average of them:

$$LS = \begin{cases} 1 & if\ (m1 = 1 \vee m2 = 1 \vee m3 = 1) \\ m2 & if\ (0 < m2 < 1 \wedge m1 = m3 = 0) \\ 0 & if\ (m1 = m2 = m3 = 0) \\ \frac{m1+m2+m3}{3} & if\ m1, m2, m3 \in (0, 1) \end{cases}$$

where, m1 = Sim(NGram), m2=Sim(Synonim Matching) and m3= Sim(Abbreviation Expansion).

5.5 Decomposing Vertices

The decomposition operations are applied in order to have the same granularity level in both models. The decomposition operation depends on the metamodel of the behavior models to be matched. For instance, for BPEL metamodel, it is possible that in one process a message exchange is modelled as an synchronous interaction, while in the second process is modelled as an asynchronous interaction. Figure 3 shows how a message exchange can be modelled as an asynchronous or synchronous interaction for an operation invoked by the process.

Synchronous interaction	**Asynchronous interaction**
Invoke (request/response)	Invoke (one way) + Receive

Fig. 3. Synchronous vs. asynchronous interactions

Therefore, an invoke operation of type request/response (having m_{in} as input message and m_{out} as output message) can be decomposed in an *invoke operation* (one way, having message m_{in}) and a *receive* operation (having m_{out} as message).

Another example of decomposition operation is related to the granularity of the exchanged message. For example, the first process requires messages *submitOrder* and *sendShippingPrefereces* separately, but the second process needs all of this information included in the *submitOrder* message. In this case, an *invoke* operation (having a message composed of two parts $m1$ and $m2$) will be decomposed in two *invoke* operations (having as messages $m1$ and $m2$, respetively).

These decomposition functions are specific to BPEL model. For other applications, user can specify a different decomposition function. The decomposition function has always the same signature: it takes as argument a vertex and returns two vertices resulting from decomposition (that are supposed to be sequential). The function behavior is specific to the application (metamodel of the protocols to be matched) specifying how the labels and attributes of the two vertices are obtained from the decomposed vertex.

5.6 Example

Suppose that we would like to find the similarity between two hotel reservation services whose models have been described using BPEL.

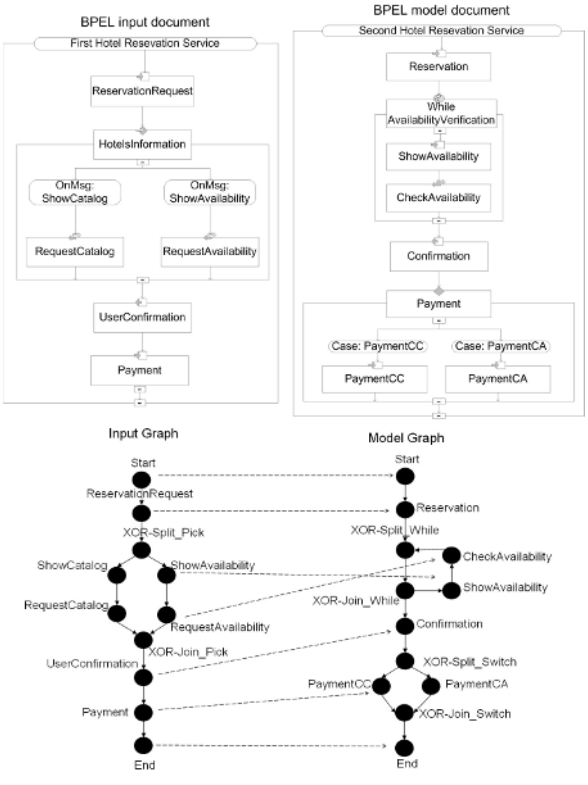


Fig. 4. Example

The first service has the following activities: First, the customer should place his *Reservation Request* (Activity type: Receive). Then the reservation service requires the *Hotels information* (Activity type: Pick), either Catalog (*RequestCatalog*, Activity type: Invoke) or Availability information (*RequestAvailability*, Activity type: Invoke). Next, a confirmation (*UserConfirmation* Type: Reply) is sent to user. Finally, the reservation service finishes the process by receiving the reservation *Payment* (Type: Receive). The second service model has the following activities sequence: first, the customer should place his *Reservation* (Activity type: Receive). Then, the hotel reservation service receives the customer reservation dates (*ShowAvailability* Type: Receive) and verifies the hotels availability (*CheckAvailability* Type: Invoke), until finding availabile rooms. Next, a confirmation (*Confirmation* Type: Reply) is sent to user. Finally, the hotel reservation service requires the customer to pay (*Payment* Type: Switch), either with credit card (are *PaymentCC* Type: Receive) or out of his checking account (*PaymentCA* Type:

Receive). Our system converts each BPEL document into a graph (input graph and model graph, Figure 4). Next, the graphs are compared by the similarity analyzer module. The dotted lines in Figure 4 represent the mappings found by the system between the two graphs using the comparison rules. In conclusion, the edit script will show that the two graphs are similar, but the activities *ShowAvailability*, *CheckAvailability* and *PaymentCC* of the model graph are parts of different structured activities in the input graph. However, the matchmaking algorithm will find similar activities for the right branch of the input graph (*Start*, *ReservationRequest*, *ShowAvailability*, *RequestAvailability*, *UserConfirmation*, *Payment* and *End*).

6 Implementation and Experiments

We implemented the first version of a desktop system having the architecture presented in the previous section. In this section, we present an experimental study of the matchmaking algorithm. The theoretical complexity of the graph matchmaking algorithm [22] is ($O(m^2n^2)$ in the best case (when the distance between the model and the input graph is minimal) and $O(m^n n)$ in the worst case (m = the total number of vertices in the input graph; n = the total number of vertices in the graph to be compared). The goal of the experiments is to find how well the algorithm performs for BPEL process matchmaking. Since most of the existing BPEL process have less than 50 activities, we considered a maximum of 50 activities.

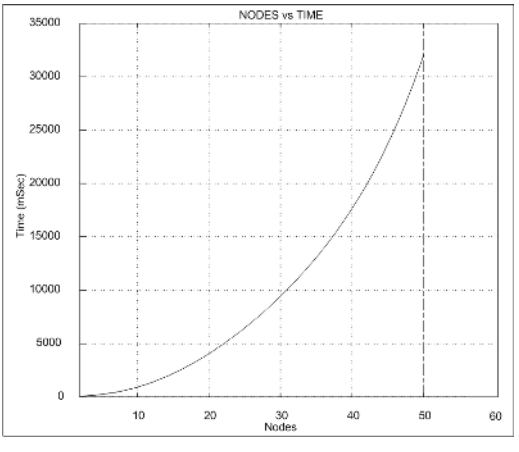


Fig. 5. Matchmaking two BPEL documents

Figure 5 shows the system behavior for two graphs with different structures and different names for activities operations and portTypes. For the comparison of operations and portTypes, the linguistic comparison is used. Despite the exponential theoretical cost, the graph shows that the matchmaking algorithm can be used for BPEL documents having less than 50 activities. The current implementation does not include the two new graph edit operations. We are currently investigating how to efficiently implement them and evaluating the supplementary cost.

7 Conclusion

In this paper we proposed a solution for service retrieval based on behavioral specification. First we motivated the need to retrieve services based on their behavior model. By using a graph representation formalism for services, we proposed to use a graph error correcting matching algorithm in order to allow an approximate matching. Starting from the classical graph edit distance, we proposed two new graph edit operations to take into account the difference of granularity levels that could appear in two models. We defined a similarity measure for behavior matchmaking and we showed how to combine fragments in order to satisfy user requirements. We exemplified our approach for behavior matching using the BPEL model. The behavioral matchmaking was implemented as a web service that takes as input the graph representations of two BPEL processes and calculates the degree of similarity between them and outputs also the transformations needed to transform one process into the other.

We are working on generalizing the decomposition and joining graph edit operations to tackle the situation when one node in the first graph corresponds to a subgraph of the second graph. This situation appears when the first service has a single operation (activity) to achieve certain functionality, while in the second service the same behavior is achieved by receiving several messages.

The next step of this work will be to address the problem of comparing a process with a set of processes in a library. Due to the complexity of the matchmaking algorithm, some optimization techniques have to be developped (indexes, clustering to regroup similar services in the library, etc.). We will also experimentally evaluate the performance of the behavior based retrieval method in terms of precision and recall.

Acknowledgements

The researcher Juan Carlos Corrales is Alban Program Fellowship recipient (High-level scholarship program for Latin America, http://www.programalban.org).

References

1. Foster, I., Voeckler, J., Wilde, M., Zhao, Y.: Chimera: A virtual data system for representing, querying and automating data derivation. In: Proc. of 14th Conf. on Scientific and Statistical Database Management. (2002)
2. Benatallah, B., Casati, F., Toumani, F.: Web services conversation modeling: A cornerstone for e-business automation. IEEE Internet Computing (2004)
3. Benatallah, B., Casati, F., Grigori, D., Motahari Nezhad, H.R., Toumani, F.: Developing adapters for web services integration. In: Proc. of CAISE. (2005)
4. Paolucci, M., Kawamura, T., Payne, T.R., Sycara, K.: Semantic matching of web services capabilities. In: Proc. of First International Semantic Web Conference (ISWC). (2002)
5. Bernstein, A., Klein, M.: Towards high-precision service retrieval. In: Proc. of Int. Semantic Web Conference (ISWC). (2002)
6. Benatallah, B., Hacid, M., Rey, C., Toumani, F.: Semantic reasoning for web services discovery. In: Proc. of WWW Workshop on E-Services and the Semantic Web. (2003)

7. Kawamura, T., De Blasio, J., Hasegawa, T., Paolucci, M., Sycara, K.: A preliminary report of a public experiment of a semantic service matchmaker combined with a uddi business registry. In: Proc. of 1st International Conference on Service Oriented Computing (ICSOC). (2003)
8. Cardoso, J., Sheth, A.: Semantic e-workflow composition. Journal of Intelligent Information Systems **21** (2003) 191–225
9. Wu, J., Wu, Z.: Similarity-based web service matching. In: Proc. of IEEE International Conference on Services Computing. (2005)
10. Trastour, D., Bartolini, C., Gonzalez-Castillo, J.: A semantic web approach to service description for matchmaking of services. In: Proc. of Int. Semantic Web Working Symposium (SWWS). (2001)
11. S. Bansal, S., Vidal, J.M.: Matchmaking of web services based on the DAML-S service model. In: Proc. of Int. Joint Conference on Autonomous Agents and Multiagent Systems. (2003) 926–927
12. Zdravkovic, J., P. Johanesson, P.: Cooperation of processes through message level agreement. In: Proc. of Int. Conf. On Advanced Information Systems Engineering (CAISE). (2004)
13. Piccinelli, G., Di Vitantonio, G., Mokrushin, L.: Dynamic service aggregation in electronic marketplaces. Computer Networks **2**(37) (2001)
14. Wombacher, A., Mahleko, B., Fankhauser, P., Neuhold, E.: Matchmaking for business processes based on choreographies. In: Proc. of IEEE International Conference on e-Technology, e-Commerce and e-Service. (2004)
15. Benatallah, B., Casati, F., Toumani, F.: Analysis and management of web services protocols. In: Proc. of ER. (2004)
16. Bordeaux, L., et al.: When are two web services compatible? In: Proc. of TES. (2004)
17. Dong, L., Halevy, A., Madhavan, J., Nemes, E., , Zhang, J.: Similarity search for web services. In: Proc. of VLDB. (2004)
18. Wombacher, A., Mahleko, B., Fankhauser, P.: A grammar-based index for matching business processes. In: Proc. of IEEE International Conference on Web Services. (2005) 21–30
19. Shen, Z., Su, J.: Web services discovery based on behavior signatures. In: Proc. of IEEE International Conference on Services Computing. (2005)
20. Shapiro, L.G., Haralick, R.M.: Structural descriptions and inexact matching. IEEE Trans. Pattern Anal. Mach. Intell. **3** (1981)
21. Bunke, H.: Recent developments in graph matching. In: Proc. of 15th Int. Conf. on Pattern Recognition. (2000) 117 – 124
22. Messmer, B.: Graph Matching Algorithms and Applications. PhD thesis, University of Bern (1995)
23. Andrews, T., Curbera, F., Dholakia, H., Goland, Y., Klein, J., Leymann, F., Liu, K., Roller, D., Smith, D., Thatte, S., Trickovic, I., Weerawarana, S.: Business process execution language for web services, version 1.1. In: Standards proposal by BEA Systems, International Business Machines Corporation, and Microsoft Corporation. (2003)
24. Mendling, J., Ziemann, J.: Transformation of bpel processes to epcs. In: Proc. of the 4th GI Workshop on Event-Driven Process Chains (EPK2005). (2005)
25. Patil, A., Oundhakar, S., Sheth, A., Verna, K.: Meteor-s web service annotation framework. In: Proc. of WWW Conference. (2004)
26. Angell, R.C., Freund, G.E., Willett, P.: Automatic spelling correction using a trigram similarity measure. Information Processing and Management **19**(4) (1983) 255–261
27. Miller, G.: Wordnet: A lexical database for english. Communications of the ACM **38**(11) (1995) 39–41

Evaluation of Technical Measures for Workflow Similarity Based on a Pilot Study

Andreas Wombacher

University of Twente,
7500 AE Enschede, The Netherlands
A.Wombacher@utwente.nl

Abstract. Service discovery of state dependent services has to take workflow aspects into account. To increase the usability of a service discovery, the result list of services should be ordered with regard to the relevance of the services. Means of ordering a list of workflows due to their similarity with regard to a query are missing. In this paper different similarity measures are presented and evaluated based on a pilot of an empirical study. In particular the different measures are compared with the study results. It turns out that the quality of the different measures differ significantly.

1 Introduction

A service oriented architecture is based on services maintained by independent service providers and invoked by service requesters. A service invocation of a stateless service consists of a single request-response sequence. In case services are statefull further interaction may be necessary. The set of allowed interaction sequences is also known as a choreography. The challenge is to find services which guarantee a successful interaction of service requester and service provider which is also known as service discovery.

Service discovery is not only applied at the run-time of a system to realize late binding of services, but can also be applied during the design process to support the re-use of services and to reduce the maintenance costs of services. In particular, at design time service discovery can be used to determine services which can be composed using e.g. a programming in the large approach [4, 28, 22]. Further, service discovery may be used to check whether a certain service has already been implemented before starting your own implementation. In addition, it can be applied to reduce maintenance costs of a collection of services by clustering services based on their functionality. The explication of shared functionality and its re-use in services of the cluster ensures that the functionality must be maintained only once. Due to the increasing number of services in enterprises these techniques get more and more important.

Service discovery with a focus on choreographies has been addressed in different approaches like e.g. [18, 15, 7, 11, 25, 10, 8]. However, the usability of the service discovery depends on the correctness and the applicability of the derived results. Let's consider the following two scenarios:

- the service discovery provides an extensive list of hundred or more services: in this case a user of such a system would expect the result list to be ordered such that the most significant results are at the top and the least significant results are at the bottom of the list;

R. Meersman, Z. Tari et al. (Eds.): OTM 2006, LNCS 4275, pp. 255–272, 2006.

– the service discovery provides no result at all since the query is too specific: in this case a user would expect that the services being most similar to the query - up to a certain threshold - are provided in the result list ordered based on their significance.

In either case a metric is needed to express the equivalence / bisimulation of the query stated in the service discovery with the service descriptions contained in the repository. Such a metric is the similarity of services, or to be more focused the similarity of choreographies.

From a conceptual point of view choreographies represent allowed sequences of interactions, which can be represented as a workflow model. As a consequence, similarity of choreographies can be represented in a more general way as similarity of workflow models. In this paper, different workflow similarity measures are investigated, which have been proposed in previous work [32,21] and are evaluated using the preliminary results of a pilot study [33]. In particular, workflow mining measures and measures comparing collections of workflow states are applied as similarity measures. The presented evaluation focuses on some measures of workflows and therefore is incomplete. Additional approaches are discussed in the related work section. The above mentioned approaches have been selected, because mining is a big research area and the used measures seem to work there pretty well and the approach based on collections of workflow states turned out to be most promising in a previous more technical study [32]. Collections of workflow states are originally used to index workflow models / choreographies improving the performance of service discovery [14,13]. The workflow mining measures are originally used to determine the compatibility of logging data and a workflow model [21].

The contribution of this paper is to evaluate these measures by applying them on a questionnaire [20] used in a pilot study [33]. The pilot study aimed to check whether the used hypotheses and the constructed questionnaire are usable, thus, the empirical study performed on it will provide reasonable results. However, the already obtained results generated by workflow experts during the pilot study give some general indication on the importance of different aspects of a workflow similarity measure which is now used to evaluate the technical workflow similarity measures. The outcome of this comparison is that the collection of workflow states provide good results as long as there are not to many cycles contained in the used workflows. Further, it seems that the mining measures are less applicable to similarity in coordination processes.

The paper continues with a discussion of related work. Next, the pilot study and its results are summarized in Section 3. Then the different measures are introduced and their evaluation based on the pilot study results is described in Section 4, 5 and 6 respectively. Section 7 summarizes the findings and discusses future work.

2 Related Work

Similarity in general is a measurement indicating “closeness” between two entities, that is, in our case a measure indicating the equivalence or bisimulation of workflows. Similarity is a symmetric function, which is normalized (values are between 0 and 1) and fulfills the triangular inequation, that is, the sum of the similarities of workflow A and B, and B and C is bigger than the similarity of workflow A and C.

The approaches mentioned below are categorized into language and structure based approaches [1].

Language based approaches often make use of distance measures of strings as a basis to calculate the similarity measure. There exist different distance measures for strings like for example the Hamming distance used in information theory [23] or the edit distance usually applied on strings in the context of text. The edit distance (or Levenshtein distance) [12] between two strings is the smallest number of substitutions, insertions, and deletions of symbols that can be used to transform one string into another. This definition based on a single string can be extended quite easily to a set of strings, i.e. languages. However, this extension does not work in case at least one language is infinite. In [17] an approach where costs are assigned to each change operation is proposed, which can also be applied to infinite languages. Actually, the approach calculates the minimal distance of a string accepted by the first automaton with a string accepted by the second automaton. The issue with this approach is that the similarity drills down to the difference of two strings, which is quite unspecific in case the languages contain a lot of strings.

Structural based approaches are based on relating or transforming a structural representation of a workflow. For instance, a workflow can be interpreted as a directed graph and therefore graph similarity measures can be applied. An example similarity measure based on edit distance has been proposed in [3] addressing graph isomorphism, while [16] addresses subgraph isomorphism.

Another structural approach for reconciliation of processes is presented in [6] also providing a similarity measure. The approach focuses on the common alphabet of two workflows and removes the exclusively used messages of the alphabets. Further, workflow transformation rules, as specified in [26] for Workflow Nets, are used to transform both workflows to the same automaton using only the shared alphabet.

An extended class of structural similarity measures also considers the probability of the occurrence of certain interaction sequences for calculating the distance measure. This is, e.g., the case for the mining based measure introduced in Section 4 and distance definitions based on labeled Markov processes [24, 5]. Since the setting in this paper is per se without concrete interaction sequences, there is no knowledge about probabilities of interaction sequences. In this paper an equal distribution of the interaction sequences is assumed and the focus is on the mining measure in a first glance.

In either case, the language aspects are neglected and only structural aspects are considered, which is not sufficient since e.g. language equivalent workflow models with different graph representations are not considered equivalent.

3 Pilot Study

The evaluation of the technical measures introduced in Section 4, 5 and 6 aims to indicate to which extend a measure is meaningful to a human user. Therefore, the evaluation has to be based on an empirical study with the aim to get a good understanding of the human intuition of workflow similarity. Potentially, each individual will have a different intuition of what is important for the similarity of workflows. Therefore, the best way to

[1] A detailed discussion of most of the approaches mentioned below is available in [32].

conduct an empirical study is to ask multiple persons, thus, the results will be more reliable. This can be achieved in an efficient manner using a questionnaire. To determine whether the designed questionnaire will be suitable a pilot study has been conducted. For this evaluation of the similarity measures, the preliminary findings of the pilot study [33] are used as a first indication on the quality of the technical measures. While in [33] the design, conduction and analysis of the study is described, in this paper the focus is on the comparison of some technical measures with the results of the study. In the following, the pilot study is described. In particular, a brief description of the design of the questionnaire is provided, followed by a description of the data collection phase and a summary of the preliminary finding of the pilot study.

3.1 Formal Workflow Model

Finite State Automata (FSA) [9] are the simplest possible model to represent workflows offering mainly sequence, choice and iteration of tasks. More complex models provide additional expressiveness like parallel execution or recursion. However, higher expressiveness requires to investigate more scenarios for determining/evaluating a similarity measure and the aim is to keep it simple first. Thus, Finite State Automata are used as a formal model. The results derived from this study are applicable to the service discovery scenario since there exists a service discovery approach [31] which is based on Finite State Automata extended by meta data for service matchmaking [30].

A Finite State Automaton is based on a set of states represented as circles, a start state, a set of finite or accepting states represented by circles with thick lines, and labeled transitions represented as directed arcs. In particular, a labeled transition means that a state is changed when a certain message is either sent or received. Example Finite State Automata (FSA) are depicted in Figure 1. An automaton describes the potential execution sequences of a workflow which is also called the language of an automaton. The example automata can be classified in acyclic automata (like e.g. Figure 1 A and B) providing finite languages and cyclic automata (like e.g. Figure 1 C and D) representing infinite languages.

3.2 Questionnaire

It is expected that the workflow similarity is influenced by several aspects like the language of an automaton, its structure and its semantics. With language the possible execution sequences of workflow represented as an automaton is meant. Structure means the structural representation of an automaton comparable to a directed graph. The semantics of the used transition labels determines the semantics of the complete workflow.

Since there is no clear understanding on how the different aspects depend on each other a set of hypotheses has been set up. Semantics is considered implicitly in all questions of the questionnaire by using semantically meaningful workflows. In particular, RosettaNet Partner Interface Processes (or PIPs) [19] are used as transition labels. Examples of the used labels and brief descriptions of their semantics are given in Table 1. Since some of the PIPs are covering two messages which are usually request and response messages these message are labeled without and with prime respectively.

The example automaton depicted on the left hand side of Figure 1 uses the labels described in Table 1. This workflow starts with a request for a purchase order (transi-

tion labeled *p*), followed by an acceptance of the purchase order (*p'*). Then, an invoice for this specific order (*i*) is sent. The customer can now choose to pay the order (*r*), after which the order is shipped (*n*), or to send a cancellation request (*c*) followed by a cancellation confirmation (*c'*).

Table 1. Examples of used PIPs

Code	Label	Name
PIP3A4	p and p'	Request purchase order (PO)
PIP3A9	c and c'	Request PO cancellation
PIP3B2	n	Notify of advance shipment
PIP3C3	i	Notify of invoice
PIP3C6	r	Notify of remittance advice

Based on the hypotheses (see [33] for details) the questions of the questionnaire are constructed in such a way that the intended decision criteria on ordering the results gives some indication on the validity of the hypothesis. Each question contains a reference automaton and a set of either three or four solution automata (*A, B, C, D*). A respondent has to order the solution automata by similarity with respect to the reference automaton. If a respondent finds multiple solution automata equally similar to the reference automaton, she can assign several automata to the same position of the order. Respondents are also asked to state their reason on how they derived the provided order. An example question is depicted in Figure 1. The questionnaire is available at [20].

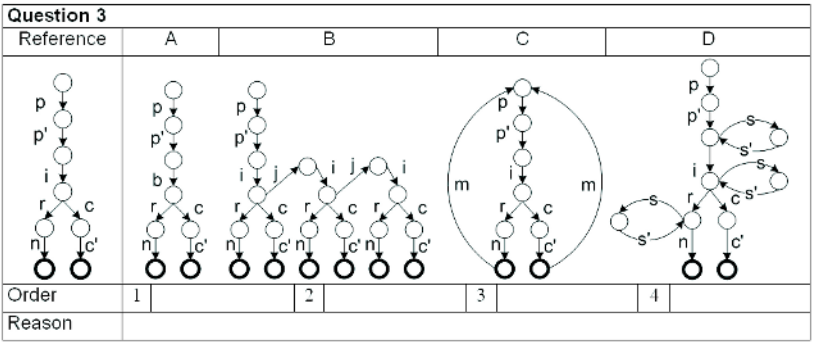


Fig. 1. Question 3

3.3 Results

The pilot study has been based on a group of 27 international technical workflow specialists from which 12 responded from seven different countries. The respondents have different backgrounds and different areas of expertise, like e.g. inter-organizational workflows, workflow matchmaking, or semantic service composition.

The questions are analyzed by having a look at the number of supporters of a hypothesis and the maximum number of equal answers, as well as the number of opposing respondents. Due to the small number of respondents in the pilot study, only those questions with a strong support can be considered for the evaluation of the similarity measures. There are hypotheses with a strong support stating that the language is more important than the structure on different levels of granularity (the corresponding questions are Q1, Q3, Q5, Q6, Q13, and Q17). The question Q14 indicates that super-automata are considered more similar than automata with extra transitions before or within the paths of the reference automaton. The questions Q16 and Q23 have quite some supporters and only a few opposing respondents. The underlying hypothesis states that an automaton having a transition as a loop is more similar than a comparable automaton not having the transition at all.

In case of the remaining hypotheses and questions the number of supporters and opponents is quite high and therefore they will not be considered for the evaluation of the measures introduced in this paper. Figure 2 represents a chart where the number of supporting respondents is given per question used in the evaluation.

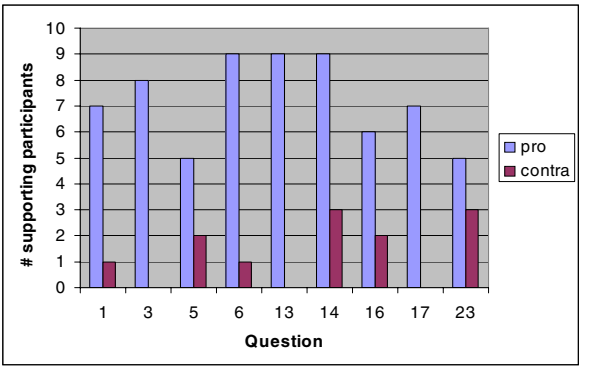


Fig. 2. Relevant questions from the pilot study

These questions will serve as an evaluation criteria for the different similarity measures introduced in the next sections, that is, a measure used in process mining (Section 4), a workflow state set (Section 5) and a workflow state list (Section 6) measure. This evaluation is preliminary due to the limited number of participants and focus on aspects relevant to a similarity measure for service discovery, where e.g. additional information on occurrences of execution sequences are usually not available. The different measures can be evaluated in other scenarios with different results.

4 Mining Conformance Based Similarity Measures

4.1 Overview Approach

Workflow mining is quite a prominent field in the workflow community. It investigates means to construct workflow models from log information gathered from application

software. The aim of mining is to determine a workflow model representing the occurrences of tasks/events in an optimal way. The work referred to is based on Workflow Nets (WF-Nets) [27]. Where a distribution of tokens (called marking) is used to represent a state and markings are changed by executing / firing a transition. Thus, the conformance testing is performed based on WF-Nets and log information. Since log information is a set of typical execution sequences of an application, the different sequences occur a different number of times. In [1] an approach has been proposed to construct log data based on a given Colored Petri Net, which can afterward be used in the ProM framework [29] to calculate the conformance of the log with a Workflow Net. In [21] three different measures have been proposed and are calculated by the ProM framework: fitness, structural and behavioral appropriateness.

Fitness is based on replaying the log based on a given workflow model and measuring the mismatch. The mismatch is indicated by two ratios: the first ratio gives an indication of the missing parts in the workflow model described by the ratio of tokens that have to be introduced in addition to generate a valid replay and the number of tokens that have been consumed during the replay. The second ratio gives an indication of the proper completion of the workflow model described by the ratio of tokens that remain in the workflow after completion of the log and the tokens that have been produced.

This definition of mismatch differs from the language based edit distance. For example, in case a workflow model contains the transition sequence ABCDE and the log contains the sequence CDE then only one additional token is needed for the replay to enable transition C without having executed transition A and B. In case of the edit distance the measure is based on the length of the missing sequence, thus two.

Structural appropriateness addresses the minimality of the workflow model structure representing a certain log. In particular, the measure is high if the number of equally labeled transitions is small. Therefore structural appropriateness is defined based on the ratio of the used transition labels and the number of transitions in the workflow model.

Behavioral appropriateness addresses the minimality of the workflow model behavior representing a certain log. The aim is to avoid that the workflow model represents much more execution sequences than sequences are contained in the log. Thus, the behavioral appropriateness is defined as the ratio of the mean number of enabled transitions and the number of transition labels.

These three independent measures have to be combined to a single measure representing the conformance of log data and a workflow model. Since not all possible combinations can be checked in this investigation, the best possible individual result out of all three measures per test case is considered at a first glance.

4.2 Evaluation

original automata. The evaluation of the mining conformance measures is conducted such that the reference automaton of each question is used to construct the corresponding log file. The log generation uses an equal distribution of all possible branches at a particular state of the workflow execution. Afterward the log data and the Workflow Nets representing the corresponding solution automaton are loaded into the ProM framework. Next, the conformance test is initiated and the values are manually

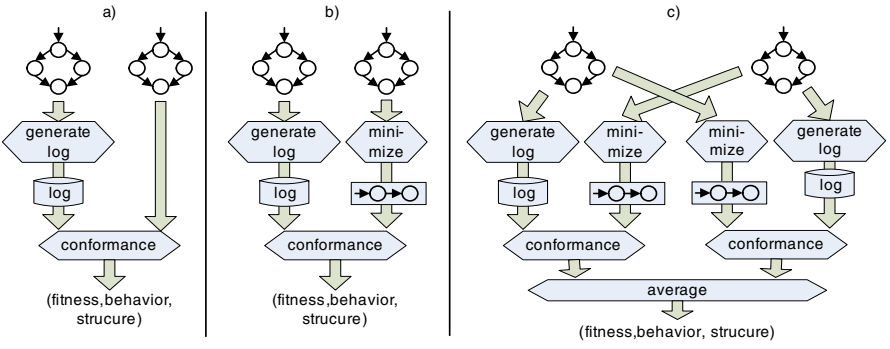


Fig. 3. Mining Conformance based similarity measure: a) original automata b) minimized automata c) symmetric approach

collected. The approach is sketched in Figure 3a). Based on these values the order of the solution automata is derived for each conformance measure respectively. The calculated order is then compared with the results of the pilot study and the conformance with the supporting answers of the pilot study are represented in graphs. The presented approach is further modified to improve the conformance with the pilot study.

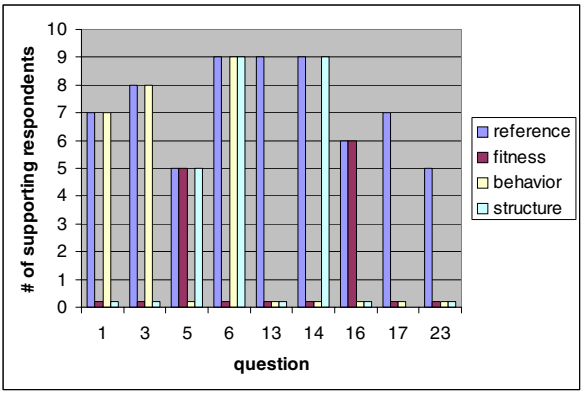


Fig. 4. Pilot results, fitness, and structural and behavioral appropriateness

Figure 4 presents the results. It can be seen that the behavior measure supports questions 1, 3 and 6, while the fitness measure supports questions 5 and 16. The structure measure indicates support for questions 5 and 14. However, there is no support at all for questions 13, 17 and 23, although question 13 has a strong support in the pilot study. To support the reader doing the comparison the reference values taken from the pilot study (see Figure 2) are also contained in Figure 4. Due to the readability the results contradicting the hypothesis underlying the corresponding question are not depicted in Figure 4. In particular, it turned out that in all questions supported by one measure there is also at least one measure contradicting the underlying hypothesis.

minimized automata. As stated in Section 4.1 all conformance testing measures are producing better results the smaller the used workflow model is. In particular, the fitness measure can only return a conformance value of one if there is only a single final state. This minimality requirement has not been considered in the original representation of the solution automata. Therefore, the evaluation process has been repeated with minimized solution automata. That is, the solution automaton is minimized first before its Workflow Net representation is loaded into the ProM framework to calculate the conformance measures. Afterward, the collection of the conformance measures and the analysis are done according to the previous evaluation. The approach is sketched in Figure 3b). The results are depicted in Figure 5 again containing the results of the pilot study (see Figure 2). The log data generated from the reference automaton is not affected by this change since the log data depends on the accepted language rather than the automaton structure.

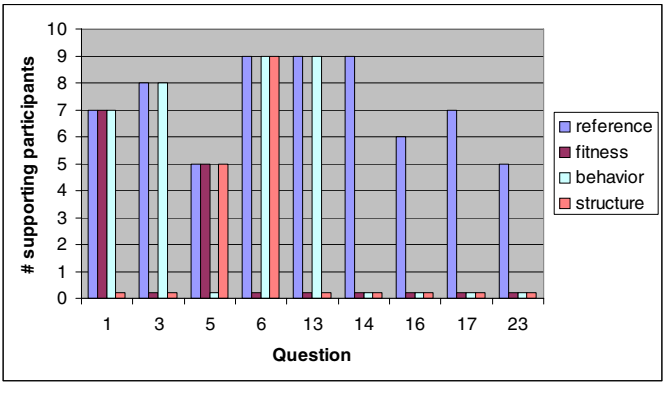


Fig. 5. Pilot results, fitness, and structural and behavioral appropriateness of minimized automata

It turns out that there is now full support of question 13 by the behavior measure although questions 17 and 23 still do not have any support. In particular, it can be seen that the fitness measure is now supporting question 1 but on the other hand side it is no longer supporting question 16. Thus, the consistency of the results derived from the different measures has slightly improved since now the more important question 13 is supported.

symmetric conformance measures based on minimized automata. The investigation of the previous results turns out that the derived measures are not symmetric, although similarity is supposed to be symmetric (see Section 2). This is because transition labels contained in the solution automaton but not contained in the log data derived from the reference automaton are simply considered to be irrelevant introducing asymmetry. Since the asymmetric version has limited success, the asymmetric definition is now applied in both directions and the average value of both intermediate results is calculated. In particular, the average of the conformance values generated based on the log of a reference automaton and those generated based on the log of a solution

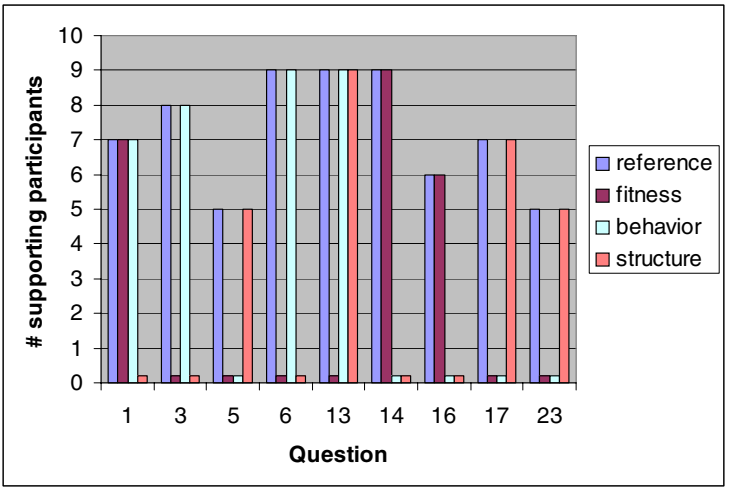


Fig. 6. Pilot results, fitness, structural and behavioral conformance based on the average and minimized automata

automaton is calculated. In this approach minimized solution and reference automata are used respectively. The approach is sketched in Figure 3c).

Figure 6 presents the results and as a reference the values of the pilot study (see Figure 2). The resulting measure is indeed symmetric. Further, it can be seen that for each question there exist at least one measure which supports the results of the pilot study. However, using the three measures to construct a single measure returning the best result for each question turns out to be difficult. This is because the three different measures result in different orders of the solution automata and it can not be derived which order is the most promising one. Finding a linear combination of the different measures to get as many supported questions as possible turned out to be difficult. In particular, all linear combinations have been tested where the factors weighting the different measures have been varied in the interval of -1 to 1 based on the step size of 0.001 using a precision of the real values of 0.0001. It turned out that in the best case a combination with a maximum of five supported questions out of the nine contained in the evaluation could be found. Thus, based on the intuitive approach of combining the measures, the applicability of the mining measures seems to be limited although there exists a supporting measure for each question.

5 Workflow State Set Based Similarity Measure

5.1 Approach

The Levenshtein or edit distance [12] specifies the smallest number of substitutions, insertions, and deletions of symbols to transform one string into another. For example, the edit distance of the strings $abab$ and aab is one by removing the first occurrence of b. Based on this distance value d the similarity value sim can be calculated by subtracting

the distance value d from the maximum difference m and dividing the difference by the maximum difference m, that is, $sim := \frac{m-d}{m}$. With regard to the previous example, the similarity is $\frac{4-1}{4} = \frac{3}{4}$.

The distance of automata can be calculated based on their language representations as long as the languages and words are finite. Since infinite strings are constructed from a finite automaton, in [13] it has been proposed to represent an automaton based on its finite set of states. Since state names in an automaton specification are meaningless, the labels of all transitions leading into the state are used as a representation of the state itself. However, the start state does not necessarily has a transition leading into the state, thus, the $ character is introduced as a special symbol. Further to represent that a state is a final state, the # character is introduced as a further special symbol.

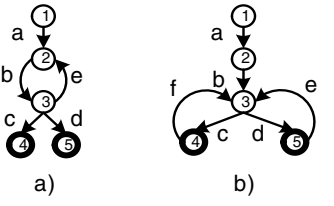


Fig. 7. Example Automata

In Figure 7a) an automaton is depicted, where the start state 1 will be represented by $, the final states 4 and 5 each by a # and the states 3, 4, and 5 by b, c, and d respectively. State 2 is represented by the characters a and e since there are two transitions resulting into state 2. As can be observed, a state can be represented by more than one character as well as a single character can represent more than one state. Using this automaton representation it is quite obvious that there exist a lot of different automata resulting in the same representation, thus, the ambiguity of the representation is quite high. To reduce this ambiguity context information has to be added comparable to resolving ambiguity in grammars (lock-ahead in context sensitive grammars). Therefore, a state is not represented by all single transitions leading to this state, but by the set of transition sequences of length n leading into this state.

Applying this approach using a sequence length of 2 to the example in Figure 7a) results in: The start state 1 is represented by $$, and the final states 4 and 5 are represented by $c\#$ and $d\#$ respectively. State 4 and 5 are described by bc and bd. In case of state 3 there are now two sequences of length two reaching state 3 which are ab and eb. State 2 is represented as $\$a$ and be. In [13] an algorithm is described to generate the state representation for a given length n. In the following a sequence resulting in and representing a state is called an n-gram, which is known from substring matching in full-text indexes [2]. A combination of n-grams can be used to construct all possible execution sequences of a single automaton and thus has a strong relation to the language accepted by the automaton.

With regard to the example in Figure 7a) the 2-gram set representation is $\{\$a, be, ab, eb, bc, db, \$\$, c\#, d\#\}$ which is the set of all 2-grams mentioned above.

5.2 Evaluation

The evaluation of n-gram sets is based on the similarity derived from the edit distance between the n-gram sets of the reference and the solution automaton respectively. The edit distance of two n-gram sets is calculated by summing up the minimum edit distance of each n-gram within the first set and an n-gram in the second set, added to the sum of the minimum edit distance of each n-gram within the second set and an n-gram in the first set. This can be formally described for two n-gram sets $A := \{a_1, \ldots, a_l\}$ and $B := \{b_1, \ldots, b_k\}$ as

$$d(A,B) := \sum_{i=1}^{l} \left(\min_{j=1..k} d(a_i, b_j) \right) + \sum_{j=1}^{k} \left(\min_{i=1..l} d(a_i, b_j) \right)$$

where $d(a_i, b_j)$ is the edit distance of the two n-grams a_i and b_j which are considered as strings. Be aware that an n-gram is a sequence of transition labels, where each label is treated as a unique token, that is, a character in terms of a string. The maximum distance between the sets of n-grams is twice the product of the maximum number of n-grams contained in one of the sets and the length n of the n-gram, that is, $m = 2 * Max(|A|, |B|) * n$, where $|A|$ specifies the size of the set of n-grams A. Let's consider the following example based on the two 1-gram sets $A = \{\$, a, b, \#\}$ and $L(B) = \{\$, a, \#\}$, then the maximum distance m is $m = 2 * 3 * 1 = 6$.

This distance definition is applied to each solution automata and the reference automaton. Further, the maximum distance is calculated per reference and solution automaton pair since it is needed to calculate the similarity measure. The derived values are collected and the corresponding order of the solution automata is derived. An overview of the approach is depicted in Figure 9a). The resulting order is compared with the results from the pilot study and the results are depicted in Figure 8. In particular, values from one to five for n have been used. It turned out that the generated orders never produced contradicting results.

The results are promising. It turns out that question 3 has no support independent of the amount of context information taken into account. The reason for this is that the n-gram set approach generates quite high distance values in case an automaton contains cycles. In particular, question 3 (see Figure 1) has two solution automata containing a lot of cycles, which result in a wrong order of the solution automata. However, the other questions are supported for at least one specific n. It can be observed that e.g. for question 1 the context information for n equals one is not sufficient to generate the correct order, while the context information bigger than one is sufficient. Further, it can be observed at question 14 that the correct order is determined as long as not too much context information is considered. In particular, in case of n equals five the correct order can no longer be derived. Again this is due to the non-proportional increase of distance values due to cycles in solution automata. A curiosity can be observed at question 23. Here the correct order is determined for all n-gram sets except for n equals two. The explanation for this is that the differences between the similarity values of the corresponding solution automata are quite small. In particular, for n equals two the similarity values for two solution automata get equal under the considered precision which results in the wrong order of solution automata. For this particular data set, the

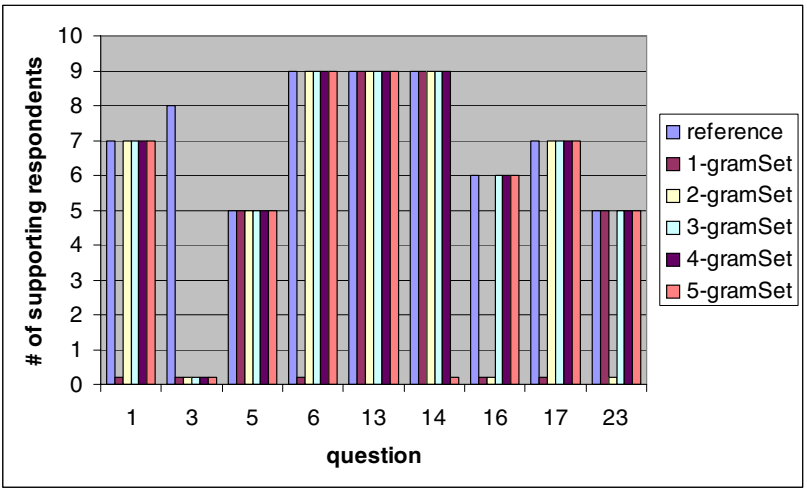


Fig. 8. Pilot results and n-gram set results

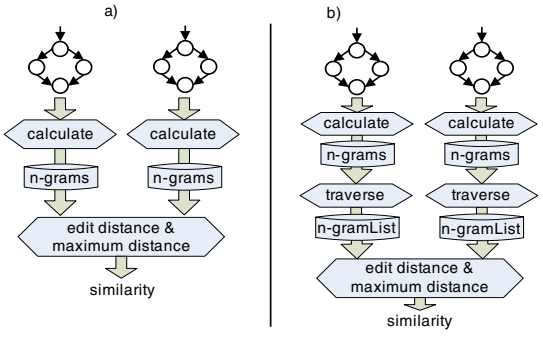


Fig. 9. State Collection Schema based Similarity measure: a) set based b) list based approach

best result is achieved for n equals four and the worst result for n equals one. However, the usage of n equals one or two in general is really unlikely due to its high ambiguity. The results for n equals three and five provide eight and seven supporting results respectively. In future work we will investigate how to determine a good estimate for the n and how to decrease the dominance of cycles on the similarity value.

6 Workflow State List Based Similarity Measure

6.1 Approach

N-gram sets represent an abstraction of the set of states of an automaton. This representation has high ambiguity which can only be resolved in a limited way by increasing the size of n. Alternatively, a set of n-gram lists can be introduced abstracting the language

of an automaton, where each execution sequence is represented by a finite n-gram list representing the state of the automaton passed during the execution of a sequence. To represent the potentially infinite sequence by a finite n-gram list repeated occurrences of n-grams (states) are removed. An automaton is then represented as a set of n-gram lists, which has less ambiguity compared to n-gram sets, but also has a much higher complexity in creating the automaton representation.

To illustrate the approach, the example in Figure 7a) is re-used. Please have a look at the generation of the 2-grams in Section 5.1. The 2-gram list representation is based on a list denoted as $< \ldots >$ containing the comma separated list of unique 2-grams. In particular, the string abc accepted by the example automaton in Figure 7a) correlates to the list of states $< 1, 2, 3, 4 >$ which is represented in a 2-gram list as $< \$\$, \$a, ab, bc, c\# >$. The set of 2-gram lists representation consists of the following four lists: $\{< \$\$, \$a, ab, bc, c\# >, < \$\$, \$a, ab, bd, d\# >, < \$\$, \$a, ab, be, eb, bc, c\#>, < \$\$, \$a, ab, be, eb, bd, d\# >\}$ representing the sequences $\{abc, abd, abebc, abebd\}$. No additional lists can be constructed because all longer sequences like e.g. $abebebc$ result in the same 2-gram list representation, that is, $< \$\$, \$a, ab, be, eb, bc, c\# >$. This is because the repeated occurrence of the 2-grams be and eb are not repeated in the list. Thus, the infinite language of a cyclic automaton can be represented as a finite set of finite n-gram lists. The calculation of n-gram lists in automata with complex cycles, that is, automata having at least two cycles sharing a single state, results in a high number of n-gram lists and has a high computational complexity. The example automaton in Figure 7b) requires to maintain more than 2500 2-gram lists.

6.2 Evaluation

The evaluation of the n-gram list approach goes along the lines of the n-gram sets except that the automaton has to be traversed sufficiently to derive the n-gram lists and that a modified distance notation has to be applied. An overview of the resulting approach is sketched in Figure 9b). Be aware that an automaton is represented as a set of n-gram lists where each n-gram list is considered to be a single string. Again, transition labels used in an n-gram are considered as a token representing a single character in a string. Thus, the edit distance of two n-gram lists $\bar{a} :=< a_1, \ldots, a_l >$ and $\bar{b} :=< b_1, \ldots, b_k >$ can be formally defined as $d(\bar{a}, \bar{b})$ where $\bar{a}$ and $\bar{b}$ are considered as strings. The edit distance of two sets of n-gram lists $\bar{A} := \{\bar{a}_1, \ldots, \bar{a}_l\}$ and $\bar{B} := \{\bar{b}_1, \ldots, \bar{b}_k\}$ is defined as

$$d(\bar{A}, \bar{B}) := \sum_{i=1}^{l} \left(\min_{j=1..k} d(\bar{a}_i, \bar{b}_j) \right) + \sum_{j=1}^{k} \left(\min_{i=1..l} d(\bar{a}_i, \bar{b}_j) \right)$$

where $d(\bar{a}_i, \bar{b}_j)$ is the edit distance of two n-gram lists as defined above.

The maximum distance between the sets of n-gram lists is twice the product of the maximum number of n-gram lists contained in one of the sets and the length of the longest n-gram list contained in one of the sets, that is,

$$m = 2 * Max(|\bar{A}|, |\bar{B}|) * Max_{\bar{s} \in \bar{A} \cup \bar{B}}(|\bar{s}|)$$

where $|\bar{A}|$ specifies the length of the set of n-gram lists $\bar{A}$ and $|\bar{s}|$ the length of an n-gram list $\bar{s}$. Let's consider the following example based on the two sets of 1-gram lists

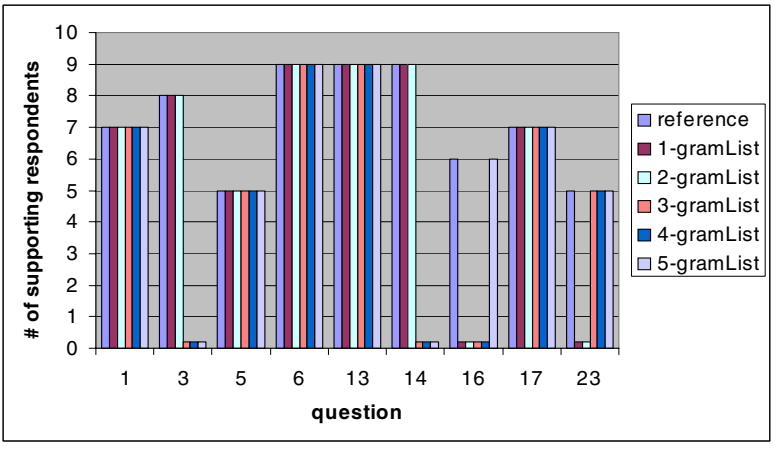


Fig. 10. Pilot and n-gram list results

$\bar{A} = \{< \$, a, \# >, < \$, b, \# >\}$ and $\bar{B} = \{< \$, a, b, c, \# >\}$ then the maximum distance m is $m = 2 * 2 * 5 = 20$.

The edit distance and the maximum values are calculated for each pair of reference and solution automata. Based on these values the solution automata are ordered and the order is compared with the results from the pilot study. The results are depicted in Figure 10 and represent the pilot results and the results for sets of n-gram lists varying n from one to five. Be aware that the variation of n up to five is only possible because non of the automata in the relevant questions does contain a complex cycle with a length of the cycles smaller than five [2].

The results are comparable to the ones of n-gram sets. The overall results are fine. Questions 3 and 14 are only supported when n is smaller than four, while questions like 16 and 23 are only supported starting from n greater than two. It can be observed that the 1-gram list supports already the questions 1,3,5,6,13,14, and 17. It is interesting to see that this approach supports in particular question 3 which could not be supported by any of the n-gram sets. However, questions 16 and 23 are not supported. As a conclusion the 1-gram list delivers good results with reasonable complexity, although criteria for selecting an appropriate n are subject to future work.

7 Conclusion and Future Work

Service discovery of statefull services requires to search for services based on their choreography, that is, their workflow, and present the most significant results in an ordered list to the human user. A measure of significance is the similarity of a workflow describing a service and the query represented by a workflow again. Therefore different similarity measures have been summarized and have been evaluated with regard to a pilot study on the human understanding of workflow similarity.

[2] The solution automaton C in Question 3 (see Figure 1) contains complex cycles, but the length of a single cycle is six. Thus, the combinational explosion does not apply yet.

As a conclusion of this specific service discovery evaluation it turns out that the very simple approach based on n-gram sets delivers the best results with regard to the support of the pilot study results. However, the approach can be improved by lowering the impact of cycles on the similarity measure. This idea is compliant with the weighting of execution sequences in the mining measures and instantiates in the support or non support of question 3 respectively. The mining measures are based on generated execution sequence log data, where the generation of log data assumes that each choice at a certain state is equally likely. The n-gram list approach turned out to be comparable with the n-gram set approach but with a much higher complexity. As a consequence, the n-gram set approach is preferable based on this preliminary evaluation.

Future work will be to conduct the empirical study and to further improve the n-gram set approach. In particular, investigate the criteria for selecting a sufficient n and to reduce the effects of cycles on the measure. Further, additional similarity measures have to be implemented and evaluated based an the conducted study. Finally, the empirical study has to be conducted with a bigger number of participants.

References

1. A. Alves De Medeiros and C. Gnther. Process mining: Using CPN tools to create test logs for mining algorithms. In *Proceedings of the Sixth Workshop on the Practical Use of Coloured Petri Nets and CPN Tools (CPN)*, 2005.
2. R. A. Baeza-Yates. Text retrieval: theory and practice. In J. van Leeuwen, editor, *Proceedings of the 12th IFIP World Computer Congress*, pages 465–476, Madrid, Spain, 1992. North-Holland.
3. G. Chartrand, G. Kubicki, and M. Schultz. Graph similarity and distance in graphs. *Aequationes Mathematicae*, 55:129–145, 1998.
4. F. DeRemer and H. H. Kron. Programming-in-the-large versus programming-in-the-small. *IEEE Transactions on Software Engineering*, 2:80–86, 1976.
5. J. Desharnais, V. Gupta, R. Jagadeesan, and P. Panangaden. Metrics for labelled markov processes. *Theoretical Computer Science*, 318, 2004.
6. Z. Du, J. Huai, Y. Liu, C. Hu, and L. Lei. IPR: Automated interaction process reconciliation. In *Proceedings of IEEE/ACM International Conference on Web Intelligence (WI)*, 2005. accepted for publication.
7. E. Folmer and D. Krukkert. openXchange as ebXML implementation and validation; the first results. In *Proceeding of XML Europe 2003 Conference & Exposition*, May 2003.
8. X. Fu, T. Bultan, and J. Su. Realizability of conversation protocols with message contents. In *Proceedings IEEE International Conference on Web Services (ICWS)*, pages 96–103. IEEE Computer Society, 2004.
9. J. E. Hopcroft, R. Motwani, and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison Wesley, 2001.
10. E. Kindler, A. Martens, and W. Reisig. Inter-operability of workflow applications: Local criteria for global soundness. In *Business Process Management, Models, Techniques, and Empirical Studies*, pages 235–253. Springer-Verlag, 2000.
11. D. Krukkert. Matchmaking of ebXML business processes. Technical Report IST-28584-OX_D2.3_v.2.0, openXchange Project, Oct 2003.
12. L. I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals,. *Soviet Physics–Doklady*, 10(8):707–710, 1966.

13. B. Mahleko, A. Wombacher, and P. Fankhauser. A grammar-based index for matching business processes. In *Proceedings of IEEE International Conference on Web Services (ICWS)*, pages 21–30. IEEE Computer Society, 2005.
14. B. Mahleko, A. Wombacher, and P. Fankhauser. Process-annotated service discovery facilitated by an n-gram based index. In *Proceedings of IEEE International Confernce on e-Technology, e-Commerce and e-Service (EEE)*, pages 2–8, 2005.
15. M. Mecella, B. Pernici, and P. Craca. Compatibility of e-services in a cooperative multiplatform environment. In F. Casati, D. Georgakopoulos, and M. Shan, editors, *Proceedings of 2rd International Workshop on Technologies for E-Services (TES)*, pages 44–57. Springer LNCS 2193, 2001.
16. B. T. Messmer and H. Bunke. A new algorithm for error-tolerant subgraph isomorphism detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(5):493–504, 1998.
17. M. Mohri. Edit-distance of weighted automata: General definitions and algorithms. *International Journal of Foundations of Computer Science*, 14(6):957–982, 2003.
18. C. Molina-Jimenez, S. Shrivastava, E. Solaiman, and J. Warne. Contract representation for run-time monitoring and enforcement. In *Proceedings of Conference on Electronic Commerce (CEC)*, pages 103–110. IEEE, 2003.
19. RosettaNet. RosettaNet home page. http://www.rosettanet.org, 2004.
20. M. Rozie and A. Wombacher. Questionnaire of the empirical workflow similarity study. http://www.cs.utwente.nl/∼wombachera/papers/questionnaire_v1.0.zip, 2005.
21. A. Rozinat and W. van der Aalst. Conformance testing: Measuring the fit and appropriateness of event logs and process models. In *Business Process Management Workshops*, pages 163–176, 2005.
22. M. P. Singh, A. K. Chopra, N. Desai, and A. U. Mallya. Protocols for processes: programming in the large for open systems. *SIGPLAN Notices*, 39(12):73–83, 2004.
23. H. Tzschach and G. Hasslinger. *Codes fuer den stoerungssicheren Datentransfer*. Oldenburg Verlag, 1993.
24. F. van Breugel. A behavioural pseudometric for metric labelled transition systems. In M. Abadi and L. de Alfaro, editors, *Proceedings 16th International Conference on Concurrency Theory (CONCUR)*, volume 3653 of *Lecture Notes in Computer Science*, pages 141–155. Springer, 2005.
25. W. van der Aalst. Interorganizational workflows: An approach based on message sequence charts and petri nets. *Systems Analysis - Modelling - Simulation*, 34(3):335–367, 1999.
26. W. van der Aalst and T. Basten. Inheritance of workflows: an approach to tackling problems related to change. *Theor. Comput. Sci.*, 270(1-2):125–203, uary.
27. W. van der Aalst and K. Hee. *Workflow Management - Models, Methods, and Systems*. MIT Press, 2002.
28. W. van der Aalst, A. H. M. ter Hofstede, and M. Weske. Business process management: A survey. In W. van der Aalst, A. H. M. ter Hofstede, and M. Weske, editors, *Proceedings International Conference Business Process Management (BPM)*, volume 2678 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2003.
29. B. F. van Dongen, A. K. A. de Medeiros, H. M. W. Verbeek, A. J. M. M. Weijters, and W. van der Aalst. The proM framework: A new era in process mining tool support. In G. Ciardo and P. Darondeau, editors, *Proceedings 26th International Conference on Applications and Theory of Petri Nets (ICATPN)*, volume 3536 of *Lecture Notes in Computer Science*, pages 444–454. Springer, 2005.
30. A. Wombacher, P. Fankhauser, B. Mahleko, and E. Neuhold. Matchmaking for business processes based on choreographies. In *Proceedings of International Conference on e-Technology, e-Commerce and e-Service (EEE-04)*. IEEE Computer Society, 2004.

31. A. Wombacher, B. Mahleko, and E. Neuhold. IPSI-PF:a business process matchmaking engine based on annotated finite state automata. *Journal on Information Systems and E-Business Management*, 3(2):127–150, 2005.
32. A. Wombacher and M. Rozie. Evaluation of workflow similarity measures in service discovery. In *Tagungsband der Multikonferenz Wirtschaftsinformatik - Service Oriented E-Commerce Track (MKWI)*, volume P-80 of *Lecture Notes in Informatics (LNI)*, pages 57–72. Gesellschaft fuer Informatik, 2006.
33. A. Wombacher and M. Rozie. Piloting an empirical study on meassures for workflow similarity. In *accepted at IEEE International Conference on Services Computing (SCC)*, 2006.

Evolution of Process Choreographies in DYCHOR

Stefanie Rinderle[1], Andreas Wombacher[2], and Manfred Reichert[2]

[1] Dept. DBIS, University of Ulm, Germany
stefanie.rinderle@uni-ulm.de
[2] Informaton Systems Group, University of Twente, The Netherlands
{a.wombacher, m.u.reichert}@ewi.utwente.nl

Abstract. Process-aware information systems have to be frequently adapted due to business process changes. One important challenge not adequately addressed so far concerns the evolution of process choreographies, i.e., the change of interactions between partner processes in a cross-organizational setting. If respective modifications are applied in an uncontrolled manner, inconsistencies or errors might occur in the sequel. In particular, modifications of private processes performed by a single party may affect the implementation of the private processes of partners as well. In this paper we present the DYCHOR (DYnamic CHOReographies) framework which allows process engineers to detect how changes of private processes may affect related public views and - if so - how they can be propagated to the public and private processes of partners. In particular, DYCHOR exploits the semantics of the applied changes in order to automatically determine the adaptations necessary for the partner processes. Altogether our framework provides an important contribution towards the realization of adaptive, cross-organizational processes.

1 Introduction

The economic success of an enterprise more and more depends on its ability to flexibly and quickly react on changes at the market, the development, or the manufacturing side. For this reason companies are developing a growing interest in improving the efficiency and quality of their internal business processes and in optimizing their interactions with business partners and customers. Recently, we have seen an increasing adoption of business process automation technologies by enterprises as well as emerging standards for business process orchestration and choreography in order to meet these goals. Respective technologies enable the definition, execution, and monitoring of the operational processes of an enterprise. In connection with Web service technology, in addition, the benefits of business process automation and optimization from within a single enterprise can be transferred to cross-organizational business processes (*process choreographies*) as well. The next step within this evolution will be the emergence of the agile enterprise being able to rapidly set up new processes and to quickly adapt existing ones to changes in its environment.

R. Meersman, Z. Tari et al. (Eds.): OTM 2006, LNCS 4275, pp. 273–290, 2006.

One important challenge not adequately addressed so far concerns the evolution of process choreographies, i.e., the controlled change of the interactions between partner processes in a cross-organizational setting. If one party changes its process in an uncontrolled manner, inconsistencies or errors regarding these interactions might occur. Generally, the partners involved in a process choreography exchange messages via their public processes, which can be considered as special views on their private processes (i.e., the process orchestrations). If one of these partners has to change the implementation of his private process (e.g., to adapt it to new laws or optimized processes) the challenging question arises whether this change affects the interactions with partner processes and their implementation as well. Obviously, as long as a modified business process is not part of a process choreography, change effects can be kept local. The same applies if changes of a private process have no impact on related public views.

In general, however, we cannot always assume this. The modification of a private process may not only influence corresponding public processes, but also the public and private processes of its partners. For this reason, it is indispensable for any IT infrastructure to provide adequate methods for (automatically) *propagating* changes of a private process to the partner processes (if required). This important issue has not been considered by current approaches so far. As a consequence adaptations of process choreographies have turned out to be both costly and error-prone. Note that the handling of respective changes is not trivial since we must be able to precisely state which effects on partner processes result after adaptating a (private) process. In any case we need precise and formal statements about this in order to avoid implementation holes later on.

In this paper we deal with these challenges and present our DYCHOR approach which allows for the controlled evolution of process choreographies. We show how changes of a private process may affect related public views and - if so - how they can be propagated to the public/private processes of partners as well. To be able to precisely state whether change propagations to partner processes become necessary we introduce a formal model based on annotated Finite State Automata. We further exploit the semantics of the applied change operations in order to derive necessary adaptations automatically. Due to the autonomy of partners, however, private partner processes cannot be adapted automatically to changes of a process choreography. DYCHOR allows for the comprehensive assistance of users in accomplishing this task in a correct and effective manner. In this paper we restrict our considerations to structural changes (e.g., the insertion or deletion of process activities). Other adaptations of process models (e.g., the change of transition conditions) require a similar approach, but are outside the scope of this paper. We do also not address dynamic changes (i.e., the migration of running choreographies to respective changes at the type level) in this paper. Dynamic adaptations of choreographies and process instances, however, constitute an important part of our change framework [1,2].

Sect. 2 introduces an application scenario which we use throughout the paper in order to illustrate basic concepts of our framework. In Sect. 3 we discuss basic issues related to process choreographies and interactions between partner

processes. In particular, we introduce our formal model and show how it can be used to automatically generate public processes out of private ones. This provides the basis for dealing with process changes. Sect. 4 presents a classification of changes and Sect. 5 provides methods for propagating changes on behalf of selected scenarios. Sect. 6 sketches implementation issues and Sect. 7 discusses related work. We close with a summary and an outlook in Sect. 8.

2 Practical Scenario

Further discussions are based on a simple procurement process within a virtual enterprise (cf. Fig. 1). It comprises a buyer, an accounting department, and a logistics department. The accounting department approves an order (*order* message) sent by a buyer and forwards it to the logistics department (*deliver* message) to deliver requested goods. The logistics department confirms the receipt (*deliver_conf* message) to the accounting department, which forwards this message (extended by the expected delivery date and the parcel tracking number) to the buyer (*delivery* message). The buyer can do parcel tracking (*get_status* and *status* messages) of the shipped goods. Corresponding messages are forwarded by the accounting department to the logistics department.

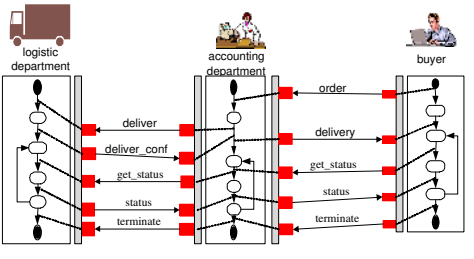


Fig. 1. Example Overview

This scenario represents a *process choreography*, i.e., a conversation between partner processes. More precisely, the participating partners exchange messages via their *public* processes, which constitute special views on *private processes* [3]. We describe the private process of the accounting department in more detail denoting it according to the BPEL specification [4]. To keep the example simple, we abstract from the structure of the exchanged messages and use simplified message names. Concrete message structures could be, for example, taken from the RosettaNet Partner Interface Processes (PIPs) 3A4 (Request Purchase Order), 3A7 (Notify of Purchase Order Update), and 3B2 (Notify of Advanced Shipment) [5].

Regarding Web services, for example, messages are exchanged by invoking operations at the respective partner sites. A Web service may comprise one or more operations (grouped within porttypes) which can be specified using WSDL.

Each operation then represents a potential message exchange between partners. If an operation contains only one single input message, it is considered to be asynchronous, otherwise the operation is synchronous. Regarding our example all operations are asynchronous except the synchronous *getStatusOP* operation provided by the *logistics* service.

We base the description of private processes on such porttype definitions (i.e., Web service specifications) by directly referring to them. In the following, private processes are denoted in BPEL [4] and are therefore specified in terms of tasks (named *activities* in the BPEL terminology) representing basic pieces of work to be performed by potentially nested services. The control flow of a BPEL process constrains possible execution orders of its activities and is based on constructs for selective (*switch* and *pick* activities), sequential (*sequence* activity), and parallel (*flow* activity) execution. In addition, a BPEL process defines the data flow between activities (variable handling and *assign* activity for mapping data between messages) regardless of their concrete implementation. Based on this understanding, the process model of one partner includes activities realizing its interaction with the other partners. These interactions are represented by exchanging messages (*receive*, *reply*, *invoke*, and *pick* activities in BPEL).

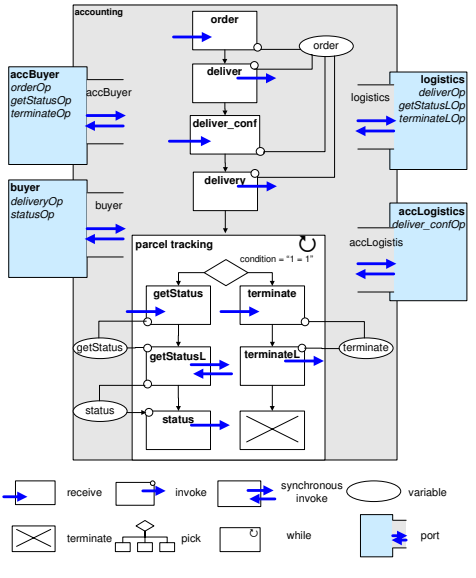


Fig. 2. Accounting BPEL Private Process

The BPEL specification of the accounting department private process is depicted in Fig. 2. The partnerLink definition associates a partner name to a bilateral interaction between two roles. The association of roles to concrete parties and operations is done in the partnerLinkType definition contained in the related WSDL file. The process starts by receiving an *order* message sent by the buyer, which is forwarded to the logistics department via a *deliver* message.

The logistics department answers asynchronously with a *deliver_conf* message. The accounting department process receives this message and forwards it to the buyer via a *delivery* message. Since the buyer is allowed to do parcel tracking arbitrarily often, this step is embedded in a non-terminating loop within the accounting process. More precisely, the accounting department may receive a *get_status* message sent by the buyer, which is followed by a synchronous invocation of the logistics *get_statusL* operation (representing two messages) and the reporting of the respective status back to the buyer (via a *status* message). Alternatively, it must be possible to terminate accounting as well as logistics process at some point in time. For this, a *termination* message can be initiated by the buyer; this message is then send to the accounting department process, which forwards it to the logistics process. After this both processes are terminated.

As a second example consider the private process of the buyer, which is depicted in Fig. 3. – We omit further details and focus on the bilateral interaction between the accounting and buyer process in the following.

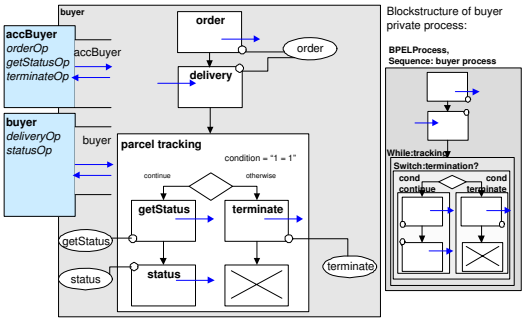


Fig. 3. Buyer BPEL Private Process

3 Process Choreographies

We discuss basic issues related to the evolution of choreographies between partner processes. We show how this can be supported in a (semi-)automated way.

3.1 Overview

For several reasons business processes steadily evolve. Thus process-oriented information systems have to be continuously adapted as well. As long as the modified processes are not part of a process choreography, change effects can be kept local. The same applies if changes of a private process have no impact on related public processes. In general, however, we cannot always assume this. Regarding process choreographies the modification of a private process may not only influence related public processes, but also the public and private processes of partners. As an example take an activity inserted into a private process and invoking an external operation of a partner process (by sending a corresponding

message to it). If the partner process is not adapted accordingly (e.g., by inserting a receive activity processing the message sent) the execution of the modified process choreography could fail. Thus it is crucial to provide adequate methods to (automatically) *propagate* changes of a private process to partner processes.

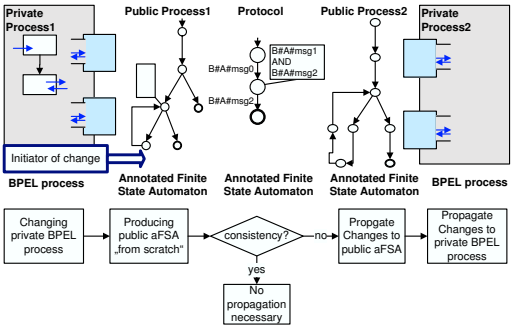


Fig. 4. DYCHOR Approach

Fig. 4 depicts our DYCHOR approach for the evolution of choreographies. Assume that private process 1 (left side) is modified and therefore is regarded as initiator of the following choreography change (if necessary). Then, at first, the public view on this process is recreated in order to reflect changes that might affect the interactions with partner processes. If this results in a modification of public process 1 (and only then) we further check whether adaptations of public process 2 (right side of figure) become necessary as well. This is accomplished by calculating the consistency of the two public processes, i.e., the guarantee of a deadlock free execution of the interaction. In case of inconsistency the change of public process 1 has to be propagated to public process 2[1]; otherwise the execution of the process choreography will fail. DYCHOR exploits semantics of the applied changes in order to automatically adapt public process 2 in such a case. After having performed respective modifications the adaptation of private process 2 also becomes necessary. However, due to the autonomy of the partners and due to the privacy of the mission critical business decisions (represented in the private process), an automatic adaptation of private processes is generally not desired. Nevertheless, the system should assist process engineers in accomplishing this task by suggesting respective adaptations of private process 2.

3.2 Formal Model

The sketched approach (i.e., the correct propagation of private process changes) requires a formal model for representing public processes. Different approaches have been proposed in literature, which can be classified according to their underlying communication model: The models suggested in [6] and [7], for example,

[1] A general correctness criterion for this is provided in Section 4.2.

support asynchronous communication. By contrast synchronous communication is supported by [8]. Since Web services often use synchronous communication based on the HTTP protocol, in the following we apply the annotated Finite State Automata model as introduced in [8].

DYCHOR uses annotated Finite State Automata (aFSA) to represent message sequences that can be handled by a public process. Transitions of such an aFSA are labeled, whereas a label $A\#B\#msg$ indicates that party A sends message msg to party B (see, for example, the left aFSA in Fig. 5). Furthermore, aFSAs can differentiate between mandatory and optional messages. This is achieved by annotating states with logical expressions. In the right aFSA from Fig. 5, for example, the depicted conjunctive annotation expresses that both messages $B\#A\#msg1$ and $B\#A\#msg2$, which may be sent by party B, have to be supported by a trading partner. Thus the messages are mandatory. Obviously, the aFSAs of two interacting public processes must meet certain constraints in order to ensure correct execution of the respective process choreography. We formalize this and summarize basic aFSA characteristics necessary for the further understanding. Hence, we introduce the definition of formulas[2] used in the annotations, before introducing the aFSA.

Definition 1 ***(Definition of Formulas)***
The syntax of the supported logical formulas is given as follows: (i) the constants true and false are formulas, (ii) the variables $v \in \Sigma$ are formulas, where Sigma is a finite set of messages, (iii) if ϕ is a formula, so is $\neg\phi$, (iv) if ϕ and ψ are formulas, so is $\phi \wedge \psi$ and $\phi \vee \psi$. – The set of all formulas is defined as E.

Based on the set of formulas E the standard Finite State Automaton (FSA) [10] is extended as follows:

Definition 2 ***(annotated Finite State Automaton (aFSA))***
An annotated Finite State Automaton A is represented as a tuple $A = (Q, \Sigma, \Delta, q_0, F, QA)$ where Q is a finite set of states, Σ is a finite set of messages, $\Delta : Q \times \Sigma \times Q$ represents labeled transitions, $q_0 \in Q$ is a start state, $F \subseteq Q$ constitutes a set of final states, and $QA : Q \times E$ is a finite relation of states and logical terms within the set E of formulas.

The graphical representation of an annotated Finite State Automaton (aFSA) is based on the usual representation of FSA. States are represented as circles and transitions as arcs (annotated with labels). Final states are depicted as states with thick line. In addition to FSA, an aFSA can have state annotations (denoted as squares connected to the respective states). Fig. 5 shows two aFSA examples: Transitions are labeled whereas a label represents a message exchanged between party A and party B.

In our DYCHOR framework, a public process (in terms of aFSA models) can be automatically derived from the specification of a private one. In [11], for a subset of BPEL, we have provided respective mapping rules. Based on the given

[2] The logical formulas are specified adapting the definition in [9].

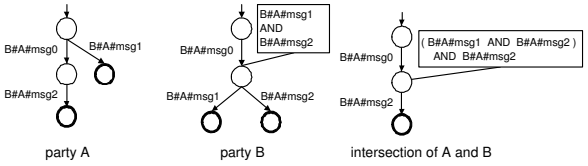


Fig. 5. aFSA Representation

aFSA definition, intersection and emptiness operations can be defined (cf. [8]), which are quite similar to the ones of standard FSA.

Definition 3 ***(Intersection of two aFSAs)***
Let $A_1 = (Q_1, \Sigma_1, \Delta_1, q_{10}, F_1, QA_1)$ *and* $A_2 = (Q_2, \Sigma_2, \Delta_2, q_{20}, F_2, QA_2)$ *be two aFSA. The intersection* $A := A_1 \cap A_2$ *of these automata is given by* $A = (Q, \Sigma, \Delta, q_0, F, QA)$*, with:* $Q = Q_1 \times Q_2$*,* $\Sigma = \Sigma_1 \cap \Sigma_2$*,* $q_0 = (q_{10}, q_{20})$*,* $F = F_1 \times F_2$*,* $\Delta = \{((q_{11}, q_{21}), \alpha, (q_{12}, q_{22}))|\beta \in \{\alpha, \varepsilon\}, (q_{11}, \beta, q_{12}) \in \Delta_1, (q_{21}, \beta, q_{22}) \in \Delta_2\}$*, and* $QA = \bigcup_{(q_1,e_1)\in QA_1,(q_2,e_2)\in QA_2}\{((q_1, q_2), e_1 \wedge e_2)\}$

In particular, the intersection of two aFSAs is based on the usual cross product construction of automata intersection, where state annotations are combined by conjunction. Fig. 5 illustrates the intersection applied on party A and B. Note that the resulting aFSA only contains those transitions that can be processed by both automata. The annotation in the intersection automaton is the conjunction of the annotation contained in party B and the default annotation of party A, that is, $B\#A\#msg2$, resulting in $(B\#A\#msg1 \wedge B\#A\#msg2) \wedge B\#A\#msg2$.

Based on the intersection automaton, it can be checked whether the accepted language is empty. Emptiness means that the a set of message exchanges exists, where all associated mandatory transitions are supported by a trading partner's aFSA. Again this emptiness test is based on standard automaton emptiness test, where it is checked whether the automaton contains a single path to a final state. Regarding aFSAs this emptiness test has to be extended by requiring that all transitions of a conjunction associated to a single state are available in the automaton and a final state can be reached following each of these transitions. As a consequence, two automata are consistent, if their intersection is non-empty; i.e., there is at least one path from the start to a final state, where each formula annotated to a state on this path evaluates to *true*. A variable becomes *true*, if there is a transition labeled equally to the variable from the current state to another state where the annotation evaluates to *true*. Finally the automaton is non-empty, if the annotation of the start state is *true*.

For the above example the intersection automaton for parties A and B is depicted in Fig. 5. This aFSA is empty since it does not contain the mandatory transition labeled $B\#A\#msg1$: The variable $B\#A\#msg2$ of the annotation evaluates to *true* since there is a path to a final state. By contrast the variable $B\#A\#msg1$ is evaluated to $false$ because there is no such transition available at that state providing a path to a final state.

The non-emptiness of the intersection of two automata guarantees for the absence of deadlock with respect to the execution of these two automata. This property can be derived due to the differentiation between mandatory and optional messages in an automaton. Deadlock freeness is also called *consistency*. If consistency is defined between two parties then we call it *bilateral consistency*.

3.3 Public Process Generation

We assume the private process being specified with BPEL. We sketch how BPEL "blocks" from a private process have to be mapped to states of the related public process (represented by an aFSA). As we will see later, this mapping is useful when changing process choreographies. In this context it is not worth applying the mapping on the originator side of a change. However, when propagating changes of a public process to its underlying private process the mapping can be used to determine the blocks in the private process to be modified.

The mapping is illustrated on behalf of the buyer process. It is based on a depth first traversal of the BPEL structure where each block represents a part of the automaton. As a consequence of the strict nesting of a BPEL document, the names of the blocks are associated with a particular state of the resulting automaton model. Regarding the private and public part of the buyer process (cf. Fig. 3 + 6 a) we obtain the mapping shown in Table 1. It represents the relation between the state numbers (aFSA of the public process) and the BPEL block names (BPEL specification of private process) of the private and the public process. Note that a single state in the public process may be assigned to several BPEL elements since, in general, not all elements have an effect on the public process. As a consequence, the required modifications can be limited to the first block mentioned due to the depth first traversal of the private process.

Table 1. Buyer Mapping Table

State Number	BPEL Block Name
1	BPELProcess, Sequence:buyer process
2	Sequence:buyer process
3	Sequence:buyer process, While:tracking, Switch:termination?, Sequence:cond continue, Sequence:cond terminate
4	Sequence:cond continue
5	Sequence:cond terminate

3.4 View Generation

As a basis for bilateral consistency checking, it has to be ensured that the processes to be compared are representing the bilateral message exchanges only.

Fig. 6. (a) Buyer Public Process (b)Accounting Public Process

Deriving the bilateral view of a public process is illustrated next on behalf of the accounting process. The accounting private process (cf. Fig. 2) can be transformed in a public process (cf. Fig. 6 b).

The view $\tau_P(wf)$ of party P on the public process wf is generated by relabeling all transitions not related to P. E.g., in the buyer view $\tau_{Buyer}(Acc)$ of the accounting process, messages exchanged with Logistics are relabeled with the empty word ε. Effected messages are $A\#L\#deliverOp$, $L\#A\#deliver\_confOp$, $A\#L\#terminateLOp$, $A\#L\#get\_statusOp$, and $L\#A\#get\_statusOp$. The minimized Buyer view of the Accounting public process is shown in Fig. 7a. Applying the same method for Logistics results in the automaton depicted in Fig. 7b.

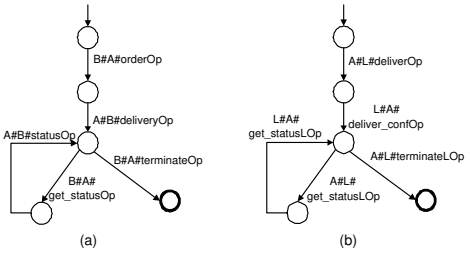


Fig. 7. Accounting Public Process: (a) Buyer View (b) Logistics View

4 Process Choreography Evolution

DYCHOR classifies process changes in two dimensions: The first one (*change framework*) specifies whether the change adds message sequences to an automaton (*additive* change) or removes messages from it (*subtractive* change). The second dimension (*change propagation*) indicates whether a process change has effects on trading partners or not, i.e., whether the protocol the trading partners agreed has to be modified (*variant* change) or not (*invariant* change).

4.1 Change Framework

We give a definition for the difference between two aFSAs, which is used to characterize two basic kinds of change operations on public processes.

Definition 4 ***(Difference of two aFSA)***
Let $A_1 = (Q_1, \Sigma_1, \Delta_1, q_{10}, F_1, QA_1)$ *and* $A_2 = (Q_2, \Sigma_2, \Delta_2, q_{20}, F_2, QA_2)$ *be two*

aFSA. The difference $A := A_1 \setminus A_2$ *of these two aFSA is given by* $A = (Q, \Sigma, \Delta, q_0, F, QA_1)$ *with:* $Q = Q_1 \times Q_2$, $\Sigma = \Sigma_1 \cap \Sigma_2$, $q_0 = (q_{10}, q_{20})$, $F = F_1 \times (Q_2 \setminus F_2)$, $\Delta = \{((q_{11}, q_{21}), \alpha, (q_{12}, q_{22}))|\beta \in \{\alpha, \varepsilon\}, (q_{11}, \beta, q_{12}) \in \Delta_1, (q_{21}, \beta, q_{22}) \in \Delta_2\}$

This definition requires that the automata are complete; i.e, for every state there exists an outgoing transition for each element of the alphabet Σ. In this paper we focus on additive and subtractive changes and their application to aFSAs. Based on such basic change operations more complex changes can be defined. Our framework considers other operations (e.g., to shift process activities) as well as complex changes (defined by applying a set of basic changes operations). Their treatment, however, is outside the scope of the paper. Based on the difference operator we can give a formal definition for additive/subtractive changes:

Definition 5 ***(Additive / Subtractive Change Operations)***
Let A be the aFSA of a public process and let δ be a change operation which transforms A into another aFSA A'. Then:

- *δ is an additive change operation* $:\Longleftrightarrow A' \setminus A \neq \emptyset$
- *δ is a subtractive change operation* $:\Longleftrightarrow A \setminus A' \neq \emptyset$

Based on this definition additive (subtractive) changes of an aFSA correspond to the addition (deletion) of potential message sequences to (from) this aFSA. Note that this does not relate to the structural complexity of the respective private or public processes.

4.2 Propagation Criterion and Invariant Changes

Let A and B be the aFSAs of two public partner processes and let $A \cap B \neq \emptyset$ be the protocol (choreography) between them. If A is changed to A' (by applying change operation δ) the challenging question is whether δ has to be propagated to B or not. Intuitively, no propagation is needed if the protocols before and after applying δ are equivalent. Formally:

$$A \cap B \equiv A' \cap B \Longleftrightarrow (A \setminus A') \cap B = \emptyset \wedge (A' \setminus A) \cap B = \emptyset$$

This constraint, however, is too restrictive since we can also ignore options that are completely under the control (i.e., are to be decided) by the party having performed the change. More precisely, no propagation is needed if $A' \cap B \neq \emptyset$ (assuming that A and B have been bilaterally consistent before the change).

Definition 6 *(Variant and Invariant Changes)*
Let A and B be the aFSAs of two public processes which are consistent, i.e., $A \cap B \neq \emptyset$. *Let δ be a change operation which transforms A into another aFSA A'. Then:*

- *δ is an invariant change* $:\Longleftrightarrow A' \cap B \neq \emptyset$
- *δ is a variant change* $:\Longleftrightarrow A' \cap B = \emptyset$

The aFSA B expresses all options it considers as being mandatory for the respective public process. Thus if public process A' has been changed in a way such that these options are no longer met, change propagation becomes necessary. Accordingly we can state that changes are invariant (i.e., no change propagation is needed) if the intersection between A' and B does not become empty. Note that this can apply for both additive and subtractive changes.

In summary, if the changed public process A' is still consistent with the public process B of a partner it is considered as being invariant and no further actions are needed. By contrast if A' and B turn out to be inconsistent, additional actions become necessary in order to guarantee the successful execution of the processes. How corresponding actions look like is discussed in the following section.

5 Selected Evolution Scenarios

In the following, we provide methods for the propagation of additive changes to partner processes. Due to lack of space we omit a discussion of further changes here (for details on, for example, subtractive changes see [12]).

5.1 Invariant Additive Change

At first we consider invariant additive changes. For example, assume that the accounting process wants to provide an additional order message format to buyers. This change can be realized by adding an alternative activity (order_2) to the accounting process which then receives and processes respective messages *B#A#order_2Op* (cf. Fig. 8).

Since the added message *B#A#order_2Op* is received by the accounting workflow, the buyer view on the respective public process changes (cf. Fig. 9a). However, from the viewpoint of the buyer this change does not require an immediate treatment and propagation to its public and private process. The reason is that the intersection automaton (cf. Fig. 9b) of the modified public view of the buyer on the accounting process and the buyer's current public process (cf Fig. 6) is non-empty. Thus, no change propagation is required.

Invariant subtractive changes can be handled accordingly and are therefore not further treated in this paper. Generally, when adding received messages to a process or removing sent messages from it we can obtain invariant changes.

5.2 Variant Additive Changes

The formal basis for variant additive changes is provided by Def. 5 and Def. 6: Let A and B be the aFSAs of two public processes and let $A \cap B \neq \emptyset$ be the protocol between them. Let further δ be a change operation transforming A into A'. Then: δ is called *variant additive change* if the following constraint holds: $A' \setminus A \neq \emptyset \ \wedge \ A' \cap B = \emptyset$.

According to Def. 6 change propagation to B and the related private process become necessary now. We illustrate this scenario by an example. Assume that

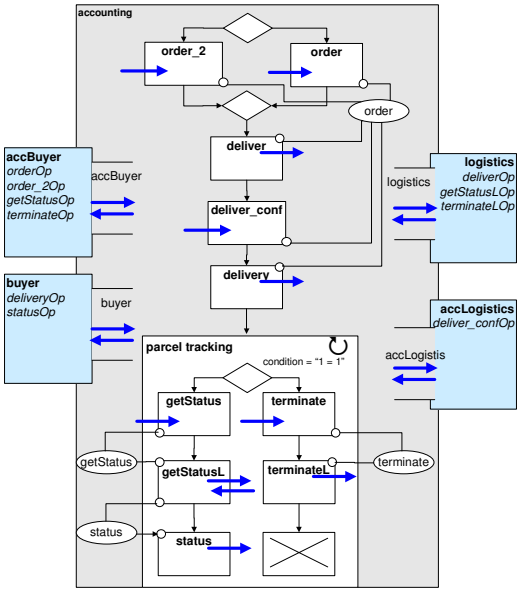


Fig. 8. Invariant Change of Accounting Private BPEL Process

the accounting private process shall be extended with the option to cancel orders (e.g., due to a product being out of stock). This change can be accomplished by adding a respective decision node and an activity to send the cancel message to the buyer ($A\#B\#cancelOp$) – the result is depicted in Fig. 10.

Next we derive the new version of the accounting public process and apply the buyer view on it (cf. Fig. 11a). Then we calculate the intersection of this automaton with the one of the buyer public process (cf. Fig. 6) which results in the automaton shown in Fig. 11b). Note that this automaton is empty since there exists no transition labeled $A\#B\#cancelOp$ on any path to a final state. This makes the annotation containing this message invalid and therefore results in an empty automaton. As a consequence, the variant change of the accounting process has to be propagated to the buyer process.

We now sketch the steps necessary for propagating addiditve changes to the opponent's private/public process:

1. Recalculate the opponent's public view on the new public process of the change originator and determine the newly inserted sequence (i.e., the messages potentially exchanged with the opponent's public process).
2. Calculate the union of the opponent's current public process and the newly introduced message sequence (cf. Step 1) The resulting aFSA provides the basis for potential adaptations of the opponent's public process.
3. Based on the outcome of Step 2 we can derive those regions of the opponent's private process where adaptations may have to be performed.

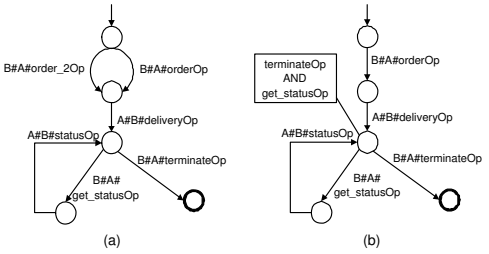


Fig. 9. (a) Public Buyer View on Accounting Process After Invariant Change (b) Intersection of a) with Buyer Public Process

4. Perform the necessary changes of the opponent's private process.
5. Recalculate the opponent's public process. If it is consistent with the public process of the change originator we are finished. Otherwise, go back to the previous step and repeat it with a modified set of changes.

We explain the different steps along our example:

ad 1) We determine the changes of the buyer view on the accounting public process A'. Based on this we calculate potential adaptations of the buyer public process B. More precisely we determine $A'' := \tau_{Buyer}(A') \setminus B$ (cf. Fig. 11c). In general, we have to consider the difference $\tau_{Buyer}(A') \setminus (\tau_{Buyer}(A) \cap B)$. However, since only those message sequences must be added to B which have not been contained before, derivation of $\tau_{Buyer}(A') \setminus B$ is sufficient.

ad 2) We calculate the union of the described difference (cf. Step 1) with the original buyer public process. Based on this we can derive potential changes of the buyer public process. – The union of two aFSAs can be created using the complement and intersection operator in accordance to the deMorgan law: $A \cup B \equiv \overline{\overline{A} \cap \overline{B}}$; thus, $B' := A'' \cup B$ (cf. Fig. 11d).

ad 3) We apply the potential changes to the buyer public process. The regions to be adapted in the corresponding private process can then be derived from the states that have been modified for the buyer public process. For this we use the mapping that was created when generating the buyer public process out of the corresponding private one. Note that observable states can be mapped to a particular process region and that non-present transitions provide a hint on what is missing exactly.

In order to derive the states which have been changed when transforming aFSA B to aFSA B' the difference automaton is traversed parallel to the original public process (comparable to bisimulation). In particular, the first state where the difference automaton contains a transition which is not contained in the original public process, indicates the state where a new transition has been added. The missing transition indicates which message has to be additionally considered by the private process. With regard to the Buyer public process, this is the case for state no. 2 in the original public process as depicted in Fig. 6.

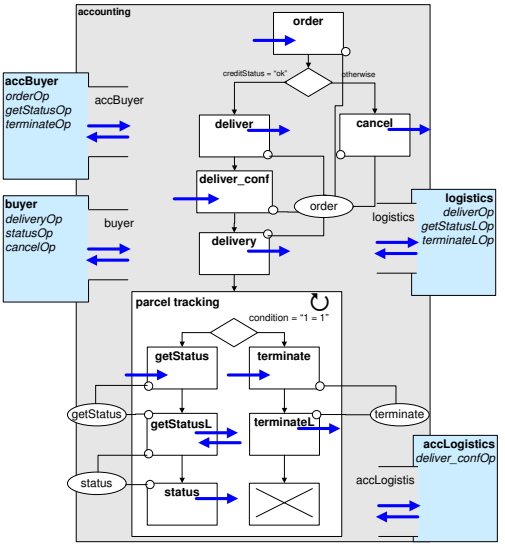


Fig. 10. Additive Change of Accounting Private BPEL Process

From the mapping table we can derive that the change in the Buyer private process is related to the block specified by the sequence activity labeled "buyer process". The receive delivery activity contained in the sequence has to be changed to a pick activity allowing to receive either the delivery or the cancel message. Information to be added can be derived from the difference automaton depicted in Fig. 11c). Fig. 12 shows the resulting Buyer private process.

ad 4) and 5) Finally, we perform the change of the private process accordingly and recalculate the public process on it. After this we check whether intersection of the changed buyer public process and the buyer view of the accounting public process is non-empty, i.e. whether related aFSAs are bilaterally consistent.

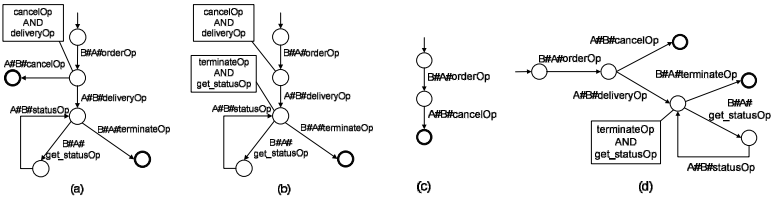


Fig. 11. Accounting Process: (a) Public (Buyer) View (b) Intersection of a) with Buyer Public View (c) Difference at Buyer View of Accounting Public Process (minimized) (d) New Buyer Public Workflow (minimized)

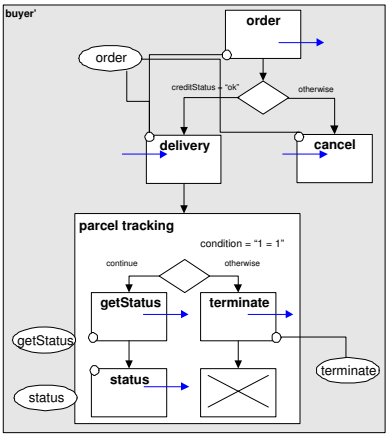


Fig. 12. Buyer Process after Propagation of Additive Changes

6 Implementation Issues

We have implemented the core of the presented approach in a proof-of-concept prototype [8]. Further, a partial mapping from BPEL (private processes) to annotated Finite State Automata (public processes) has been realized [11] and been used for implementing service discovery [13]. The extension of classical UDDI proposed in this context uses BPEL specifications of public processes and bilateral consistency to improve the precision of service discovery results. Finally, we have proposed a protocol to derive a potential cross-organizational process (i.e., a potential service composition in Web Service notation) in a decentralized way resulting in a set of services as a basis for consistency checking [14].

These building blocks can be used for setting up the concrete change framework of DYCHOR. As indicated the only information which has to be exchanged between partners is about the changes applied to public processes. The difference calculation as well as the necessary adaptations of the own public and private processes can be accomplished locally. Finally, decentralized consistency checking can be applied to guarantee the successful introduction of the changes and the consistency of the changed choreography.

7 Related Work

Handling of changes is known in software engineering as refactoring like e.g. [15], where the aim is to propagate changes without altering the behavior of the application. These approaches propagate changes on a centrally available source base and do not know different abstractions layers comparable to choreographies and orchestrations as discussed in this paper.

Checking consistency of a cross-organizational process can be based on the set of potential execution sequences. A straightforward approach is to check consistency on a centralized process representation, which has to be split into several

public processes afterwards. This principle was applied to different process description formalism in the past, like Workflow Nets (WF Nets) [3], guarded Finite State Automata [16], Colored Place/Transition Nets [17], and Statecharts [18]. However, these top-down approaches are based on centralized consistency checking, which is different to the DYCHOR approach described in this paper.

By contrast, the bottom-up approach of constructing the cross-organizational process out of several public processes has not been investigated in sufficient detail so far. Respective proposals have been made, for example, in [6,16,7]. However, they require centralized decision making and are also not constructive; i.e., they only specify criteria for various notions of consistency but do not provide an approach to adapt public processes in a way making the overall cross-organizational process consistent. In addition, these approaches neither address synchronous communication nor allow for decentralized consistency checking.

Issues related to dynamic workflow change have been investigated in detail (e.g., [19,20,1]). Respective approaches address ad-hoc changes of single process instances as well as process schema evolution (i.e., controlled change of process types and propagation of these modifications to already running process instances [20,1]). However, these approaches focus on the adaptation of process orchestrations, i.e., process instances controlled by a single endpoint. By contrast, issues related to changes of process choreographies have been neglected so far. What can be learned from approaches dealing with dynamic changes of process orchestrations is the idea of controlled change propagation. These approaches aim at propagating process type changes to running process instances without causing inconsistencies or errors. Similarly, we have provided an approach for the controlled propagation of the changes of private processes within a choreography to the choreography itself and the respective partner processes.

8 Conclusion and Future Work

In this paper we have presented the DYCHOR framework for introducing changes to private processes, for recalculating related public views automatically, and for propagating resulting modifications to partner processes if required. We have provided a formal model and precise criteria allowing us to automatically decide which adaptations become necessary due to changes of private partner processes. The treatment of different change scenarios adds to the completeness of our approach. Finally, we have implemented the basic mechanisms presented in this paper in a proof-of-concept prototype. We will analyze how to adopt the implemented approach within a real application scenario.

Another challenging issue is the treatment of running process instances (participating in a choreography) when changing private and public process models. In particular, for long-running choreographies, the propagation of choreography changes to already running instances is highly desirable. In future work we will address this issue by elaborating different strategies. These strategies range from managing change propagation by a central coordinator to completely decentralized solutions (e.g., optimistic vs. pessimistic instance migrations). The ultimate

challenge will be to address the problem of concurrent process choreography and process instance changes, i.e., how to propagate process choreography changes to process instances which have already been subject to ad-hoc changes themselves. Meeting these challenges will be key for future service-oriented infrastructures, ultimately resulting in highly adaptive process choreographies.

References

1. Rinderle, S., Reichert, M., Dadam, P.: Correctness criteria for dynamic changes in workflow systems – a survey. DKE **50** (2004) 9–34
2. Rinderle, S., Reichert, M., Dadam, P.: Flexible support of team processes by adaptive workflow systems. Distributed and Parallel Databases **16** (2004) 91–116
3. Aalst, W., Weske, M.: The P2P approach to interorganizational workflows. In: Proc. CAiSE'06, Interlaken, Switzerland (2001)
4. Andrews et al., T.: Bpel4ws v 1.1 (2003)
5. RosettaNet: RosettaNet home page. http://www.rosettanet.org (2004)
6. Aalst, W.: Interorganizational workflows: An approach based on message sequence charts and petri nets. Systems Analysis - Modelling - Simulation **34** (1999) 335–367
7. Kindler, E., Martens, A., Reisig, W.: Inter-operability of workflow applications: Local criteria for global soundness. In: Business Process Management, Models, Techniques, and Empirical Studies, Springer-Verlag (2000) 235–253
8. Wombacher, A., Fankhauser, P., Mahleko, B., Neuhold:, E.: Matchmaking for business processes based on choreographies. IJWS **1** (2004) 14–32
9. Chomicki, J., Saake, G., eds.: Logics for Database and Information Systems. Kluwer (1998)
10. Hopcroft, J.E., Motwani, R., Ullman, J.D.: Introduction to Automata Theory, Languages, and Computation. Addison Wesley (2001)
11. Wombacher, A., Fankhauser, P., Neuhold, E.: Transforming BPEL into annotated deterministic finite state automata enabling process annotated service discovery. In: Proc. of Intl. Conf. on Web Services (ICWS). (2004) 316–323
12. Rinderle, S., Wombacher, A., Reichert, M.: On the controlled evolution of process choreographies. Technical Report TR-CTIT-05-47, University of Twente (2005)
13. Wombacher, A., Mahleko, B., Neuhold, E.: IPSI-PF: A business process matchmaking engine. In: Proc. of Conf. on Electronic Commerce (CEC). (2004) 137–145
14. Wombacher, A.: Decentralized decision making protocol for service composition. In: Proc IEEE Int Conf on Web Services (ICWS). (2005) (accepted for publication).
15. Mens, T., Tourwe, T.: A survey of software refactoring. IEEE Transactions on Software Engineering **30** (2004) 126–139
16. Fu, X., Bultan, T., Su, J.: Realizability of conversation protocols with message contents. In: Proc. IEEE Intl. Conf. on Web Services (ICWS). (2004) 96–103
17. Yi, X., Kochut, K.J.: Process composition of web services with complex conversation protocols. In: Proc. Conf. on Design, Analysis, and Simulation of Distributed Systems Symposium at Adavanced Simulation Technology. (2004) 141–148
18. Wodtke, D., Weikum, G.: A formal foundation for distributed workflow execution based on state charts. In: Proc. ICDT'06. (1997) 230–246
19. v.d. Aalst, W., Basten, T.: Inheritance of workflows: An approach to tackling problems related to change. Theoret. Comp. Science **270** (2002) 125–203
20. Casati, F., Ceri, S., Pernici, B., Pozzi, G.: Workflow evolution. DKE **24** (1998) 211–238

Worklets: A Service-Oriented Implementation of Dynamic Flexibility in Workflows

Michael Adams[1], Arthur H.M. ter Hofstede[1], David Edmond[1], and Wil M.P. van der Aalst[1,2]

[1] Business Process Management Group
Queensland University of Technology, Brisbane, Australia
{m3.adams, a.terhofstede, d.edmond}@qut.edu.au
[2] Department of Technology Management
Eindhoven University of Technology, Eindhoven, The Netherlands
w.m.p.v.d.aalst@tm.tue.nl

Abstract. This paper presents the realisation, using a Service Oriented Architecture, of an approach for dynamic flexibility and evolution in workflows through the support of flexible work practices, based not on proprietary frameworks, but on accepted ideas of how people actually work. A set of principles have been derived from a sound theoretical base and applied to the development of *worklets*, an extensible repertoire of self-contained sub-processes aligned to each task, from which a dynamic runtime selection is made depending on the context of the particular work instance.

1 Introduction

Workflow management systems are used to configure and control structured business processes from which well-defined workflow models and instances can be derived [1,2]. However, the proprietary process definition frameworks imposed make it difficult to support (i) dynamic evolution (i.e. modifying process definitions during execution) following unexpected or developmental change in the business processes being modelled [3]; and (ii) deviations from the prescribed process model at runtime [4,5,6].

Without support for dynamic evolution, the occurrence of a process deviation requires either suspension of execution while the deviation is handled manually, or an entire process abort. However, since most processes are long and complex, neither manual intervention nor process termination are satisfactory solutions [7]. Manual handling incurs an added penalty: the corrective actions undertaken are not added to 'organisational memory' [8,9], and so natural process evolution is not incorporated into future iterations of the process. Other evolution issues include problems of migration, synchronisation and version control [4,10].

These limitations mean a large subset of business processes do not easily map to the rigid modelling structures provided [11], due to the lack of flexibility inherent in a framework that, by definition, imposes rigidity. Process models are 'system-centric', or *straight-jacketed* [12] into the supplied framework, rather

R. Meersman, Z. Tari et al. (Eds.): OTM 2006, LNCS 4275, pp. 291–308, 2006.

than truly reflecting the way work is actually performed [13]. As a result, users are forced to work outside of the system, and/or constantly revise the static process model, in order to successfully support their activities, thereby negating the efficiency gains sought by implementing a workflow solution in the first place.

Since the mid-nineties many researchers have worked on problems related to workflow change (cf. Section 7). This paper is based on and extends the approach proposed in [14]. It introduces a realisation of *'worklets'*, an extensible repertoire of self-contained sub-processes and associated selection rules, grounded in a formal set of work practice principles called *Activity Theory*, to support the modelling, analysis and enactment of business processes. This approach directly provides for dynamic change and process evolution without having to resort to off-system intervention and/or system downtime. It has been implemented as a discrete service for the well-known, open-source workflow environment YAWL [15,16] using a Service Oriented Architecture (SOA), and as such its applicability is not limited to that environment. Also, being open-source, it is freely available for use and extension.

The paper is organised as follows: Section 2 provides a brief overview of Activity Theory and lists relevant principles derived from it by the authors, then introduces the worklet paradigm. Section 3 describes the implementation of the discrete worklet service. Section 4 details the worklet service architecture. Section 5 discusses process definition methods, while Section 6 describes how the worklet approach utilises *Ripple Down Rules* (RDR) to achieve contextual, dynamic selection of worklets at runtime. Section 7 discusses related work, and finally Section 8 outlines future directions and concludes the paper.

2 Achieving Flexibility Through Worklets

Workflow management systems provide support for business processes that are generally predictable and repetitive. However, the prescriptive, assembly-line frameworks imposed by workflow systems limit the ability to model and enact flexible work practices where deviations are a normal part of every work activity [12,17]. For these environments, formal representations of business processes may be said to provide merely a contingency around which tasks can be formulated dynamically [18], rather than a prescriptive blueprint that must be strictly adhered to.

Rather than continue to try to force business processes into inflexible frameworks (with limited success), a more adaptable approach is needed that is based on accepted ideas of how people actually work.

A powerful set of descriptive and clarifying principles that describe how work is conceived, performed and reflected upon is *Activity Theory*, which focusses on understanding human activity and work practices, incorporating notions of intentionality, history, mediation, collaboration and development [19]. (A full exploration of Activity Theory can be found in [20,21]). In [22], the current authors undertook a detailed study of Activity Theory and derived from it a

set of principles that describe the nature of participation in organisational work practices. Briefly, the relevant principles are:

1. Activities (i.e. work processes) are *hierarchical* (consist of one or more actions), *communal* (involve a community of participants working towards a common objective), *contextual* (conditions and circumstances deeply affect the way the objective is achieved), *dynamic* (evolve asynchronously), and *mediated* (by tools, rules and divisions of labour).
2. Actions (i.e. tasks) are undertaken and understood contextually. A *repertoire* of applicable actions is maintained and made available for each action of an activity; the activity is performed by making contextual choices from the repertoire of each action in turn.
3. A work plan is not a prescription of work to be performed, but merely a guide which may be modified during execution depending on context.
4. Deviations from a plan will naturally occur with every execution, giving rise to learning experiences which can then be incorporated into future instantiations of the plan.

Consideration of these derived principles have led to the conception, development and implementation of a flexible workflow support system that:

- regards the process model as a guide to an activity's objective, rather than a prescription for it;
- provides a repertoire (or catalogue) of applicable actions to be made available for each task at each execution of a process model;
- provides for choices to be made dynamically from the repertoire at runtime by considering the specific context of the executing instance; and
- allows the repertoire of actions to be dynamically extended at runtime, thus incorporating unexpected process deviations, not only for the current instance, but for other current and future instantiations of the process model, leading to natural process evolution.

Thus, each task of a process instance may be linked to an extensible repertoire of actions, one of which will be contextually chosen at runtime to carry out the task. In this work, we present these repertoire-member actions as *"worklets"*. In effect, a worklet is a small, self-contained, complete workflow process which handles one specific task (action) in a larger, composite process (activity)[1]. A top-level or parent process model is developed that captures the entire workflow at a macro level. From that manager process, worklets are contextually selected and invoked from the repertoire of each task when the task instance becomes enabled during execution.

In addition, new worklets for handling a task may be added to the repertoire at any time (even during process execution) as different approaches to completing

[1] In Activity Theory terms, a worklet may represent one action within an activity, or may represent an entire activity.

a task are developed, derived from the context of each process instance. Importantly, the new worklet becomes part of the process model for all current and future instantiations, avoiding issues of version control. In this way, the process model undergoes a dynamic natural evolution.

3 The Worklet Custom Service for YAWL

The *Worklet Dynamic Process Selection Service* has been implemented as a YAWL Custom Service [15,16]. The YAWL environment was chosen as the implementation platform since it provides a very powerful and expressive workflow language based on the workflow patterns identified in [23], together with a formal semantics. It also provides a workflow enactment engine, and an editor for process model creation, that support the control flow, data and (basic) resource perspectives. The YAWL environment is open-source and has a service-oriented architecture, allowing the worklet paradigm to be developed as a service independent to the core engine. Thus the deployment of the worklet service is in no way limited to the YAWL environment, but may be ported to other environments by making the necessary amendments to the service interface. As such, this implementation may also be seen as a case study in service-oriented computing whereby dynamic flexibility in workflows, orthogonal to the underlying workflow language, is provided.

Custom YAWL services interact with the YAWL engine through XML/HTTP messages via certain interface endpoints, some located on the YAWL engine side and others on the service side. Specifically, custom services may elect to be notified by the engine when certain events occur in the life-cycle of nominated process instantiations (i.e. when a workitem becomes enabled, when a workitem is cancelled, when a case completes). On receiving a workitem-enabled event, the custom service may elect to 'check-out' the workitem from the engine. On doing so, the engine marks the workitem as *executing* and effectively passes operational control for the workitem to the custom service. When the custom service has finished processing the workitem it will check it back in to the engine, at which point the engine will mark the workitem as *completed*, and proceed with the process execution.

The worklet service utilises these interactions by *dynamically substituting an enabled workitem in a YAWL process with a contextually selected worklet* – a discrete YAWL process that acts as a sub-net for the workitem and so handles one specific task in a larger, composite process activity.

An *extensible repertoire (or catalogue) of worklets is maintained for each nominated task in a parent workflow process.* Each time the service is invoked for an enabled workitem, a choice is made from the repertoire based on the data attributes and values associated with the workitem, using a set of rules to determine the most appropriate substitution (see Section 6). The workitem is checked out of the YAWL engine, the input variables of the original workitem are mapped to the net-level input variables of the selected worklet, and then the worklet is launched in the engine as a separate case. When the worklet has completed, its net-level output variables are mapped back to the output variables of the

original workitem, which is then checked back into the engine, allowing the original (parent) process to continue.

The worklet executed for a task is run as a separate case in the YAWL engine, so that, from an engine perspective, the worklet and its parent are two distinct, unrelated cases. The worklet service tracks the relationships, data mappings and synchronisations between cases, and creates a process log that may be combined with the engine's process logs via case identifiers to provide a complete operational history of each process.

Worklets may be associated with either an atomic task, or a multiple-instance atomic task. Any number of worklets can form the repertoire of an individual task, and any number of tasks in a particular specification can be associated with the worklet service. A worklet may be a member of one or more repertoires – that is, it may be re-used for several distinct tasks within and across process specifications. In the case of multiple-instance tasks, a separate worklet is launched for each child workitem. Because each child workitem may contain different data, the worklets that substitute for them are individually selected, and so may all be different.

The repertoire of worklets for a task can be added to at any time, as can the rules base used for the selection process, including while the parent process is executing. Thus the service provides for dynamic ad-hoc change and process evolution, without having to resort to off-system intervention and/or system downtime, or requiring modification of the original process specification.

4 Worklet Service Architecture

Figure 1 shows the external architecture of the worklet service. As mentioned previously, the service has been implemented as a Custom YAWL Service [16]. The YAWL engine provides a number of interfaces, two of which are used by the worklet service. Interface A provides endpoints for process definition,

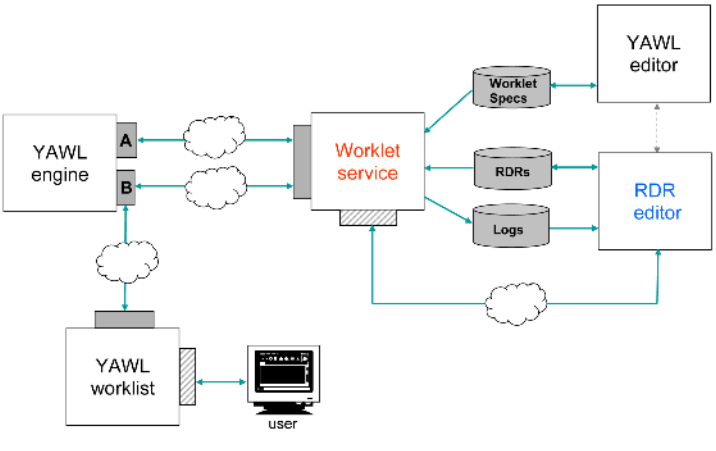


Fig. 1. External Architecture of the Worklet Service

administration and monitoring; Interface B provides endpoints for client and invoked applications and workflow interoperability [16]. The worklet service uses Interface A to upload worklet specifications into the engine, and Interface B for connecting to the engine, to start and cancel case instances, and to check workitems in and out of the engine after interrogating their associated data.

The disk entities 'Worklet specs', 'RDRs' and 'Logs' in Figure 1 comprise the *worklet repository*. The service uses the repository to store rule sets and load them for enabled workitems; to store worklet specifications for uploading to the engine; and to store generated process and audit logs. The YAWL editor is used to create new worklet specifications, and may be invoked from the RDR (Ripple Down Rules) Editor. The RDR Editor is used to create new or augment existing rule sets, making use of certain selection logs to do so, and may communicate with the worklet service via a JSP/Servlet interface to override worklet selections following rule set additions (see Section 6).

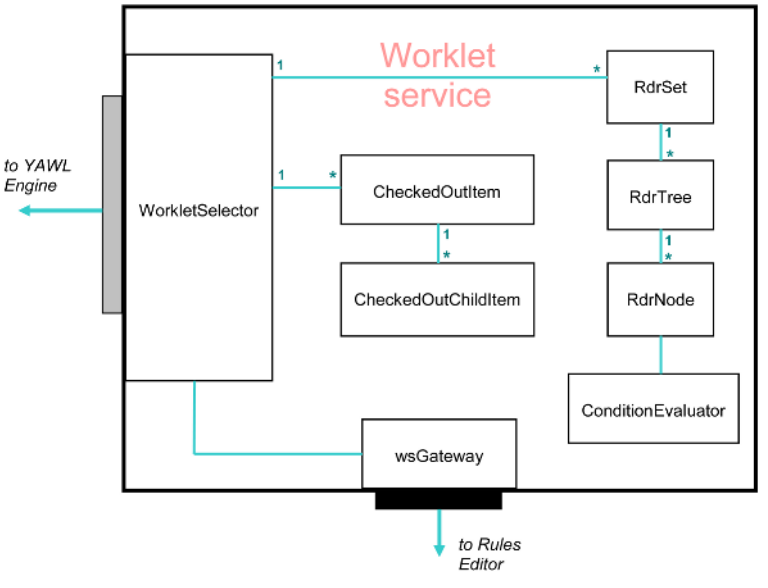


Fig. 2. Internal Architecture of the Worklet Service

Figure 2 shows a representation of the internal architecture of the worklet service. The *WorkletSelector* object handles all interactions with the YAWL engine, and administrates the service. For each workitem that it checks out of the engine, it creates a *CheckedOutItem* object. In YAWL, each workitem is a 'parent' of one or more child items – one if it is an atomic task, or a number of child items in the case of a multiple instance atomic task. Thus, the role of each CheckedOutItem object is to create and manage one or more *CheckedOutChildItems*, which hold information about worklet selection, data associated with the workitem and the results of rules searches.

The WorkletSelector, for each workitem that is checked out from the engine, also loads from file the set of rules pertaining to the specification of which the workitem is a member into an *RdrSet* object. At any time, there may be a number of RdrSets loaded into the service, one for each specification for which a workitem has been checked out. Each RdrSet manages one or more *RdrTree* objects, each tree representing the rule tree for a particular task within the specification, of which this workitem is an instance. In turn, each RdrTree owns a number of *RdrNode* objects, which contain the actual rules, conclusions and other data for each node of the rule tree.

When a rule tree is evaluated against the data set of a workitem, each of the associated nodes of that tree has its condition evaluated by the *ConditionEvaluator* object, which returns the boolean result to the node, allowing it to traverse to its true or false branch as necessary. Finally, the *wsGateway* object provides communications via a JSP/Servlet interface between the service and the Rules Editor (see Section 6 for more details).

5 Process Definition

Fundamentally, a worklet is nothing more than a workflow specification that has been designed to perform one part of a larger, parent specification. However, it differs from a decomposition or sub-net in that it is dynamically assigned to perform a particular task at runtime, while sub-nets are statically assigned at design time. So, rather than being forced to define all possible branches in a specification, the worklet service allows the definition of a much simpler specification that will evolve dynamically as more worklets are added to the repertoire for particular tasks within it.

Figure 3 shows a simple example specification (in the YAWL Editor) for a Casualty Treatment process. Note that this process specification has been intentionally simplified to demonstrate the operation of the worklet service; while it is not intended to portray a realistic process, it is desirable to not camouflage the subject of this paper by using a more complex process specification.

In this process, the *Treat* task is to be substituted at runtime with the appropriate worklet based on the patient data collected in the *Admit* and *Triage* tasks. That is, depending on each patient's actual physical data and reported symptoms, we would like to run the worklet that best treats the patient's condition.

Each task in a process specification may be flagged to notify the worklet service when it becomes enabled. In this example, only the *Treat* task is flagged so; the other tasks are handled directly by the YAWL environment. So, when a Casualty Treatment process is executed, the YAWL Engine will notify the worklet service when the *Treat* task becomes enabled. The worklet service will then examine the data of the task and use it to determine which worklet to execute as a substitute for the task.

A worklet specification is a standard YAWL process specification, and as such is created in the YAWL Editor in the usual manner. Figure 4 shows a very simple

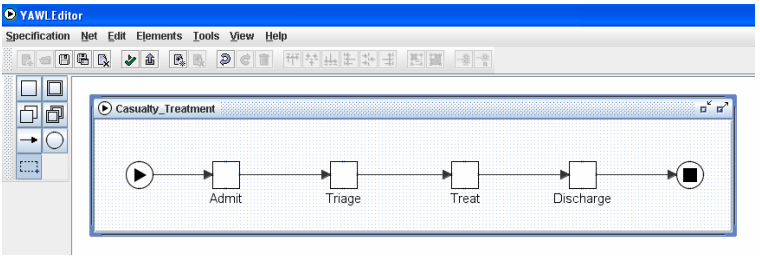


Fig. 3. Parent 'Casualty Treatment' Process

example worklet to be substituted for the *Treat* top-level task when a patient complains of a fever.

In itself, there is nothing special about the Treat Fever specification in Figure 4. Even though it will be considered by the worklet service as a member of the worklet repertoire and may thus be considered a "worklet", it also remains a standard YAWL specification and as such may be executed directly by the YAWL engine without any reference to the worklet service, if desired.

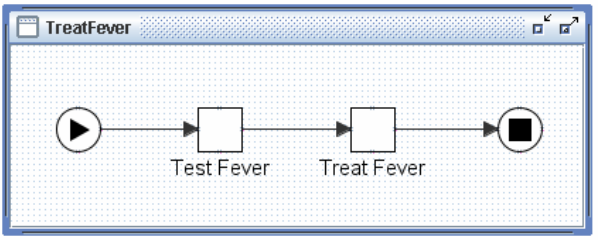


Fig. 4. The 'Treat Fever' Worklet Process

The association of tasks with the worklet service is not restricted to top-level specifications. Worklet specifications also may contain tasks that are associated with the worklet service and so may have worklets substituted for them, so that a hierarchy of executing worklets may sometimes exist. It is also possible to recursively define worklet substitutions - that is, a worklet may contain a task that, while certain conditions hold true, is substituted by another instance of the same worklet specification that contains the task.

Any number of worklets can be created for a particular task. For the Casualty Treatment example, there are currently five worklets in the repertoire for the *Treat* task, one for each of the five conditions that a patient may present with in the *Triage* task: Fever, Rash, Fracture, Wound and Abdominal Pain. In this example, which worklet is chosen for the *Treat* task depends on which of the five is given a value of True in the *Triage* task.

6 Context and Worklet Selection

The consideration of context plays a crucial role in many diverse domains, including philosophy, pragmatics, semantics, cognitive psychology and artificial intelligence [24]. In order to realise the worklet approach, the situated contextual factors relevant to each case instance were required to be quantified and recorded [25] so that the appropriate worklet can be 'intelligently' selected from the repertoire at runtime.

The types of contextual data that may be recorded and applied to a business case may be categorised as follows (examples are drawn from the *Casualty Treatment* process):

- **Generic (case independent):** data attributes that can be considered likely to occur within any process (of course, the data values change from case to case). Such data would include descriptors such as created when, created by, times invoked, last invoked, current status; and role or agent descriptors such as experience, skills, rank, history with this process and/or task and so on. Process execution states and process log data also belong to this category.
- **Case dependent with *a-priori* knowledge:** that set of data that are known to be pertinent to a particular case when it is instantiated. Generally, this data set reflects the data variables of a particular process instance. Examples are: patient name and id, blood pressure readings, height, weight, symptoms and so on; deadlines both approaching and expired; and diagnoses, treatments and prescribed medications.
- **Case dependent with no *a-priori* knowledge:** that set of data that only becomes known when the case is active and deviations from the known process occur. Examples in this category may include complications that arise in a patient's condition after triage, allergies to certain medications and so on.

Each worklet is a representation of a particular situated action, the runtime selection of which relies on the relevant context of each case instance, derived from case data. The worklet selection process is achieved through the use of Ripple Down Rules (RDR), which comprise a hierarchical set of rules with associated exceptions, first devised by Compton and Jansen [26].

The fundamental feature of RDR is that it avoids the difficulties inherent in attempting to compile, *a-priori*, a systematic understanding, organisation and assembly of all knowledge in a particular domain. Instead, it allows for general rules to be defined first with refinements added later as the need arises [27].

An RDR Knowledge Base is a collection of simple rules of the form "if *condition* then *conclusion*" (together with other associated descriptors), conceptually arranged in a binary tree structure. Each rule node may have a false ('or') branch and/or a true ('exception') branch to another rule node, except for the root node, which contains a default rule and can have a true branch only. If a rule is satisfied, the true branch is taken and the subsequent rule is evaluated; if it is not satisfied, the false branch is taken and its rule evaluated [28]. When

a terminal node is reached, if its rule is satisfied, then its conclusion is taken; if its rule is not satisfied, then the conclusion of the last rule satisfied on the path to that node is taken. This tree traversal provides implied *locality* - a rule on an exception branch is tested for applicability only if its parent (next-general) rule is also applicable.

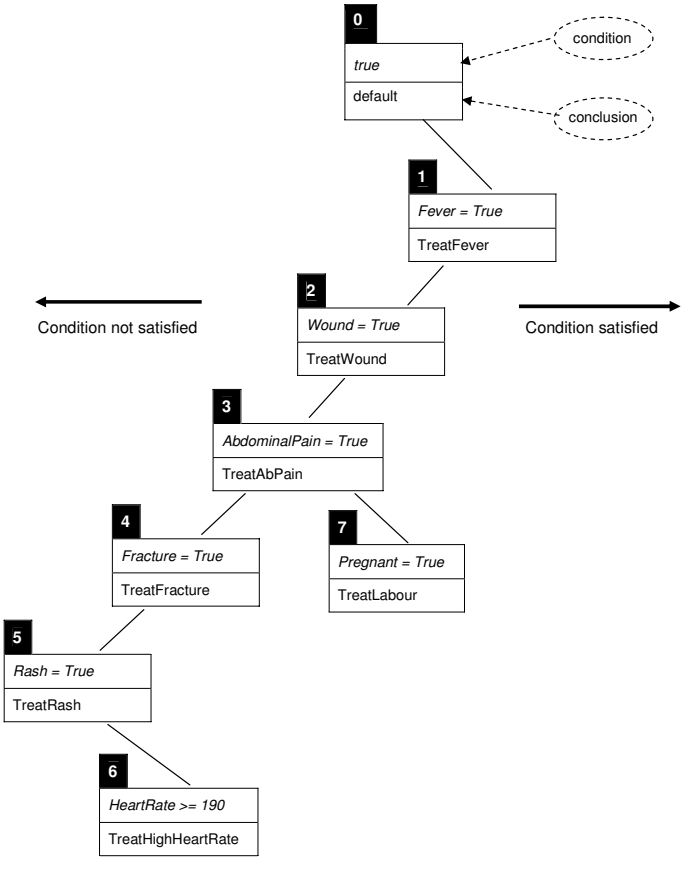


Fig. 5. Conceptual Structure of a Ripple Down Rule (*Casualty Treatment* Example)

A workflow process specification may contain a number of tasks, one or more of which may be associated with the worklet service. For each specification that contains a worklet-enabled task, the worklet service maintains a corresponding set of ripple down rules that determine which worklet will be selected as a substitute for the task at runtime, based on the current case data of that particular instance. Each worklet-enabled task in a specification has its own discrete rule set. The rule set or sets for each specification are stored as XML data in a disk file that has the same name as the specification, except with an ".xrs" extension (XML Rule Set). All rule set files are stored in the worklet repository.

Occasionally, the worklet started as a substitute for a particular workitem, while the correct choice based on the current rule set, is considered by a user to be an inappropriate choice for a particular case. For example, if a patient in a Casualty Treatment case presents with a rash *and* a heart rate of 190, while the current rule set correctly returns the TreatRash worklet, it may be more desirable to treat the racing heart rate before the rash is attended to. In such a case, when the worklet service begins an instance of the TreatRash process, a user may reject it by advising an administrator (via a button on their worklist) of the inappropriate choice. Thus the administrator would need to add a new rule to the rule set so that cases that have such data (both now and in the future) will be handled correctly.

If the worklet returned is found to be unsuitable for a particular case instance, a new rule is formulated that defines the contextual circumstances of the instance and is added as a new leaf node using the following algorithm:

- If the worklet returned was the conclusion of a satisfied terminal rule, then the new rule is added as a local exception node via a new true branch from the terminal node.
- If the worklet returned was the conclusion of a non-terminal, ancestor node (that is, the condition of the terminal rule was not satisfied), then the new rule node is added via a new false branch from the unsatisfied terminal node.

In essence, each added exception rule is a refinement of its parent rule. This method of defining new rules allows the construction and maintenance of the KB by "sub-domain" experts (i.e. those who understand and carry out the work they are responsible for) without regard to any engineering or programming assistance or skill [29].

Each rule node also incorporates a set of case descriptors that describe the actual case that was the catalyst for the creation of its rule. This case is referred to as the 'cornerstone case'. The descriptors of the cornerstone case refer to essential attributes of a case, in this example, the sex, heart rate, age, weight and so on of a patient. The condition for the new rule is determined by comparing the descriptors of the current case to those of the cornerstone case of the returned worklet and identifying a sub-set of differences. Not all differences will be relevant – to define a new rule it is only necessary to determine the factor or factors that make it necessary to handle the current case in a different fashion to the cornerstone case. The identified differences are expressed as attribute-value pairs, using the normal conditional operators. The current case descriptors become the cornerstone case for the newly formulated rule; its condition is formed by the identified attribute-values and represents the context of the case instance that caused the addition of the rule.

A separate Rules Editor tool has been developed to allow for the easy addition of new rules and associated worklets to existing rule sets, and the creation of new rule sets.

Each time the worklet service selects a worklet to execute as a substitute for a specification instance's workitem, a file is created that contains certain

descriptive data about the selection process. These files are stored in the worklet repository, again in XML format. Thus to add a new rule to the existing rule set after an inappropriate selection, the particular selection file for the case that was the catalyst for the rule addition is first loaded into the Rules Editor.

Figure 6 shows the Add New Rule screen of the Rules Editor with a selection file loaded. The Cornerstone Case panel shows the case data that existed for the creation of the original rule for the TreatRash selection. The Current Case panel shows the case data for the current case - that is, the case that is the catalyst for the addition of the new rule. The New Rule Node panel is where the details of the new rule are added. Notice that the ids of the parent node and the new node are shown as read only - the Rules Editor takes care of where in the rule tree the new rule node is to be placed, and whether it is to be added as a true child or false child node, using the algorithm described above.

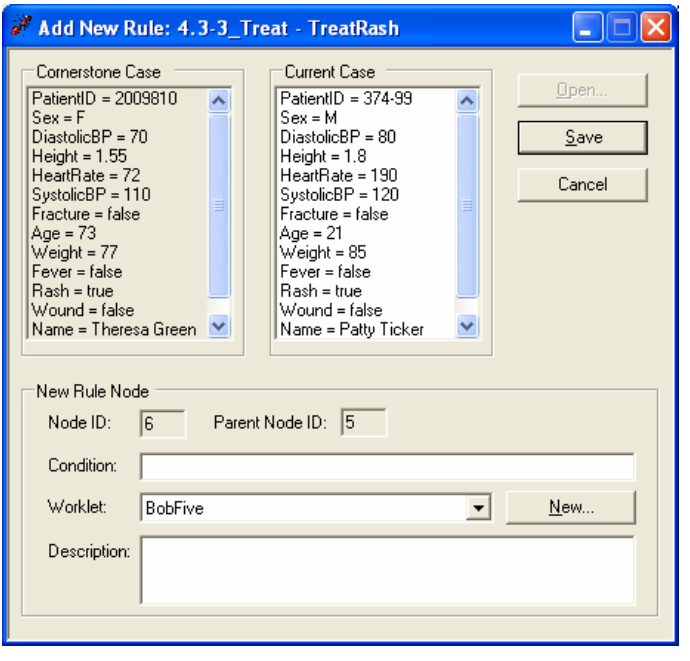


Fig. 6. Rules Editor (Add New Rule Screen)

In this example, there are many data values that differ between the two case data sets shown in Figure 6, such as PatientID, Name, Sex, Blood Pressure readings, Height, Weight and Age. However, the only differing data item of relevance here is HeartRate - that is the only data item that, in this case, makes the selection of the TreatRash worklet inappropriate. Selecting the HeartRate line in the list of Current Case data items will insert it to the condition field, where it may be modified as necessary. In this case, the new rule would become, as an example, "HeartRate $\geq$ 190".

It is not necessary to define a conjunctive rule such as "Rash = True AND HeartRate ≥ 190", since this new rule will be added as an exception to the true branch of the TreatRash node. By doing so, it will only be evaluated if the condition of its parent, "Rash = True", first evaluates to True. Therefore, any rule nodes added to the true branch of a parent node become *exception* rules, and thus refinements, of the parent rule.

After defining a condition for the new rule, the name of the worklet to be executed when this condition evaluates to true must be entered in the Worklet field of the Editor (refer Figure 6). This input is a drop-down list that contains the name of all the worklets currently in the worklet repository. An appropriate worklet for this rule may be chosen from the list, or, if none are suitable, a new worklet specification may be created.

After a new rule is added, the Editor provides an administrator with the choice to replace the previously started (inappropriate) worklet instance with an instance of the worklet defined in the new rule. If the administrator chooses to replace the worklet, the Rules Editor contacts the worklet service via HTTP and requests the change. The service responds with a dialog similar to Figure 7.

Fig. 7. Example Dialog Showing a Successful Dynamic Replacement

7 Related Work

Since the mid-nineties much research has been done on issues related to flexibility and change in workflow management systems (cf. the classification into ad-hoc, administrative, and production workflows in [30]). While it is not the intention of this paper to provide a complete overview of the work done in this area, reference is made here to a number of quite different approaches to providing dynamic flexibility in workflows.

Generally, commercial workflow management systems provide various levels of support for the decomposition of tasks and sub-processing. However, each of the products require the model to be fully defined before it can be instantiated, and changes must be incorporated by modifying the model statically. Staffware provides 're-usable process segments' that can be inserted into any process. SAP R/3 allows for the definition of 'blocks' that can be inserted into other 'blocks', thus providing some support for encapsulation and reuse. COSA supports parent-sibling processes, where data can be passed to/from a process to a sub-process. MQ Workflow allows sub-processes to be defined and called

statically from within a process. Clearly, all of these static forms of decomposition do not offer support for dynamic flexibility.

Among the non-commercial systems, *ADEPT* [31] supports modification of a process during execution (i.e. add, delete and change the sequence of tasks) both at the type (dynamic evolution) and instance levels (ad-hoc changes). Such changes are made to a traditional monolithic model and must be achieved via manual intervention. The *WASA* [32] system provides some support for dynamic change, mainly focusing on scientific applications. It allows an administrator to modify a (monolithic) specification and then restart a task, but then only at the instance level. A catalog of 'skeleton' patterns that can be instantiated or specialised at design time is supported by the *WERDE* system [5]. Again, there is no scope for specialisation changes to be made at runtime. *AgentWork* [33] provides the ability to modify process instances by dropping and adding individual tasks based on events and ECA rules. However, the rules do not offer the flexibility or extensibility of Ripple Down Rules, and changes are limited to individual tasks, rather than the process-for-task substitution provided by the worklet service. Also, the possibility exists for conflicting rules to generate incompatible actions, which requires manual intervention and resolution.

It should be noted that only a small number of academic prototypes have had any impact on the frameworks offered by commercial systems [34]. Nevertheless, there are some interesting commercial products that offer innovative features with respect to flexibility. *Caramba* [35] supports virtual teams in their ad hoc and collaborative processes by enabling links between artifacts (for example, documents and database objects), business processes (activities), and resources (persons, roles, etc.). *FLOWer* supports the concept of case-handling; the process model only describes the preferred way of doing things and a variety of mechanisms are offered to allow users to deviate in a controlled manner [1].

The implementation discussed in this paper differs considerably from the above approaches. Worklets dynamically linked together by extensible Ripple Down Rules provide an alternative method for the provision of dynamic flexibility. An approach with some similarities to worklets is is the *Process Orchestrator*, an optional component of Staffware [36], which provides for the dynamic allocation of sub-processes at runtime. It requires a construct called a "dynamic event" to be explicitly modelled that will execute a number of sub-processes listed in an 'array' when execution reaches that event. Which sub-processes execute depend on predefined data conditionals matching the current case. Unlike the worklet approach, the listed sub-processes are statically defined, as are the conditionals – there is no scope for dynamically refining conditionals, nor adding sub-processes at runtime.

8 Conclusion and Future Work

Workflow management systems impose a certain rigidity on process definition and enactment because they use frameworks based on assembly line metaphors rather than on ways work is actually planned and carried out. An analysis of

Activity Theory provided principles of work practices that were used as a template on which a workflow service has been built that better supports flexibility and dynamic evolution. By capturing contextual data, a repertoire of actions is constructed that allow for contextual choices to be made from the repertoire at runtime to efficiently carry out work tasks. These actions, or worklets, directly provide for process evolution and flexibility, and mirror accepted work practices.

The worklet implementation presents several key benefits, including:

- A process modeller can describe the standard activities and actions for a workflow process, and any deviations, using the same methodology;
- It allows re-use of existing process components and aids in the development of fault tolerant workflows using pre-existing building blocks [37];
- Its modularity simplifies the logic and verification of the standard model, since individual worklets are less complex to build and therefore easier to verify than monolithic models;
- It provides for a variety of workflow views of differing granularity, which offers ease of comprehensibility for all stakeholders;
- It allows for gradual and ongoing evolution of the model, so that global modification each time a business practice changes or a deviation occurs is unnecessary; and
- In the occurrence of an unexpected event, the process modeller needs simply to choose an existing worklet or build a new one for that event, which can be automatically added to the repertoire for current and future use as necessary, thus avoiding manifold complexities including downtime, model restructuring, versioning problems and so on.

This implementation used the open-source, service-oriented architecture of YAWL to develop a service for dynamic flexibility independent to the core engine. Thus, the implementation may be viewed as a successful case study in service-oriented computing. It is the first instalment of a comprehensive approach to dynamic workflow and is intended to be extended in the near future to also provide support for dynamic handling of process exceptions using the same service paradigm. One of the more interesting things to be investigated and incorporated is the application of process mining techniques to the various logs collected by the Worklet service; a better understanding of when and why people tend to "deviate" from a work plan is essential for providing better tool support.

All system files, source code and documentation for YAWL and the worklet service, including the examples discussed in this paper, may be downloaded via `www.yawl-system.com`.

References

1. W.M.P. van der Aalst, Mathias Weske, and Dolf Grünbauer. Case handling: A new paradigm for business process support. *Data & Knowledge Engineering*, 53(2):129–162, 2005.

2. Gregor Joeris. Defining flexible workflow execution behaviors. In Peter Dadam and Manfred Reichert, editors, *Enterprise-wide and Cross-enterprise Workflow Management: Concepts, Systems, Applications*, volume 24 of *CEUR Workshop Proceedings*, pages 49–55, Paderborn, Germany, October 1999.
3. Alex Borgida and Takahiro Murata. Tolerating exceptions in workflows: a unified framework for data and processes. In *Proceedings of the International Joint Conference on Work Activities, Coordination and Collaboration (WACC'99)*, pages 59–68, San Francisco, CA, February 1999. ACM Press.
4. S. Rinderle, M. Reichert, and P. Dadam. Correctness criteria for dynamic changes in workflow systems: A survey. *Data and Knowledge Engineering*, 50(1):9–34, 2004.
5. Fabio Casati. A discussion on approaches to handling exceptions in workflows. In *CSCW Workshop on Adaptive Workflow Systems*, Seattle, USA, November 1998.
6. C.A. Ellis, K. Keddara, and G. Rozenberg. Dynamic change within workflow systems. In N. Comstock, C. Ellis, R. Kling, J. Mylopoulos, and S. Kaplan, editors, *Proceedings of the Conference on Organizational Computing Systems*, pages 10–21, Milpitas, California, August 1995. ACM SIGOIS, ACM Press, New York.
7. Claus Hagen and Gustavo Alonso. Exception handling in workflow management systems. *IEEE Transactions on Software Engineering*, 26(10):943–958, October 2000.
8. Mark S. Ackerman and Christine Halverson. Considering an organization's memory. In *Proceedings of the ACM 1998 Conference on Computer Supported Cooperative Work*, pages 39–48. ACM Press, 1998.
9. Peter A. K. Larkin and Edward Gould. Activity theory applied to the corporate memory loss problem. In L. Svennson, U. Snis, C. Sorensen, H. Fagerlind, T. Lindroth, M. Magnusson, and C. Ostlund, editors, *Proceedings of IRIS 23 Laboratorium for Interaction Technology*, University of Trollhattan Uddevalla, 2000.
10. W.M.P. van der Aalst. Exterminating the dynamic change bug: A concrete approach to support workflow change. *Information Systems Frontiers*, 3(3):297–317, 2001.
11. Jakob E. Bardram. I love the system - I just don't use it! In *Proceedings of the 1997 International Conference on Supporting Group Work (GROUP'97)*, Phoenix, Arizona, 1997.
12. W.M.P. van der Aalst and P.J.S. Berens. Beyond workflow management: Product-driven case handling. In S. Ellis, T. Rodden, and I. Zigurs, editors, *International ACM SIGGROUP Conference on Supporting Group Work*, pages 42–51, New York, 2001. ACM Press.
13. I. Bider. Masking flexibility behind rigidity: Notes on how much flexibility people are willing to cope with. In J. Castro and E. Teniente, editors, *Proceedings of the CAiSE'05 Workshops*, volume 1, pages 7–18, Porto, Portugal, 2005. FEUP Edicoes.
14. Michael Adams, Arthur H. M. ter Hofstede, David Edmond, and W.M.P. van der Aalst. Facilitating flexibility and dynamic exception handling in workflows through worklets. In Orlando Bello, Johann Eder, Oscar Pastor, and João Falcão e Cunha, editors, *Proceedings of the CAiSE'05 Forum*, pages 45–50, Porto, Portugal, June 2005. FEUP Edicoes.
15. W.M.P. van der Aalst and A.H.M. ter Hofstede. YAWL: Yet Another Workflow Language. *Information Systems*, 30(4):245–275, 2005.
16. W.M.P. van der Aalst, L. Aldred, M. Dumas, and A.H.M. ter Hofstede. Design and implementation of the YAWL system. In A. Persson and J. Stirna, editors, *Proceedings of The 16th International Conference on Advanced Information Systems Engineering (CAiSE 04)*, volume 3084 of *LNCS*, pages 142–159, Riga, Latvia, June 2004. Springer Verlag.

17. Diane M. Strong and Steven M. Miller. Exceptions and exception handling in computerized information processes. *ACM Transactions on Information Systems*, 13(2):206–233, 1995.
18. Jakob E. Bardram. Plans as situated action: an Activity Theory approach to workflow systems. In *Proceedings of the 1997 European Conference on Computer Supported Cooperative Work (ECSCW'97)*, pages 17–32, Lancaster U.K., 1997.
19. Bonnie A. Nardi. *Activity Theory and Human-Computer Interaction*, pages 7–16. In Nardi [21], 1996.
20. Y. Engestrom. *Learning by Expanding: An Activity-Theoretical Approach to Developmental Research*. Orienta-Konsultit, Helsinki, 1987.
21. Bonnie A. Nardi, editor. *Context and Consciousness: Activity Theory and Human-Computer Interaction*. MIT Press, Cambridge, Massachusetts, 1996.
22. Michael Adams, David Edmond, and Arthur H.M. ter Hofstede. The application of activity theory to dynamic workflow adaptation issues. In *Proceedings of the 2003 Pacific Asia Conference on Information Systems (PACIS 2003)*, pages 1836–1852, Adelaide, Australia, July 2003.
23. W.M.P. van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski, and A.P. Barros. Workflow patterns. *Distributed and Parallel Databases*, 14(3):5–51, July 2003.
24. Paolo Bouquet, Chiara Ghidini, Fausto Giunchiglia, and Enrico Blanzieri. Theories and uses of context in knowledge representation and reasoning. *Journal of Pragmatics*, 35(3):455–484, 2003.
25. Debbie Richards. Combining cases and rules to provide contextualised knowledge based systems. In *Modeling and Using Context, Third International and Interdisciplinary Conference, CONTEXT 2001*, volume 2116 of *Lecture Notes in Artifical Intelligence*, pages 465–469, Dundee, UK, July 2001. Springer-Verlag, Berlin.
26. P. Compton and B. Jansen. Knowledge in context: A strategy for expert system maintenance. In J.Siekmann, editor, *Proceedings of the 2nd Australian Joint Artificial Intelligence Conference*, volume 406 of *Lecture Notes in Artificial Intelligence*, pages 292–306, Adelaide, Australia, November 1988. Springer-Verlag.
27. Tobias Scheffer. Algebraic foundation and improved methods of induction of ripple down rules. In *Procceedings of the Pacific Rim Workshop on Knowledge Acquisition*, pages 279–292, Sydney, Australia, 1996.
28. B. Drake and G. Beydoun. Predicate logic-based incremental knowledge acquisition. In P. Compton, A. Hoffmann, H. Motoda, and T. Yamaguchi, editors, *Proceedings of the sixth Pacific International Knowledge Acquisition Workshop*, pages 71–88, Sydney, December 2000.
29. Byeong Ho Kang, Phil Preston, and Paul Compton. Simulated expert evaluation of multiple classification ripple down rules. In *Proceedings of the 11th Workshop on Knowledge Acquisition, Modeling and Management*, Banff, Alberta, Canada, April 1998.
30. Dimitrios Georgakopoulos, Mark Hornick, and Amit Sheth. An overview of workflow management: From process modelling to workflow automation infrastructure. In *Distributed and Parallel Databases*, volume 3, pages 119–153. Kluwer Academic Publishers, Boston, 1995.
31. Clemens Hensinger, Manfred Reichert, Thomas Bauer, Thomas Strzeletz, and Peter Dadam. ADEPT$_{workflow}$ - advanced workflow technology for the efficient support of adaptive, enterprise-wide processes. In *Conference on Extending Database Technology*, pages 29–30, Konstanz, Germany, March 2000.

32. G. Vossen and M. Weske. The WASA approach to workflow management for scientific applications. In A. Dogac, L. Kalinichenko, M.T. Ozsu, and A. Sheth, editors, *Workflow Management Systems and Interoperability*, volume 164 of *ASI NATO Series, Series F: Computer and Systems Sciences*, pages 145–164. Springer, 1999.
33. Robert Muller, Ulrike Greiner, and Erhard Rahm. AgentWork: a workflow system supporting rule-based workflow adaptation. *Data & Knowledge Engineering*, 51(2):223–256, November 2004.
34. Michael zur Muehlen. *Workflow-based Process Controlling. Foundation, Design, and Implementation of Workflow-driven Process Information Systems*, volume 6 of *Advances in Information Systems and Management Science*. Logos, Berlin, 2004.
35. S. Dustdar. Caramba - a process-aware collaboration system supporting ad hoc and collaborative processes in virtual teams. *Distributed and Parallel Databases*, 15(1):45–66, 2004.
36. Michael Georgeff and Jon Pyke. Dynamic process orchestration. White paper, Staffware PLC `http://is.tm.tue.nl/bpm2003/download/WP%20Dynamic%20Proce%ss%20Orchestration%20v1.pdf` , March 2003.
37. Claus Hagen and Gustavo Alonso. Flexible exception handling in process support systems. Technical report no. 290, ETH Zurich, 1998.

Change Mining in Adaptive Process Management Systems

Christian W. Günther[1], Stefanie Rinderle[2], Manfred Reichert[3], and Wil van der Aalst[1]

[1] Eindhoven University of Technology, The Netherlands
{c.w.gunther, w.m.p.v.d.aalst}@tm.tue.nl
[2] University of Ulm, Germany
stefanie.rinderle@uni-ulm.de
[3] University of Twente, The Netherlands
m.u.reichert@ewi.utwente.nl

Abstract. The wide-spread adoption of process-aware information systems has resulted in a bulk of computerized information about real-world processes. This data can be utilized for process performance analysis as well as for process improvement. In this context process mining offers promising perspectives. So far, existing mining techniques have been applied to operational processes, i.e., knowledge is extracted from execution logs (process discovery), or execution logs are compared with some a-priori process model (conformance checking). However, execution logs only constitute one kind of data gathered during process enactment. In particular, adaptive processes provide additional information about process changes (e.g., ad-hoc changes of single process instances) which can be used to enable organizational learning. In this paper we present an approach for mining change logs in adaptive process management systems. The change process discovered through process mining provides an aggregated overview of all changes that happened so far. This, in turn, can serve as basis for all kinds of process improvement actions, e.g., it may trigger process redesign or better control mechanisms.

1 Introduction

The striking divergence between modeled processes and practice is largely due to the rigid, inflexible nature of commonplace *Process-Aware Information Systems* (PAISs) [10]. Whenever a small detail is modeled in the wrong manner, or external changes are imposed on the process (e.g. a new legislation or company guideline), users are forced to *deviate* from the prescribed process model. However, given the fact that process (re-)design is an expensive and time-consuming task, this results in employees working "behind the system's back". In the end, the PAIS starts to become a burden rather than the help it was intended to be. In recent years many efforts have been undertaken to deal with these drawbacks and to make PAISs more flexible. In particular, several approaches for *adaptive* process management have emerged (for an overview see [17]). Adaptive processes

R. Meersman, Z. Tari et al. (Eds.): OTM 2006, LNCS 4275, pp. 309–326, 2006.

enable users to *evolve* process definitions, such that they fit to changed situations. Adaptability can be supported by dynamic changes of different process aspects (e.g., control and data flow) at different levels (e.g., instance and type level). For example, ad-hoc changes conducted at the instance level (e.g., to add or delete process steps) allow to flexibly adapt single process instances to exceptional or changing situations [14]. Usually, such deviations are recorded in change logs (see [18]), which results in more meaningful log information when compared to traditional *Process Management Systems* (PMSs).

Adaptive PMSs like ADEPT or WASA offer flexibility at both process type level and process instance level [14,17,21]. So far, adaptive PMSs have not systematically addressed the fundamental question what we can learn from this additional information and how we can derive optimized process models from it. Process mining techniques [2], in turn, offer promising perspectives for learning from changes, but have focused on the analysis of pure execution logs (i.e., taking a behavioral and operational perspective) so far.

This paper presents a framework for integrating adaptive process management and process mining: Change information gathered within the adaptive PMS is exploited by process mining techniques. The results can be used to learn from previously applied changes and to optimize running and future processes accordingly. For this integration, first of all, we determine which runtime information about ad-hoc deviations is necessary and how it should be represented in order to achieve optimal mining results. Secondly, we develop new mining techniques based on existing ones which utilize change logs in addition to execution logs. As a result we obtain an abstract *change process* which reflects all changes applied to the instances of a particular process type so far. More precisely, a change process comprises *change operations* (as activities) and the causal relations between them. We further utilize information about the semantics of change operations (e.g., commutativity) in order to optimize our mining results. The resulting change process provides valuable knowledge about the process changes happened so far, which may serve as basis for deriving process optimizations in the sequel. Finally, the approach is implemented within a prototype integrating process mining framework ProM and ADEPT.

Sect. 2 introduces our framework for integrating process mining and adaptive process management. Sect. 3 describes how we import change log information into this framework and how changes are represented. Sect. 4 deals with our approach for discovering change processes from these logs. In Sect. 5 we discuss details of our implementation and show which tool supported is provided. Sect. 6 discusses related work and Sect. 7 concludes with a summary and an outlook.

2 Process Optimization by Integrating Process Mining and Adaptive Process Management

In this section we argue that the value of adaptive PMSs can be further leveraged by integrating them with process mining techniques. After introducing basics related to process mining, we present our overall integration framework.

2.1 Process Mining

Process-Aware Information Systems (PAISs), such as WfMS, ERP, and B2B systems, need to be configured based on *process models*. The latter specify the order in which process steps are to be executed and therefore enable the information system to ensure and control the correct execution of operational processes.

Usually, relevant events occurring in a PAIS (e.g., regarding the execution of tasks or the modification of data) are recorded in *event logs*. *Process mining* describes a family of *a-posteriori* analysis techniques exploiting the information recorded in these logs. Typically, respective approaches assume that it is possible to sequentially record events such that each event refers to an activity (i.e., a well-defined step in the process) and is related to a particular case (i.e., a process instance). Furthermore, there are other mining techniques making use of additional information such as the performer or originator of the event (i.e., the person / resource executing or initiating the activity), the timestamp of the event, or data elements recorded with the event (e.g., the size of an order).

Process mining addresses the problem that most "process owners" have very limited information about what is actually happening in their organization. In practice there is often a significant gap between what is prescribed or supposed to happen, and what *actually* happens. Only a concise assessment of the organizational reality, which process mining strives to deliver, can help in verifying process models, and ultimately be used in a process redesign effort.

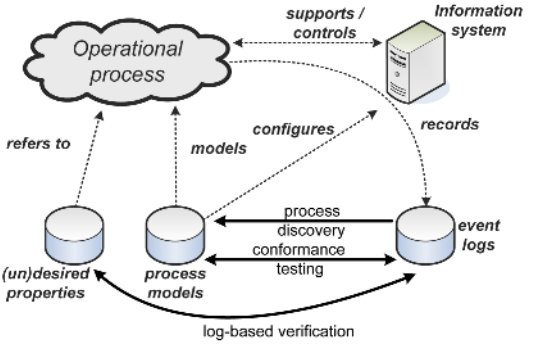


Fig. 1. Process Mining and its relation to BPM

There are three major classes of process mining techniques as indicated in Fig. 1. Traditionally, process mining has focused on *process discovery*, i.e. deriving information about the original process model, the organizational context, and execution properties from enactment logs. An example of a technique addressing the control flow perspective is the alpha algorithm [2], which can construct a Petri net model [6] describing the behavior observed in the event log. The multi-phase mining approach [7] can be used to construct an Event-driven Process Chain (EPC) based on similar information. Finally, first work regarding the mining of other model perspectives (e.g., organizational aspects [1]) and data-driven process support systems (e.g., case handling systems) has been done.

Another line of process mining research is *conformance testing*. Its aim is to analyze and measure discrepancies between the model of a process and its actual execution (as recorded in event logs). This can be used to indicate problems. Finally, *log-based verification* does not analyze enactment logs with respect to the original model, but rather checks the log for conformance with certain desired or undesired properties, e.g., expressed in terms of Linear Temporal Logic (LTL) formulas. This makes it an excellent tool to check a case for conformance to certain laws or corporate guidelines (e.g. the four-eyes principle).

At this point in time there are mature tools such as the ProM framework, featuring an extensive set of analysis techniques which can be applied to real process enactments while covering the whole spectrum depicted in Fig. 1 [9].

2.2 Integration Framework

Both process mining and adaptive workflow address fundamental issues that are widely prevalent in the current practice of BPM implementations. These problems stem from the fact that the *design*, *enactment*, and *analysis* of a business process are commonly interpreted, and implemented, as *detached phases*.

Although both techniques are valuable on their own, we argue that their full potential can be only harnessed by tight integration. While process mining can deliver concrete and reliable information about how process models need to be changed, adaptive PMSs provide the tools to safely and conveniently implement these changes. Thus, we propose the development of process mining techniques, integrated into adaptive PMSs as a *feedback cycle*. In the sequel, adaptive PMSs need to be equipped with functionality to exploit this feedback information.

The framework depicted in Fig. 2 illustrates how such an integration could look like. Adaptive PMSs, visualized in the upper part of this model, operate on pre-defined process models. The evolution of these models over time spawns a set of process changes, i.e., results in multiple *process variants*. Like in every PAIS, *enactment logs* are created which record the sequence of activities executed for each case. On top of that, adaptive PMSs additionally log the sequence of change operations imposed on a process model for every executed case, producing a set of *change logs*. Process mining techniques that integrate into such system, in form of a feedback cycle, fall into one of three major categories:

Change analysis: Process mining techniques from this category make use of change log information, besides the original process models and their variants. Their goal is to determine common and popular variants for each process model, which may be promoted to replace the original model. Possible ways to pursue this goal are through statistical analysis of changes or their abstraction to higher-level models.

Integrated analysis: This analysis uses both change and enactment logs in a combined fashion. Possible applications in this category could perform a context-aware categorization of changes as follows. After clustering change sequences, as found in the change logs, into groups, the incentive for these changes can be derived. This is performed by inspecting the state of each case, i.e. the values of case data objects, at the time of change, as known from

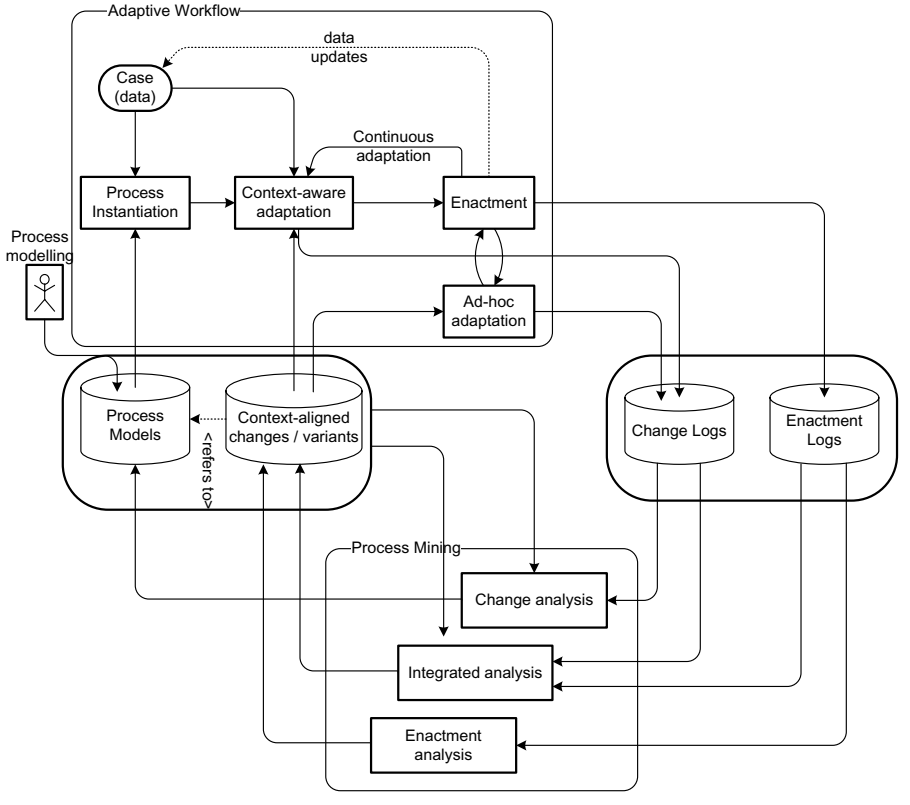


Fig. 2. Integration of Process Mining and Adaptive Process Management

the original process model and the enactment logs. A decision-tree analysis of these change clusters provides an excellent basis for guiding users in future process adaptations, based on the peculiarities of their specific case.

Enactment analysis: Based solely on the inspection of enactment logs, techniques in this category can pinpoint parts of a process model which need to be changed. For example, when a specific alternative of a process model has never been executed, the original process model may be simplified by removing that part. Further techniques may also clean the model repository from rarely used process definitions.

These examples give only directions in which the development of suitable process mining techniques may proceed. Of course, their concrete realization depends on the nature of the system at hand. For example, it may be preferable to present highlighted process models to a specialist before their deletion or change, rather than having the system performing these tasks autonomously.

When such feedback cycle is designed and implemented consistently, the resulting system is able to provide user guidance and autonomous administration to an unprecedented degree. Moreover, the tight integration of adaptive PMSs

and process mining technologies provides a powerful foundation, on which a new generation of truly intelligent and increasingly autonomous PAISs can be built.

3 Change Logs

Adaptive PMSs do not only create process enactment logs, but *they also log the sequence of changes applied to a process model*. This section introduces the basics of these change logs. We first discuss the nature of changes and then introduce MXML as general format for event logs. Based on this we show how change logs can be mapped onto the MXML format. MXML-based log files constitute the basic input for the mining approach described in Sect. 4.

3.1 General Change Framework

Logically, a process change is accomplished by applying a sequence of change operations to the respective process model [14]. The question is how to represent this change information within change logs. In principle, the information to be logged can be represented in different ways. The goal must be to find an adequate representation and appropriate analysis techniques to support the three cases described in the previous section.

Independent from the applied (high–level) change operations (e.g., adding, deleting or moving activities), for example, we could translate the change into a set of basic change primitives (i.e., basic graph primitives like `addNode` or `deleteEdge`). This still would enable us to restore process structures, but also result in a loss of information about change semantics and therefore limit traceability and change analysis. As an alternative we can explicitly store the applied high–level change operations, which combine basic primitives in a certain way.

High–level change operations are based on formal pre-/post-conditions. This enables the PMS to guarantee model correctness when changes are applied. Further, high-level change operations can be combined to *change transactions*. This becomes necessary, for example, if the application of a high-level change operation leads to an incorrect process model and this can be overcome by conducting concomitant changes. During runtime several change transactions may be applied to a particular process instance. All change transactions related to a process instance are stored in the *change log*[1] of this instance (cf. [18]).

In the following we represent change log entries by means of high-level change operations since we want to exploit their semantical content (see Fig. 3 for an example). However, basically, the mining approach introduced later can be adapted to change primitives as well. Table 1 presents examples of *high-level change operations* on process models which can be used at the process type as well as at the process instance level to create or modify models. Although the

[1] A change log is an ordered series $cL{:=}< \Delta_1, \ldots, \Delta_n >$ of change operations Δ_i (i = 1, ..n); i.e., when applying the change operations contained in cL to a correct process schema S, all intermediate process schemas S_i with $S_i := S_{i-1} + \Delta_i$ (i = 1,. . . , n; $S_0 := S$) are correct process schemas.

change operations are exemplarily defined on the ADEPT meta model (see [14] for details) they are generic in the sense that they can be easily transferred to other meta models as well (e.g. [15]).

Table 1. Examples of High-Level Change Operations on Process Schemas

Change Operation Δ **Applied to S**	opType	subject	paramList
insert(S, X, A, B, [*sc*])	insert	X	S, A, B
Effects on S: inserts activity X between node sets A and B (it is a conditional insert if *sc* is specified)			
Preconditions: node sets A and B must exist in S, and X must not be contained in S yet (i.e., no duplicate activities!)			
delete(S, X)	delete	X	S
Effects on S: deletes activity X from S			
Preconditions: activity X must be contained exactly once in S			
move(S, X, A, B, [*sc*])	move	X	S, A, B
Effects on S: moves activity X from its original position between node sets A and B (it is a conditional insert if *sc* is specified)			
Preconditions: activity X and node sets A and B must be contained exactly once in S			

3.2 The MXML Format for Process Event Logs

MXML is an XML-based format for representing and storing event log data, which is supported by the largest subset of process mining tools, such as ProM. While focusing on the core information needed for process mining, the format reserves generic fields for extra information potentially provided by a PAIS. Due to its outstanding tool support and extensibility, the MXML format has been selected for storing change log information in our approach.

The root node of a MXML document is a *WorkflowLog*. It represents a log file, i.e. a logical collection of events having been derived from one system. Every workflow log can potentially contain one *Source* element, which is used to describe that system the log has been imported from. Apart from the source descriptor, a workflow log can contain an arbitrary number of *Processes* as child elements, each grouping events that occurred during the execution of a specific process definition. The single executions of a process are represented by child elements of type *ProcessInstance*, each representing one case in the system.

Finally, process instances group an arbitrary number of *AuditTrailEntry* elements as child elements. Each of these child elements refers to one specific event which has occurred in the system. Every audit trail entry must contain at least two child elements: The *WorkflowModelElement* describes the abstract process definition element to which the event refers, e.g. the name of the activity that was executed. The second mandatory element is the *EventType*, describing the nature of the event, e.g. whether a task was scheduled, completed, etc. The optional child elements of an audit trail entry are *Timestamp* and *Originator*. The timestamp holds the date and time of when the event has occurred, while the originator identifies the resource, e.g. person, which has triggered the event.

To enable the flexible extension of this format with extra information, all mentioned elements (except the child elements of *AuditTrailEntry*) can also have a

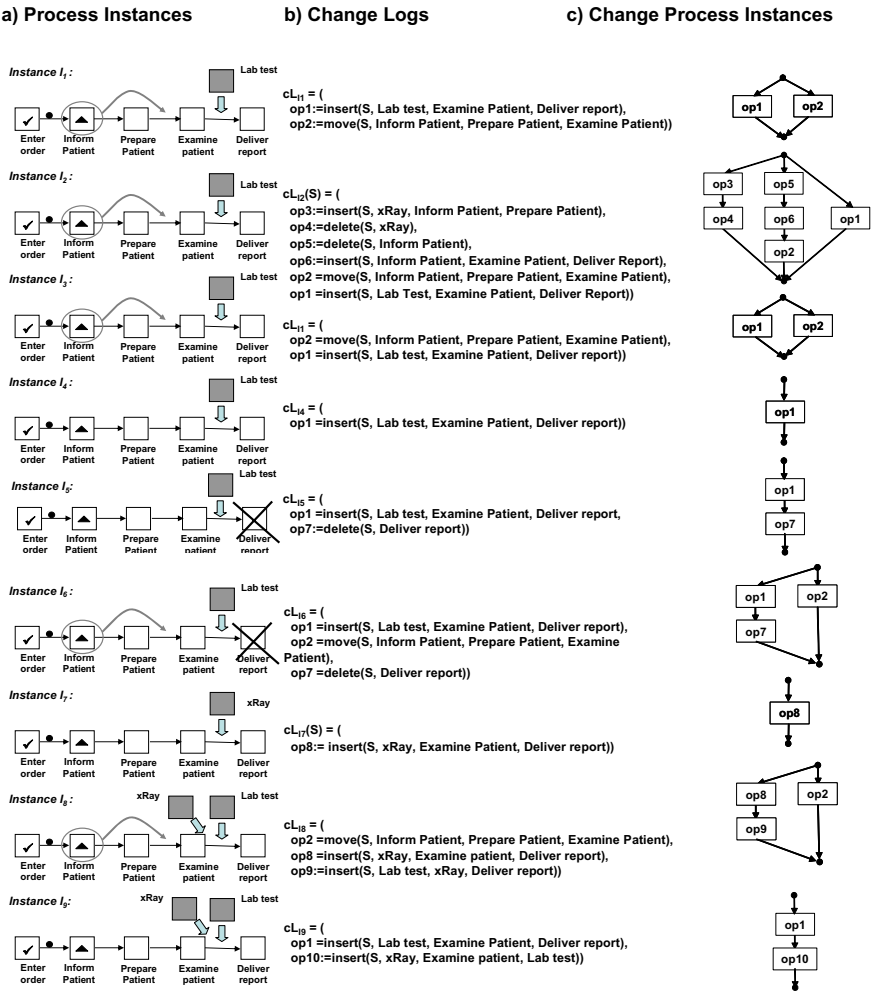


Fig. 3. Modified Process Instances and Associated Change Logs

generic *Data* child element. The data element groups an arbitrary number of *Attributes*, which are key-value pairs of strings. The following subsection describes the mapping of change log information to MXML, which is heavily based on using custom attributes of this sort.

3.3 Mapping Change Log Information to MXML

With respect to an adaptive PAIS, change log information can be structured on a number of different levels. Most of all, change events can be grouped by the process definition they address. As we are focusing on changes applied to *cases*,

i.e. executed instances of a process definition, the change events referring to one process can be further subdivided with respect to the specific case in which they were applied. Finally, groups of change events on a case level are naturally sorted by the order of their occurrence.

The described structure of change logs fits well into the common organization of enactment logs, with instance traces then corresponding to *consecutive changes* of a process model, in contrast to its execution. Thus, change logs can be mapped to the MXML format with minor modifications. Listing 1 shows an MXML audit trail entry describing the insertion of a task "Lab Test" into a process definition, as e.g. seen for Instance I_1 in Fig. 3.

```
<AuditTrailEntry>
    <Data>
        <Attribute name="CHANGE.postset">Deliver_report</Attribute>
        <Attribute name="CHANGE.type">INSERT</Attribute>
        <Attribute name="CHANGE.subject">Lab_test</Attribute>
        <Attribute name="CHANGE.rationale">Ensure that blood values
             are within specs.</Attribute>
        <Attribute name="CHANGE.preset">Examine_patient</Attribute>
    </Data>
    <WorkflowModelElement>INSERT.Lab_test</WorkflowModelElement>
    <EventType>complete</EventType>
    <Originator>N.E.Body</Originator>
</AuditTrailEntry>
```

Listing 1. Example of a change event in MXML

Change operations are characterized by the *type* (e.g., "INSERT") of change, the *subject* which has been primarily affected (e.g., the inserted task), and the *syntactical context* of the change. This syntactical context contains the change operation's *pre-* and *post-set*, referring to adjacent model elements that are either directly preceding or following the change subject in the process definition. These specific change operation properties are not covered by the MXML format, therefore they are stored as attributes in the "Data" field. The set of attributes for a change event is further extended by an optional *rationale* field, storing a human-readable reason, or incentive, for this particular change operation.

The *originator* field is used for the person having applied the respective change, while the *timestamp* field obviously describes the concise date and time of occurrence. Change events have the *event type* "complete" by default, because they can be interpreted as atomic operations. In order to retain backward compatibility of MXML change logs with traditional process mining algorithms, the *workflow model element* needs to be specified for each change event. As the change process does not follow a prescribed process model, this information is not available. Thus, a concatenation of change type and subject is used for the workflow model element field.

On top of having a different set of information, change logs also exhibit specific properties making them different from enactment logs. The next section investigates these specific properties and uses them for a first mining approach.

4 Mining Compact Change Processes

In this section we describe an approach for analyzing change log information, as found in adaptive PMSs. First, we explore the nature of change logs in more detail. This is followed by an introduction to the concept of commutativity between change operations in Sect. 4.2. This commutativity relation provides the foundation for our mining algorithm for change processes, as introduced in Sect. 4.3.

4.1 A Characterization of Change Logs

Change logs, in contrast to regular enactment logs, do not describe the execution of a defined process. This is obvious from the fact that, if the set of potential changes would have been known in advance, then these changes could have already been incorporated in the process model (making dynamic change obsolete). Thus, change logs must be interpreted as emerging sequences of activities which are taken from a set of change operations.

In Sect. 3.3 it has been defined that each change operation refers to the original process model through three associations: the *subject*, *pre-*, and *post-set* of the change. As all three associations can theoretically be bound to any subset from the original process model's set of activities[2], the set of possible change operations grows exponentially with the number of activities in the original process model. This situation is fairly different from mining a regular process model, where the number of activities is usually rather limited (e.g., up to 50–100 activities). Hence, change process mining poses an interesting challenge.

Summarizing the above characteristics, we can describe the *meta-process* of changing a process schema as a *highly unstructured* process, potentially involving a *large number of distinct activities*. These properties, when faced by a process mining algorithm, typically lead to overly precise and confusing *"spaghetti-like" models*. For a more compact representation of change processes, it is helpful to abstract from a certain subset of order relations between change operations.

When performing process mining on enactment logs (i.e., the classical application domain of process mining), the actual state of the mined process is treated like a "black box". This is a result of the nature of enactment logs, which typically only indicate transitions in the process, i.e. the execution of activities. However, the information contained in change logs allows to trace the state of the change process, which is indeed defined by the process schema that is subject to change. Moreover, one can compare the effects of different (sequences of) change operations. From that, it becomes possible to explicitly detect whether two consecutive change operations might as well have been executed in the reverse order, without changing the resulting state.

The next section introduces the concept of *commutativity* between change operations, which is used to reduce the number of ordering relations by taking into account the semantic implications of change events. Since the order of commutative change operations does not matter, we can abstract from the actually observed sequences thus simplifying the resulting model.

[2] Here we assume that the subset describing the *subject* field is limited to one.

4.2 Commutative and Dependent Change Operations

Change operations modify a process schema, either by altering the set of activities or by changing their ordering relations. Thus, each application of a change operation to a process schema results in another, different schema. A process schema can be described formally without selecting a particular notation, i.e., we abstract from the concrete operators of the process modeling language and only describe the set of activities and possible behavior.

Definition 1 (Process Schema). *A process schema is a tuple* $PS = (A, TS)$ *where*

- A *is a set of activities*
- $TS = (S, T, s_{start}, s_{end})$ *is a labeled transition system, where* S *is the set of reachable states,* $T \subseteq S \times (A \cup \{\tau\}) \times S$ *is the transition relation,* $s_{start} \in S$ *is the initial state, and* $s_{end} \in S$ *is the final state.*

$\mathcal{P}$ *is the set of all process schemas.*

The behavior of a process is described in terms of a transition system TS with some initial state s_{start} and some final state s_{end}. Note that any process modeling language can be mapped onto a labeled transition system. The transition system does not only define the set of possible traces (i.e., execution orders); it also captures the moment of choice. Moreover, it allows for "silent steps". A silent step, denoted by τ, is an activity within the system which changes the state of the process, but is not observable in the execution logs. This way we can distinguish between different types of choices (internal/external or controllable/uncontrollable) [5]. While all change operations modify the set of states S and the transition relation T, the "move" operation is the only one not changing the set of activities A.

In order to compare sequences of change operations, and to derive ordering relations between these changes, it is helpful to define an equivalence relation for process schemas.

Definition 2 (Equivalent Process Schemas). *Let* $\equiv$ *be some equivalence relation. For* $PS_1, PS_2 \in \mathcal{P} : PS_1 \equiv PS_2$ *if and only if* PS_1 *and* PS_2 *are considered to be equivalent.*

There exist many notions of process equivalence. The weakest notion of equivalence is *trace equivalence* [11,13,17], which regards two process models as equivalent if the sets of observable traces they can execute are identical. Since the number of traces a process model can generate may be infinite, such comparison may be complicated. Moreover, since trace equivalence is limited to comparing traces, it fails to correctly capture the moment at which choice occurs in a process. For example, two process schemas may generate the same set of two traces $\{ABC, ABD\}$. However, the process may be very different with respect to the moment of choice, i.e. the first process may already have a choice after A to execute either BC or BD, while the second process has a choice between

C and D just after B. *Branching bisimilarity* is one example of an equivalence, which can correctly capture this moment of choice. For a comparison of branching bisimilarity and further equivalences the reader is referred to [12]. In the context of this paper, we abstract from a concrete notion of equivalence, as the approach described can be combined with different process modeling notations and different notions of equivalence.

As stated above, each application of a change operation transforms a process schema into another process schema. This can be formalized as follows:

Definition 3 (Change in Process Schemas). *Let $PS_1, PS_2 \in \mathcal{P}$ be two process schemas and let Δ be a process change.*

- *$PS_1[\Delta\rangle$ if and only if Δ is applicable to PS_1, i.e., Δ is possible in PS_1.*
- *$PS_1[\Delta\rangle PS_2$ if and only if Δ is applicable to PS_1 (i.e., $PS_1[\Delta\rangle$) and PS_2 is the process schema resulting from the application of Δ to PS_1.*

The applicability of a change operation to a specific process schema is defined in Table 1, and is largely dictated by common sense. For example, an activity X can only be inserted into a schema S, between the node sets A and B, if these node sets are indeed contained in S and the activity X is not already contained in S. Note that we do not allow duplicate tasks, i.e. an activity can be contained only once in a process schema.

Based on the notion of process equivalence we can now define the concept of *commutativity* between change operations.

Definition 4 (Commutativity of Changes). *Let $PS \in \mathcal{P}$ be a process schema, and let Δ_1 and Δ_2 be two process changes. Δ_1 and Δ_2 are commutative in PS if and only if:*

- *There exist $PS_1, PS_2 \in \mathcal{P}$ such that $PS[\Delta_1\rangle PS_1$ and $PS_1[\Delta_2\rangle PS_2$,*
- *There exist $PS_3, PS_4 \in \mathcal{P}$ such that $PS[\Delta_2\rangle PS_3$ and $PS_3[\Delta_1\rangle PS_4$,*
- *$PS_2 \equiv PS_4$.*

Two change operations are commutative, if they have exactly the same effect on a process schema, regardless of the order in which they are applied. If two change operations are not commutative, we regard them as *dependent*, i.e., the effect of the second change depends on the first one. The concept of commutativity captures the ordering relation between two consecutive change operations. If two change operations are commutative according to Def. 4 they can be applied in any given order, therefore there exists no ordering relation between them.

In the next subsection we demonstrate that existing process mining algorithms can be enhanced with the concept of commutativity, thereby abstracting from ordering relations that are irrelevant from a semantical point of view (i.e., their order does not influence the resulting process schema).

4.3 Mining Change Processes

Mining change processes is to a large degree identical to mining regular processes from enactment logs. Therefore, we have chosen not to develop an entirely new

algorithm, but rather to base our approach on an existing process mining technique. Among the available algorithms, the *multi-phase* algorithm [7,8] has been selected, which is very robust in handling fuzzy branching situations (i.e., it can employ the "OR" semantics to split and join nodes, in cases where neither "AND" nor "XOR" are suitable). Although we illustrate our approach using a particular algorithm, it is important to note that any process mining algorithm based on explicitly detecting causalities can be extended in this way (e.g., also the different variants of the α-algorithm).

The multi-phase mining algorithm is able to construct basic workflow graphs, Petri nets, and EPC models from the causality relations derived from the log. For an in-depth description of this algorithm, the reader is referred to [7,8]. The basic idea of the multi-phase mining algorithm is to discover the process schema in two steps. First a model is generated for each individual process instance. Since there are no choices in a single instance, the model only needs to capture causal dependencies. Using causality relations derived from observed execution orders and the commutativity of specific change operations, it is relatively easy to construct such instance models. In the second step these instance models are aggregated to obtain an overall model for the entire set of change logs.

The causal relations for the multi-phase algorithm [7,8] are derived from the change log as follows. If a change operation A is followed by another change B in at least one process instance, and no instance contains B followed by A, the algorithm assumes a possible causal relation from A to B (i.e., "A may cause B"). In the example log introduced in Fig. 3, instance I_2 features a change operation deleting "Inform Patient" followed by another change, inserting the same activity again. As no other instance contains these changes in reverse order, a causal relation is established between them.

Fig. 4 shows a Petri net model [6] of the change process mined from the example change log instances in Fig. 3. The detected causal relation between deleting and inserting "Inform patient" is shown as a directed link between these activities. Note that in order to give the change process explicit start and end points, artificial activities have been added. Although the model contains only seven activities, up to three of them can be executed concurrently. Note further that the process is very flexible, i.e. all activities can potentially be skipped. From the very small data basis given in Fig. 3, where change log instances hardly have common subsequences, this model delivers a high degree of abstraction.

If two change operations are found to appear in both orders in the log, it is assumed that they can be executed in any order, i.e. concurrently. (Note that there might be some order between concurrent changes determined by context factors not directly accessible to the system. We aim at integrating such information in our future work). An example for this is inserting "xRay" and inserting "Lab Test", which appear in this order in instance I_8, and in reverse order in instance I_9. As a result, there is no causal relation, and thus no direct link between these change operations in the model shown in Fig. 4.

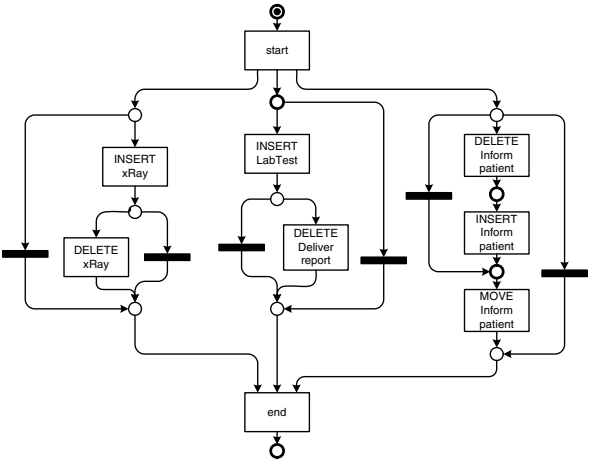


Fig. 4. Mined Example Process (Petri net notation)

Apart from observed concurrency, as described above, we can introduce the concept of *commutativity-induced concurrency*, using the notion of commutativity introduced in the previous subsection (cf. Definition 4). From the set of observed causal relations, we can exclude causal relations between change operations that are commutative. For example, instance I_2 features deleting activity "xRay" directly followed by deleting "Inform Patient". As no other process instance contains these change operations in reverse order, a regular process mining algorithm would establish a causal relation between them.

However, it is obvious that it makes no difference in which order two activities are removed from a process schema. As the resulting process schemas are identical, these two changes are *commutative*. Thus, we can safely discard a causal relation between deleting "xRay" and deleting "Inform Patient", which is why there is no link in the resulting change process shown in Fig. 4.

Commutativity-induced concurrency removes unnecessary causal relations, i.e. those causal relations that do not reflect actual dependencies between change operations. Extending the multi-phase mining algorithm with this concept significantly improves the clarity and quality of the mined change process. If it were not for commutativity-induced concurrency, every two change operations would need to be observed in both orders to find them concurrent. This is especially significant in the context of change logs, since one can expect changes to a process schema to happen far less frequently than the actual execution of the schema, resulting in less log data.

5 Implementation and Tool Support

To enable experimentation with change logs and their analysis, an import plug-in has been implemented for the ProM*import* framework, which allows to extract both enactment and change logs from instance files of the ADEPT demonstrator

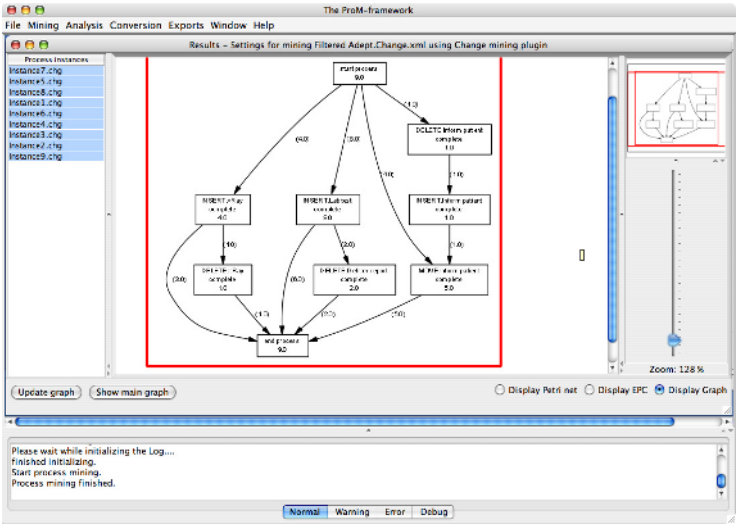


Fig. 5. Change Mining Plug-in within ProM

prototype [16]. ProM*import*[3] is a flexible and open framework for the rapid prototyping of MXML import facilities from all kinds of PAISs. The ADEPT demonstrator prototype provides the full set of process change facilities found in the ADEPT distribution, except for implementation features like work distribution and the like. The combination of both makes it possible to create and modify a process model in the ADEPT demonstrator prototype, after which a respective change log can be imported and written to the MXML-based change log format described in Sect. 3.3.

These change logs can then be loaded into the ProM framework[4]. A dedicated change mining plug-in has been developed, which implements the commutativity-enhanced multi-phase algorithm described in Sect. 4.3. It is also possible to mine only a selection of the change logs found in the log. The resulting change process can be visualized in the form of a workflow graph, Petri net, or EPC.

Figure 5 shows the change mining plug-in within the ProM framework, displaying the example process introduced in Fig. 3 in terms of a process graph. The activities and arcs are annotated with frequencies, indicating how often the respective node or path has been found in the log.

6 Related Work

Although process mining techniques have been intensively studied in recent years [2,3,4,7,8], no systematic research on analyzing process change logs has

[3] ProM*import* is available under an Open Source license at http://promimport.sourceforge.net/.

[4] ProM is available under an Open Source license at http://prom.sourceforge.net/.

been conducted so far. Existing approaches mainly deal with the discovery of process models from execution logs, conformance testing, and log-based verification (cf. Sect. 2.1). However, execution logs in traditional PMSs only reflect what has been modeled before, but do not capture information about process changes. While earlier work on process mining has mainly focused on issues related to control flow mining, recent work additionally uses event-based data for mining model perspectives other than control flow (e.g., social networks [1], actor assignments, and decision mining [19]).

In recent years, several approaches for adaptive process management have emerged [17], most of them supporting changes of certain process aspects and changes at different levels. Examples of adaptive PMSs include ADEPT [16], CBRflow [20], and WASA [21]. Though these PMSs provide more meaningful process logs when compared to traditional workflow systems, so far, only little work has been done on fundamental questions like what we can learn from this additional log information, how we can utilize change logs, and how we can derive optimized process models from them. CBRflow has focused on the question how to facilitate exception handling in adaptive PMSs. In this context case-based reasoning (CBR) techniques have been adopted in order to capture contextual knowledge about ad-hoc changes in change logs, and to assist actors in reusing previous changes. [20]. This complementary approach results in semantically enriched log-files (e.g., containing information about the frequency of a particular change, user ratings, etc.) which can be helpful for our future work.

7 Summary and Outlook

In this paper we presented an approach for integrating adaptive process management and process mining in order to exploit knowledge about process changes from change logs. For this we have developed a mining technique and implemented it as plug-in of the ProM framework taking ADEPT change logs as input. We demonstrated that change log information (as created by adaptive PMSs like ADEPT) can be imported into the ProM framework. Based on this we have sketched how to discover a (minimal) change process which captures all modifications applied to a particular process instance so far. This discovery is based on the analysis of the (temporal) dependencies existing between the change operations applied to the respective process instance. How single change processes can be combined to one aggregated change process (capturing all instance changes applied) has been presented afterwards. Finally we have described the implementation framework behind our approach. Altogether, the presented approach can be very helpful for process engineers to get an overview about which instance changes have been applied at the system level and what we can learn from them. Corresponding knowledge is indispensable to make the right decisions with respect to the introduction of changes at the process type level (e.g., to reduce the need for ad-hoc changes at the instance level in future).

In our future work we want to further improve user support by augmenting change processes with additional contextual information (e.g., about the reason why changes have been applied or the originator of the change). From this

we expect better comprehensibility of change decisions and higher reusability of change knowledge (in similar situations). The detection of this more context-based information will be accomplished by applying advanced mining techniques (e.g., decision mining [19]) to change log information.

Acknowledgements. This research has been supported by the Technology Foundation STW, applied science division of NWO and the technology programme of the Dutch Ministry of Economic Affairs.

References

1. W.M.P. van der Aalst, H.A. Reijers, and M. Song. Discovering Social Networks from Event Logs. *Computer Supported Cooperative work*, 14(6):549–593, 2005.
2. W.M.P. van der Aalst, A.J.M.M. Weijters, and L. Maruster. Workflow Mining: Discovering Process Models from Event Logs. *IEEE Transactions on Knowledge and Data Engineering*, 16(9):1128–1142, 2004.
3. R. Agrawal, D. Gunopulos, and F. Leymann. Mining Process Models from Workflow Logs. In *Sixth International Conference on Extending Database Technology*, pages 469–483, 1998.
4. J.E. Cook and A.L. Wolf. Discovering Models of Software Processes from Event-Based Data. *ACM Transactions on Software Engineering and Methodology*, 7(3):215–249, 1998.
5. J. Dehnert and W.M.P. van der Aalst. Bridging the Gap Between Business Models and Workflow Specifications. *International Journal of Cooperative Information Systems*, 13(3):289–332, 2004.
6. J. Desel, W. Reisig, and G. Rozenberg, editors. *Lectures on Concurrency and Petri Nets*, volume 3098 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 2004.
7. B.F. van Dongen and W.M.P. van der Aalst. Multi-Phase Process Mining: Building Instance Graphs. In P. Atzeni, W. Chu, H. Lu, S. Zhou, and T.W. Ling, editors, *International Conference on Conceptual Modeling (ER 2004)*, volume 3288 of *Lecture Notes in Computer Science*, pages 362–376. Springer-Verlag, Berlin, 2004.
8. B.F. van Dongen and W.M.P. van der Aalst. Multi-Phase Process Mining: Aggregating Instance Graphs into EPCs and Petri Nets. In *Proceedings of the 2nd International Workshop on Applications of Petri Nets to Coordination, Worklflow and Business Process Management (PNCWB) at the ICATPN 2005*, 2005.
9. B.F. van Dongen, A.K. de Medeiros, H.M.W. Verbeek, A.J.M.M. Weijters, and W.M.P. van der Aalst. The ProM framework: A new era in process mining tool support. In G. Ciardo and P. Darondeau, editors, *Proceedings of the 26th International Conference on Applications and Theory of Petri Nets (ICATPN 2005)*, volume 3536 of *Lecture Notes in Computer Science*, pages 444–454. Springer-Verlag, Berlin, 2005.
10. M. Dumas, W.M.P. van der Aalst, and A.H.M. ter Hofstede. *Process-Aware Information Systems: Bridging People and Software through Process Technology*. Wiley & Sons, 2005.
11. R. van Glabbeek and U. Goltz. Refinement of Actions and Equivalence Notions for Concurrent Systems. *Acta Informatica*, 37(4–5):229–327, 2001.
12. R.J. van Glabbeek and W.P. Weijland. Branching Time and Abstraction in Bisimulation Semantics. *Journal of the ACM*, 43(3):555–600, 1996.

13. B. Kiepuszewski. *Expressiveness and Suitability of Languages for Control Flow Modelling in Workflows*. PhD thesis, Queensland University of Technology, Brisbane, 2002. (available via *http://www.workflowpatterns.com/*).
14. M. Reichert and P. Dadam. ADEPTflex - Supporting Dynamic Changes of Workflows Without Loosing Control. *Journal of Intelligent Information Systems*, 10(2):93–129, 1998.
15. M. Reichert, S. Rinderle, and P. Dadam. On the common support of workflow type and instance changes under correctness constraints. In *Proc. Int'l Conf. on Cooperative Information Systems (CoopIS'03)*, pages 407–425, Catania, 2003.
16. M. Reichert, S. Rinderle, U. Kreher, and P. Dadam. Adaptive process management with ADEPT2. In *Proc. 21st Int'l Conf. on Data Engineering (ICDE'05)*, pages 1113–1114, Tokyo, 2005.
17. S. Rinderle, M. Reichert, and P. Dadam. Correctness Criteria for Dynamic Changes in Workflow Systems – A Survey. *Data and Knowledge Engineering, Special Issue on Advances in Business Process Management*, 50(1):9–34, 2004.
18. S. Rinderle, M. Reichert, M. Jurisch, and U. Kreher. On Representing, Purging, and Utilizing Change Logs in Process Management Systems. In *Proc. Int'l Conf. on Business Process Management (BPM'06)*, Vienna, 2006.
19. A. Rozinat and W.M.P. van der Aalst. Decision mining in prom. In *Proc. Int'l Conf. on Business Process Management (BPM'06)*, Vienna, 2006.
20. B. Weber, S. Rinderle, W. Wild, and M. Reichert. CCBR-Driven Business Process Evolution. In *Proc. Int. Conf. on Cased based Reasoning (ICCBR'05)*, Chicago, 2005.
21. M. Weske. Formal foundation and conceptual design of dynamic adaptations in a workflow management system. In *Proc. Hawaii International Conference on System Sciences (HICSS-34)*, 2001.

A Link-Based Ranking Model for Services

Camelia Constantin[1], Bernd Amann[1], and David Gross-Amblard[2]

[1] LIP6, Univ.Paris 6, France
camelia.constantin@lip6.fr, bernd.amann@lip6.fr
[2] CEDRIC, CNAM, France
dgram@cnam.fr

Abstract. The number of services on the web is growing every day and finding useful and efficient ranking methods for services has become an important issue in modern web applications. In this paper we present a link-based importance model and efficient algorithms for distributed services collaborating through service calls. We adapt the PageRank algorithm and define a service importance that reflects its activity and its contribution to the quality of other services.

1 Introduction

The basic task of a ranking model is to define scores for ordering a set of entities according to some specific criteria. A large number of ranking models and algorithms have been proposed for various kinds of applications and information entities (documents, tuples, services) in different domains like document retrieval, Web search, service discovery and P2P query processing. Content-based ranking models classify entities according to the *relevance* of their contents or other associated metadata to a given input query. This kind of ranking has proven its efficiency in document retrieval systems and quality-based data and service selection. Link-based ranking exploits any kind of structural, semantic or navigational links between entities for estimating the *importance* of a service among all other services. The most prominent example for this kind of ranking is certainly Google's PageRank, which estimates the importance of a page in the Web graph for efficiently ranking Web search results. This article presents a new link-based ranking model and algorithms for distributed collaborating services.

We consider service-oriented applications relying on a set of services that collaborate for executing certain tasks. Our notion of service is abstract and should be understood as any kind of node, with some local behavior and data, which calls other nodes by exchanging messages. We do not put any restriction neither on the structure and contents of the exchanged messages nor on the functionality, local data and the implementation of each individual service. This means that our model applies to heterogeneous service infrastructures including, for instance, simple generic file sharing and document printing services, database query interfaces and Web services encapsulating some complex application-specific behavior.

R. Meersman, Z. Tari et al. (Eds.): OTM 2006, LNCS 4275, pp. 327–344, 2006.

Many kinds of applications fit our definition of a service and could benefit from our importance based ranking approach. For instance, in a distributed P2P search engine, each peer can be defined as a service searching some local documents and calling other peer services. Search results can obviously be ranked according to some content-based criteria like document/query relevance, but in some situations, it also might be useful to rank results according to the importance of their sources. Another example is standard web service discovery and selection. Whereas it is possible to compare and rank web services according to their WSDL and UDDI descriptions, these techniques are generally based on homogeneous and semantically rich service descriptions. We claim that link-based measures taking into account "collaboration links" between services are an interesting alternative for ranking heterogeneous services in large-scale service-oriented architectures (SOA).

We will use in the sequel an example of a distributed news syndication system, where each node can play the role of a news server (provider), a news consumer (client) or both (portal). Figure 2 shows an example of five collaborating news services s_1 to s_5. Services s_4 and s_5 are provided by the French news agency *Agence France Press (AFP)* and the U.S. agency *Reuters*. Services s_2 and s_3 correspond to two French journals providing daily news. Finally, s_1 corresponds to the *Google News* service. All services collaborate by exchanging news via service calls. This is illustrated in Figure 1 below showing a graph where each edge $s_i \hookrightarrow^t s_j$ corresponds to a service call of service s_i to service s_j at time instant t. We see for example that *AFP* has been called by service *Le Monde*

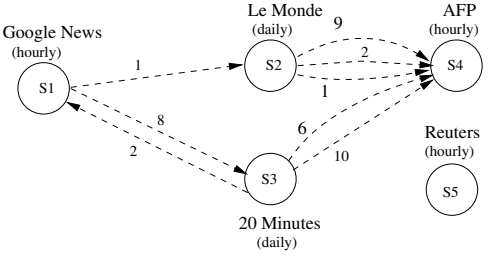


Fig. 1. A service call log

at three different instants 1, 2 and 9. In our model we assume that each such call $s_i \hookrightarrow^t s_j$ represents a contribution of the called service s_j to the calling service s_i and that this contribution also depends on timestamp t. For example, suppose that *Google News* is interested in all news published by *Le Monde* which updates its news every day. Then, the call-dependent contribution of *Le Monde* for *Google News* at some given moment τ could be estimated by the *age* of the last call received by *Le Monde* from *Google News* before τ. If this age is less than one day, *Le Monde* is estimated to be highly useful for *Google News* at moment τ. On the other hand, if the age of the last call exceeds a certain period, e.g. one month, all results received from *Le Monde* become useless for *Google News*.

The contribution of a service to some of its clients not only depends on the timestamps of the received service calls, but also on the way of how the service contributes to the client's quality compared to other services. If we assume that *Google News* considers the news obtained from *Le Monde* of high quality, the contribution of this service might be estimated higher than the contribution of *20 Minutes* which provides less relevant news. Our notion of "quality" is abstract and each individual service might use different quality features for estimating the contribution of other services. For example, the French newspaper *Le Monde* supposes that, in general, the French news agency *AFP* provides more interesting news for its readers than the U.S. agency *Reuters*. On the other hand, *20 Minutes* tries to reduce cost and therefore prefers *Google News* which is free of charge to *AFP*, which is not. This "quality contribution" is illustrated in the graph of Figure 2 where each edge $s_i \rightarrow^c s_j$ is labeled by the value $c \in [0, 1]$ representing the *contribution* of service s_j to the quality of its client s_i *relative to the other services* used by s_i.

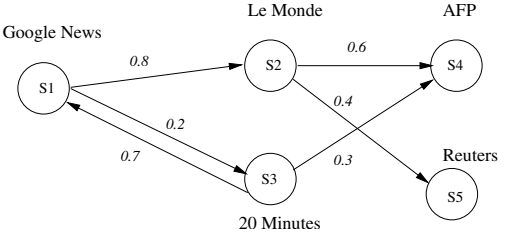


Fig. 2. Quality contribution graph

We are not interested in the way each service defines and acquires this knowledge about the relative quality contribution of other services. Whereas, for the sake of simplicity, this information is considered to be public in our model, Section 4 shows that public access is not needed for computing service importance.

The main goal of this article is to define a formal model and algorithms for ranking collaborating services according to their contribution to other services. The basic idea is to define and apply a link-based importance model considering service calls and quality contribution scores for estimating their importance among all other services. For example, if we consider only the quality contribution graph in Figure 2, it is easy to argue that *AFP* is more important than *Reuters* since it globally contributes more to the other services. This argument is enforced by Figure 1 which shows that *Reuters* has never been called by any other service. However, if we try to compare *Le Monde* and *20 Minutes* in the same way, we observe that both criteria (contribution and usage) are independent of each other. For example, *Le Monde* is obviously more important than *20 Minutes* when considering only its contribution to the quality of *Google News* but, as it is shown in Figure 1, it has been called a long time ago, which decreases its contribution to *Google News* compared to the one of *20 Minutes*.

The rest of the article is organized as follows. The next section presents related work. The formal model is defined in Section 3, followed by the presentation of two scalable distributed algorithms for computing service importance. These algorithms have been implemented and evaluated by simulation. The obtained results are described in Section 5. Section 6 shortly presents ongoing implementation and future work.

2 Related Work

Ranking services has been recognized as an important issue in the context of Web service discovery and dynamic service selection. Multiple criteria can be used for ranking services. For example, [7] propose to use advanced sampling techniques for comparing and ranking data-intensive Web services according to their local data. Recommendation based techniques exploit user feedback [17,12] and voting services [11] for dynamic service selection. Other ranking criteria are based on the conformance of QoS features during given time periods [21,12]. All these parameters can be combined using different weights according to specific user requirements [21,17]. In this context, our method can be seen as a recommendation-based approach for ranking services where each service call corresponds to an implicit vote taking into account service quality and observed usage during some time period.

Link-based ranking methods like PageRank [18] and HITS [14] have been developed and applied with success for ranking Web search results. The basic idea of this family of ranking models and algorithms is to consider that each page p propagates a fraction of its importance to all other pages p' it references. Most of the existing algorithms compute the importance of a page by exploiting the Web's hyperlink structure. OPIC [1] avoids the construction of the Web matrix by an adaptive on-line algorithm for estimating page importance dynamically during the Web crawling process.

Our ranking method exploits temporal information concerning the usage of a service for calculating a time-dependent importance score. More recent link-based approaches also consider time-dependent importance measures. For example, [4] observes that link-based importance penalizes new pages on the Web and proposes to decrease the PageRank of older pages. The same kind of argument is used by [8] that computes the importance of a page based on PageRank and its time derivative. [10] exploits temporal information for ranking news and their sources. A recent article is more important than an old one, and a source that produced recent important news is more important than other sources.

Importance based models and algorithms also have been applied in the context of P2P infrastructures. For example, [22] defines and computes importance scores for pages distributed in a P2P network. The importance of a page is defined as a combination of its local importance inside its peer and the global importance of its peer among the other peers. A similar approach is presented by [19], where each peer refines the score of its local pages by periodically meeting with other peers. A synchronous algorithm for reputation management in P2P systems is described in [13]. Each peer computes global trust value based

on the trust values of other peers and their local trust on it. [20] proposes a distributed asynchronous version of *PageRank* based on chaotic iterations. A totally asynchronous computation is presented by [15]. Our asynchronous iterative algorithm for computing the importance of collaborating services is inspired by [20,15] and considers the modification proposed by [5] for the termination of the asynchronous computation.

3 A Link-Based Service Importance Model

A service-oriented system is a set of services exchanging messages. We suppose that all messages are time-stamped by a synchronized continuous clock producing an infinite set of clock values $\mathcal{T}$[1]. For the sake of simplicity, we do not distinguish between different end-users and external applications and we assume that they are all represented by a single service in the system.

We define a *logging function* that associates to each pair of services (s_i, s_j) the time-stamps of all messages sent by s_i to s_j. More formally:

Definition 1 (logging function). *Let $\mathcal{S}$ be a finite set of identified services exchanging messages. A* logging function $\Lambda : \mathcal{T} \times S \times S \rightarrow 2^{\mathcal{T}}$ *returns for each instant $t \in \mathcal{T}$ and pair of distinct services (s_i, s_j), the set of time-stamps $\Lambda(t, s_i, s_j) = \{t_i | t_i \leq t\} \subseteq \mathcal{T}$ of all messages sent by s_i to s_j before t.*

Logging function Λ registers the collaboration between services as a set of *exchanged* messages. Note that Λ ignores local messages. At a higher level of abstraction it is possible to encode different kinds of message exchange patterns for composing messages into *service calls*. For example, in a request-response pattern (protocol) the service sending the first message (request) is considered to be the client of the receiver service whereas in a solicit-response protocol the sender of the first message (solicit) will be considered as the server of the second one. In the following, we will assume that logging function Λ only registers the timestamps t of request-response service calls $s_i \hookrightarrow^t s_j$ where t corresponds to the timestamp of the request message sent from s_i to s_j.

Definition 2 (service call graph). *Function Λ observes the activity between distinct services and generates a directed* service call graph $SC(t) = (\mathcal{S}, \mathcal{C}, \Lambda)$ *where (i) $\mathcal{S}$ is the set of vertices (ii) $\mathcal{C}$ is the set of edges, such that $(s_i \rightarrow s_j) \in \mathcal{C}$ iff $\Lambda(t, s_i, s_j) \neq \emptyset$.*

A service that is called contributes, directly or indirectly, to other services in the system. We denote by $Out(t, s_i) = \{s_j \mid s_j \in \mathcal{S} \wedge \Lambda(t, s_i, s_j) \neq \emptyset\}$ the set of services called by s_i until time instant t. Similarly, $In(t, s_j) = \{s_i \mid s_i \in \mathcal{S} \wedge \Lambda(t, s_i, s_j) \neq \emptyset\}$ is the set of services that called s_j before t. Then, a service s_j can contribute at some instant t directly to all services $s_i \in In(t, s_j)$ and transitively to all services s_k to which s_i contributes.

As shown in the introduction, we claim that we should distinguish between an intrinsic contribution of a service s_j to the *quality* a service s_i with respect

[1] Our model does not require exact synchronization between local clocks.

to all other services and a usage-based contribution that represents the way in which s_i actually *uses* s_j (independently of all other services).

3.1 Service Quality Contribution Scores

We assume that each service defines some measure for estimating its quality according to certain application-specific criteria. The key point behind these measures is that the quality of a service also depends on the quality of the services that it calls. For example, in a P2P search engine, the quality of a service might be defined by the average number of relevant documents returned to its clients. Other quality criteria might be the average freshness of its results (in the case of data replication with update), the average response time or even some simple business criteria based on the price of each service call.

We define the contribution score of a service s_j to the quality of the service s_i by a function Υ:

Definition 3 (local quality contribution). *Let $\mathcal{S}$ be a set of services. The* contribution function $\Upsilon : \mathcal{S} \times \mathcal{S} \to [0, 1]$ *defines for each pair of distinct services (s_i, s_j) a* local contribution score $\Upsilon(s_i, s_j) = \pi_{ji}$ *of service s_j to the quality of service s_i such that $\sum_{s_j \in Out(s_i)} \pi_{ji} = 1$.*

Function $\Upsilon(s_i, s_j)$ does not return a quantitative value but a *score* π_{ji} for comparing the contribution of all services $s_j \in Out(t, s_i)$ to the quality of service s_i. We also suppose that these scores are static and statistically independent of a particular call between s_i and s_j [2].

Each service s_i defines the local contribution π_{ki} of all services s_k that it uses independently on the services s_j that are used by s_k. Nonetheless the contribution of s_j to the quality of s_i through s_k can be estimated as the part of π_{ki} which is due to s_j, *i.e* $\pi_{ki} * \pi_{jk}$.

The *quality contribution graph* $SC(t, \Upsilon)$ is obtained by adding to each edge $s_i \to s_j$ in service call graph $SC(t)$ a label $\Upsilon(s_i, s_j) = \pi_{ji}$ (see for example the graph in Figure 2). Any path $p = s_i \to^{\pi_{ki}} s_k \to \ldots s_l \to^{\pi_{jl}} s_j$ in this graph then represents a contribution of service s_j to service s_i with a contribution score $\pi_p = \pi_{ki} * ... * \pi_{jl}$. We define the *global contribution* of some service s_j to the quality of *any* service $s_i \in \mathcal{S}$ as follows:

Definition 4 (global quality contribution). *We denote by $\mathcal{P}_{ij}$ the set of all possible paths from s_i to s_j in the contribution graph $SC(t, \Upsilon)$. The* global contribution π^*_{ji} *of service s_j to the quality of s_i is the sum of the contribution of s_j to s_i on all the possible paths p from s_i to s_j in $SC(t, \Upsilon)$: $\pi^*_{ji} = \sum_{p \in \mathcal{P}_{ij}} \pi_p$.*

Example 1. In Figure 2, *Le Monde* locally contributes to the quality of *Google News* with a score of 0.8. Since *Le Monde* calls *AFP*, *AFP* indirectly contributes to the quality of *Google News* with score $0.8 * 0.6 = 0.48$. Observe that the contribution graph might contain cycles which lead to an infinite number of

[2] This restriction can be relaxed but it simplifies the presentation and the understanding of our importance model.

contribution paths. For example, there exists an infinite number of contribution paths from *Google News* to *AFP* that pass through *20 Minutes*. The set of paths can be reduced to a finite one by eliminating the ones with $\pi_p < \varepsilon$ for a given ε. The contribution of *AFP* to the quality of *Google News* can then be computed as: $(0.8 * 0.6 + 0.2 * 0.3) * \Sigma_{i \geq 0}(0.7 * 0.2)^i = 0.63$.

3.2 Service Usage

The calls registered in $\Lambda(t, s_i, s_j)$ represent the effective usage of s_j by s_i at some given instant t (independently of the other services in the system). Obviously, a service can effectively contribute to the quality of other services only if it is called. In addition, the number and timestamps of the incoming calls influence its local and global contribution to other services. For each service couple (s_i, s_j) we introduce a *service usage score* that expresses the way in which s_j is used by s_i, depending on the calls received from s_i:

Definition 5 (local service usage). *The service usage function* $U_d : \mathcal{T} \times \mathcal{S} \times \mathcal{S} \rightarrow [0, 1]$ *returns for each time instant* t *and pair of services* (s_i, s_j) *a* local service usage score $U_d(t, s_i, s_j) = u_{ji}(t)$ *of service* s_j *by service* s_i *obtained by calls in* $\Lambda(t, s_i, s_j)$.

The definition and implementation of the usage function depends on the application semantics and any function aggregating service call time-stamps in $\Lambda(t, s_i, s_j)$ with an appropriate semantics could be chosen. Each service s_j can choose a specific function for each client $s_i \in In(t, s_j)$, or apply the same function to all of its clients. In the following examples we assume wlg. that the usage score $u_{ji}(t)$ decreases with the *age* of the last call received from client s_i where the decrease factor is the same for all clients of s_j. For instance, if s_j is a data provider which updates its data very frequently, it might choose a usage function which decreases very rapidly for all incoming service calls in order to take into account that it should be called frequently for obtaining maximal usage.

Example 2. Service s_1 (*Google News*) is interested by all news produced by service s_2 (*Le Monde*). Since *Le Monde* updates its news every day, *Google News* has to call it daily. In this case the service usage of s_2 by s_1 is maximal, *i.e* $u_{21}(t) = 1$ for any t. If the only call $s_1 \hookrightarrow^1 s_2$ from *Google News* to *Le Monde* has happened several days before instant 10, the usage score of *Google News* for *Le Monde* has decreased and might even be 0, reflecting the fact that no news received by *Le Monde* are still useful.

Service usage does not take into account possible logical or temporal relationships between incoming and outgoing calls of a service s_i. We introduce the notion of *call usage* that describes this kind of relationship between service calls.

Call usage scores are obtained by a function combining the information on the incoming service calls in $\Lambda(t, s_i, s_k)$ with the information on the outgoing service calls in $\Lambda(t, s_k, s_j)$ of the same service s_k. More formally:

Definition 6 (call usage). *Let $\mathcal{S}$ be a set of services observed by a logging function Λ.* Call usage function $U_i : \mathcal{T} \times \mathcal{S} \times \mathcal{S} \times \mathcal{S} \to [0,1]$ *returns for each instant t and triple of services (s_i, s_k, s_j) the* call usage score $U_i(t, s_i, s_k, s_j) = u_{jki}(t)$ *of outgoing service calls $s_k \to^{t'} s_j \in \Lambda(t, s_k, s_j)$ for incoming service calls $s_i \to^{t''} s_k \in \Lambda(t, s_i, s_k)$.*

Score $u_{jki}(t)$ defines the degree in which calls received by s_j from s_k contribute to calls received by s_k from s_i. Similarly to local service usage scores, the way in which incoming and outgoing service calls are compared depends on the application and the service implementation. For example a service s_i whose calls to some service s_k regularly trigger calls from s_k to some other service s_j leads to a high call usage score $u_{jki}(t)$.

Example 3. If we consider that all calls received by *Le Monde* only can exploit news generated before, the call usage of the calls to *AFP* for the calls from *Google News* via *Le Monde* at instant 1 is equal to 0 (*Le Monde* did not call *AFP* before instant 1).

We use the definition of call usage to generalize the notion of service usage by defining the *global service usage score* of a service s_j by any service $s_i \in \mathcal{S}$. When a service s_i calls service s_k that calls s_j, we can compute the service usage of s_j for service s_i by taking into consideration the service usage $u_{jk}(t)$ of s_j by s_k combined with the call usage $u_{jki}(t)$ between calls in $\Lambda(t, s_i, s_k)$ and calls in $\Lambda(t, s_k, s_j)$. Consequently, the service usage of s_j by service s_i through s_k is estimated as $u_{jki}(t) * u_{jk}(t)$. More generally, for a call path $p = s_i \to s_l \to s_m \to \ldots \to s_n \to s_k \to s_j$ in $SC(t)$ the global usage score of s_j for s_i on this path is defined by the product $u_p(t) = u_{mli}(t) * \ldots * u_{jkn}(t) * u_{jk}(t)$.

Definition 7 (global service usage). *Let $\mathcal{P}_{ij}$ be the set of all possible paths from s_i to s_j in the service call graph $SC(t)$. The* global service usage score $u^*_{ji}(t)$ *of service s_j by service s_i at instant t is the sum of the service usage scores of s_j for s_i on paths in $\mathcal{P}_{ij}$: $u^*_{ji}(t) = \sum_{p \in \mathcal{P}_{ij}} u_p(t)$.*

3.3 Effective Contribution and Importance

Our objective is to compare and rank services with respect to their activity observed by logging function Λ combined with the information about how each service contributes to the quality of other services in a service-oriented system.

The importance of s_j is related to its *effective contribution* to *all* services $s_i \in \mathcal{S}$, which is computed by combining contribution and usage scores on all call paths in $SC(t)$. The main idea is to weight contribution scores on each path from s_i to s_j with the corresponding usage score, an important service being one that highly contributes by its usage to the quality of other services.

Definition 8 (local effective contribution). *The* local effective contribution $\tilde{\pi}_{ji}(t)$ *of service s_j to service $s_i \in In(t, s_j)$ at instant t is the product of the local quality contribution and the local usage score of s_j by s_i : $\tilde{\pi}_{ji}(t) = \pi_{ji} * u_{ji}(t)$.*

As for quality contributions and for usage scores, we can extend the notion of local effective contribution to services connected by service-call paths.

For a given service-call path $p = s_i \rightarrow s_l \rightarrow s_m \rightarrow \ldots \rightarrow s_n \rightarrow s_k \rightarrow s_j$ in $SC(t)$ the *effective contribution* of s_j to s_i on this path, denoted by $\tilde{\pi}_p(t)$, is computed by multiplying the quality contribution score with the service usage sore of s_j for s_i on this path:

$$\tilde{\pi}_p(t) = \pi_p * u_p(t) = \overbrace{\pi_{li} * \pi_{ml} \ldots \pi_{kn} * \pi_{jk}}^{\text{quality contribution}} * \overbrace{u_{mli}(t) * \ldots * u_{jkn}(t) * u_{jk}(t)}^{\text{usage contribution}} \quad (1)$$

Definition 9 (global effective contribution). *Let $\mathcal{P}_{ij}$ be the set of all possible paths from s_i to s_j in the service call graph $SC(t)$. The* global effective contribution score *$\tilde{\pi}^*_{ji}(t)$ of s_j to s_i is obtained as the sum of the effective contribution on all possible paths $p \in \mathcal{P}_{ij}$:*

$$\tilde{\pi}^*_{ji}(t) = \sum_{p \in \mathcal{P}_{ij}} \tilde{\pi}_p(t) \quad (2)$$

A service is then important if it has a great effective contribution to all other services in the system.

Definition 10 (service importance). *The* importance *of a service s_j is defined as the sum of its global effective contributions to all services $s_i \in \mathcal{S}$:*

$$I_j(t) = \sum_{s_i \in \mathcal{S}} \tilde{\pi}^*_{ji}(t) \quad (3)$$

Example 4. Figures 2 and 1 show that service s_4=*AFP* contributes to service s_2=*Le Monde* with score $\pi_{42} = 0.6$ and has been called by *Le Monde* at instant 9. If service usage $u_{42}(t)$ at instant $t = 10$ is estimated by the age of the last call and the time elapsed between instant 10 and 9 is only several hours, the effective contribution of *AFP* to *Le Monde* is high (close to the quality contribution score). On the other hand, even if *Le Monde* highly contributes to the quality of *Google News*, it has been called by *Google News* several days before instant 10, which leads to a low effective contribution. By a similar kind of reasoning we can see that even if the quality contribution $\pi_{42} * \pi_{21}$ of *AFP* to *Google News* through *Le Monde* is high, the service contribution of *AFP* to *Google News* via *Le Monde* is low. This is due to the low call contribution score of the calls to *AFP* for the calls from *Google News* (when the call at a given instant exploits news generated before this instant, as described in example 3). On the other hand, the quality contribution $\pi_{43} * \pi_{31}$ of *AFP* to *Google News* via *20 Minutes* is enforced by a high service usage score $u_{431}(t) * u_{43}(t)$.

4 Computing Importance

This section presents two algorithms for computing the importance of services at time τ in a service-oriented system $\mathcal{S}$. The fundamental idea is to exploit the

existing service-oriented architecture for distributing the computation on the different service nodes. The presented algorithms can efficiently be deployed on large-scale service infrastructures since each service computes its importance by exchanging messages only with its already known neighbor services in the service call graph. The computed importance values could then be registered along with other information on the services in an UDDI registry.

In both algorithms, each service s_k computes its own importance $I_k(\tau)$ at τ by exchanging messages with neighbor services in $In(\tau, s_k)$ and $Out(\tau, s_k)$. Computation is iterative until convergence: each service s_k starts its computation with an initial importance value $I_k^0(\tau)$ and computes a new importance approximation based on *importance values* $\nu_{ki}(\tau)$ received from its neighbors $s_i \in In(\tau, s_k)$. It also sends importance updates $\nu_{jk}(\tau)$ to services $s_j \in Out(\tau, s_k)$ and stops its computation when the relative error between the newly computed importance and the previous one is lower than a given (sufficiently small) ε.

The importance value $\nu_{jk}(\tau)$ received by s_j from each $s_k \in In(\tau, s_j)$ expresses (i) the quality contribution π_{jk} of service s_j to service s_k and, recursively, (ii) the quality contribution and call usage of service s_j for other services s_i via s_k. More formally, the *received importance* $\nu_{jk}(\tau)$ received by s_j from some client s_k is computed using the importance values $\nu_{ki}(\tau)$ received by s_k from its clients s_i:

$$\nu_{jk}(\tau) = \pi_{jk} + \sum_{s_i \in In(\tau, s_k)} \nu_{ki}(\tau) * u_{jki}(\tau) * \pi_{jk} \tag{4}$$

Observe that by definition $\nu_{jk}(\tau) = \pi_{jk}$ if no service call from service s_k to service s_j has been useful to any service call received by service s_k.

The importance $I_j(\tau)$ of a service s_j at some moment τ is the sum of the received importance values $\nu_{jk}(\tau)$ weighted by the direct utility scores $u_{jk}(\tau)$:

$$I_j(\tau) = \sum_{s_k \in In(\tau, s_j)} \nu_{jk}(\tau) * u_{jk}(\tau) \tag{5}$$

We suppose that s_j knows its own service usage $u_{jk}(\tau)$ at the beginning of the algorithm. The proof that equation 5 computes the same importance values as defined by equation 3 is given in [9].

We propose in the following section two distributed algorithms which are different by the protocol used for exchanging importance values between services. In the first algorithm services synchronize their computation by sending new importance values only after having received importance values from all of its clients. In the second algorithm importance messages are not synchronized. Both algorithms are evaluated and compared in Section 5.

4.1 Synchronous Distributed Computation

In this algorithm each service s_k first collects importance values $\nu_{ki}(\tau)$ from all services $s_i \in In(\tau, s_k)$, recomputes its importance and sends it to services $s_j \in Out(\tau, s_k)$. Variables O_k and N_k are vectors where o_{ki} and n_{ki} are old and new importance values received by s_k from $s_i \in In(\tau, s_k)$ during the computation.

All services choose a common time τ and start the computation with an initial importance value I_k^0 equal to 0. Then each service $s_k \in \mathcal{S}$ executes the following algorithm:

ComputSync(τ)
Input: a time τ, $\rho = \emptyset$
Output: importance of the service s_k at time τ
begin
  $O_k = N_k = 0$
  do
    for each $s_j \in Out(\tau, s_k)$ **do** // forward importance values
      $\nu_{jk}(\tau) = \lambda * \pi_{jk} + \lambda * \sum_{s_i \in In(s_k)} n_{ki} * u_{jki}(\tau) * \pi_{jk}$
      send $\nu_{jk}(\tau)$ to s_j
    endfor
    $Tmp = In(\tau, s_k)$ //services that did not send their importance
    while $Tmp \neq \emptyset$ **do** // compute new importance
      $n_{ki} = \nu_{ki}(\tau)$ //wait for next importance message
      $Tmp = Tmp \setminus \{s_i\}$
    endwhile
    if $|N_k - O_k|/O_k \geq \varepsilon$ **then**
      $\rho = \rho \setminus \{s_i\}$
    else
      $\rho = \rho \cup \{s_i\}$ // local convergence
    endif
    $I_k(\tau) = \sum_{s_i \in In(s_k)} n_{ki} * u_{ki}(\tau)$ //compute service importance
    $O_k = N_k$
  while $\rho \neq \mathcal{S}$ // stop when all the services have converged
end

The above algorithm implements the Jacobi iterations which compute the received importance as the solution of the linear system presented in [9]. Importance values are multiplied by a constant value $\lambda \in (0, 1)$ (similarly to [18]) which guarantees that the solution of the system exists and is unique and that the Jacobi iterations converge to the solution of the system independently on the initial importance values I_k^0 (see [9] for a formal proof). The received importance of service s_j converges locally when $|N_k - O_k|/O_k < \varepsilon$ for a given ε. Local convergence does not necessarily imply global convergence, some services converge faster than others. ρ denotes the set of services that have converged after each step of the algorithm. Computation stops when all services $s_k \in \mathcal{S}$ have converged locally, *i.e*, when $\rho = \mathcal{S}$.

4.2 Asynchronous Distributed Computation

The algorithm presented in this section avoids additional computation messages and the delays due to synchronization by embedding importance values into application-specific service calls at instants $t > \tau$. The only "synchronization" between services consists in choosing a time τ for starting the computation.

Similarly to the previous algorithm, we use a vector N_k whose elements are initialized to 0 at the beginning of the algorithm. The algorithm is defined as follows:

ComputAsync(τ): At each service call $s_k \hookrightarrow^t s_j$, where $t > \tau$:

- Sender s_k
  1. computes importance value $\nu_{jk}(\tau) = \lambda * (\pi_{jk} + \sum_{s_i \in In(\tau, s_k)} n_{ki} * u_{jki}(\tau) * \pi_{jk})$
  2. embeds $\nu_{jk}(\tau)$ into the service call $s_k \hookrightarrow^t s_j$ (e.g. as an additional element of a SOAP message)
  3. calls service s_j
- Receiver s_j
  1. memorizes the received importance value in n_{jk} and
  2. updates its importance $I_j(\tau) = \sum_{s_k \in In(\tau, s_j)} n_{jk} * u_{jk}(\tau)$.

Each service s_k recomputes its local importance at each incoming service call without waiting for all services $s_i \in In(\tau, s_k)$ to send their importance. Service s_k communicates its new importance values $\nu_{jk}(\tau)$ to its clients $s_j \in Out(\tau, s_k)$ only when it issues a new call to s_j. The result is an asynchronous protocol where each services computes its importance at its own pace, based on possibly outdated importance values. Note that some services might update their importance or might communicate more frequently than others.

The above algorithm implements the totally asynchronous iterations that compute the solution of the linear system presented in [9]. We suppose that our system fulfils the *total asynchronism* assumption [5], i.e., all importance values are updated infinitely often and old values are potentially purged from the system. Each service s_k has to be eventually informed on the importance updates from all services $s_i \in In(\tau, s_k)$. Then the totally asynchronous iterations converge to the solution of the linear system (see [9]). For the algorithm to terminate, conforming to [5], the importance $\nu_{jk}(\tau)$ is sent to s_j only if it differs with more than ε from the last importance sent by s_k to s_j. When the computation converges, no update are sent to s_j anymore. The algorithm termination is detected when i) no message is in transit and ii) any recomputation of $I_k(\tau)$ does not change its value (conforming to [5]).

The above algorithms compute the importance of each service s_j at time instant τ iteratively based on equations 4 and 5. In [9] we show that the importance obtained when the algorithms converge corresponds to the sum of the global effective contribution of s_j to all services $s_i \in \mathcal{S}$ (equation 3). It also is easy to show that each services consumes limited memory of linear size depending on the number of clients (for storing old and new importance values). Global complexity (in terms of the maximal and average number of iterations) is evaluated in the following section.

5 Experimental Evaluation

We implemented both algorithms in Java (JDK 1.5) on a AMD Turion 64 laptop (1.6GHz, 2Gb RAM) under SUSE 10.0 Linux. In the following experiments we

considered a network of 1000 collaborating services which were simulated in form of Java threads. We generated different network configurations (described later) defined by the way each service s_i chooses its neighbor services $s_j \in Out(\tau, s_i)$ that contribute to its quality at some time instant τ. Quality contribution is distributed uniformly for each service in $Out(\tau, s_i)$: $\pi_{ji} = 1/|Out(\tau, s_i)|$. For modeling service and call utility we assign to each edge from s_i to s_j a random value $\delta_{ji} \in [0, 10]$ representing the age of the last call from s_i to s_j with respect to the instant τ. We consider that all service use the same utility functions $u_{ji}(\tau) = 1 - \alpha * \delta_{ji}$ (service utility) and $u_{jki} = 1 - \alpha * |\delta_{ki} - \delta_{jk}|$. Utility factor α is used to control the influence of the utility function on the importance computation ($\alpha = 0$ means that all service contribution links are taken into consideration during computation ignoring the age of the last service calls).

Each service s_i starts its computation with an importance value of 0, and stops its computing after *local convergence* (when $|N_i - O_i|/O_i < \varepsilon$). In the synchronous algorithm described in section 4, some services might converge faster than others and we consider that a service s_i performs an *iteration* when it receives importance updates from all services in $In(\tau, s_i)$ that did not converge yet (this is different from the presented algorithm, where each service waits for updates from all services in $In(\tau, s_i)$). For asynchronous computation we suppose that a service s_i performs an iteration when it makes a number of importance updates which is equal to the number of its neighbors in $In(\tau, s_i)$. In the following, unless specified otherwise, all experiments are run with threshold $\varepsilon = 10^{-4}$, utility factor $\alpha = 0$ and $\lambda = 0.85$.

5.1 Service Graph Generation

In the following we will call the services in $Out(\tau, s_i)$ the neighbors of s_i. The neighbors of a service are chosen by the following four strategies, leading to service importance graphs with different topologies:

Max-graph [MAX]. This graph is similar to the Web graph model proposed by [2]. Each service chooses as neighbors with probability 0.75 five "popular" services, i.e. services which already have been chosen by many other services.

Linear-copying graph [LC]. Each service randomly selects a "prototype" service p among all existing services. It then chooses five neighbors among all services where each such neighbor is with probability 0.75 a neighbor of p. The obtained graph is similar with the Web graph model proposed by [16].

Small-world network [SW]. This configuration simulates a *small-world network* by creating 20 communities composed of 50 services. Each service in such community randomly connects to 5 neighbors in the same community and each community interacts on average with 5 services of other, randomly drawn, communities.

Client-server configuration [CS]. This configuration combines the above three strategies to model a client-server setting with 80 client communities calling services of a single server community (SW). Each client community contains 10

services with a "prototype" service connected to some randomly chosen server services. The other 9 services in the same community are connected randomly to 1 service in the same community and to at most to 5 server services, each server being with probability 0.75 a neighbor of the community's prototype (LC strategy). The server community is MAX graph composed of 200 server services connected in average to 5 other server services.

5.2 Experimental Results

Figure 3 shows, for all four graph configurations, the average number of computation messages generated per service until convergence using global synchronization, local synchronization (do not wait for services that already have converged) and no synchronization. We see that global synchronization generates the highest number of messages since services that already have locally converged at some iteration step i still keep sending messages until all services have converged. With local synchronization, services which have converged stop sending messages. Using the asynchronous algorithm services converge faster than with the locally synchronous algorithm(as shown by figure 5). The reason for this gain is that a service does not systematically send freshly computed importance values to all its neighbors, as in the synchronous algorithm, but "optimizes" communication by sending new importance values only once in a while. We see that SW and LC generate more messages than the other configurations. Services in SW are randomly connected, which might lead to many cycles and long contribution paths for many services. The same argument holds for prototype services in LC which are chosen randomly. On the contrary, contribution paths in MAX are rather short, as all services link to the popular services. There are many services which are not neighbors of other services and that converge very fast. In CS there are many independent small communities with few connections.

Figure 4 shows the influence of the threshold value ε used for convergence on the number of iterations. As expected, lower values for ε lead to a higher number of iterations, since we should take into consideration longer paths (with lower contribution) to achieve convergence. Nonetheless Figure 4 illustrates that the

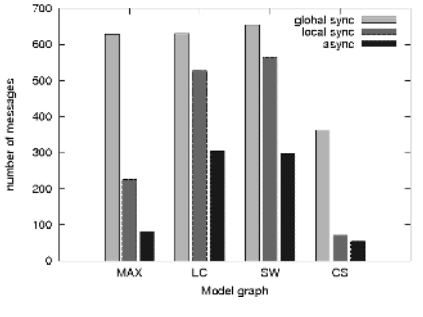


Fig. 3. Number of messages for the different models

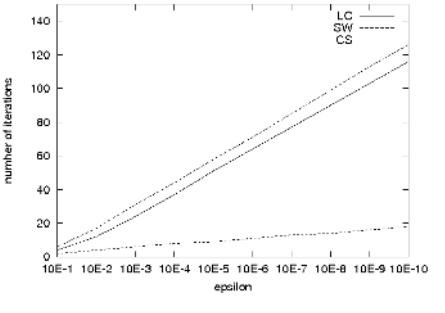
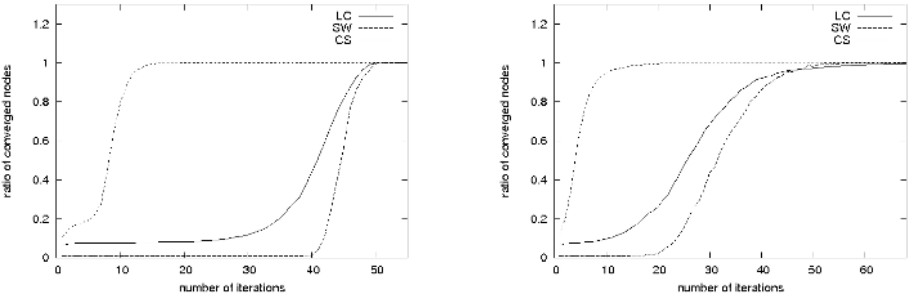


Fig. 4. Number of iterations for convergence w.r.t. ε



(a) synchronous algorithm (b) asynchronous algorithm

Fig. 5. Ratio of converged nodes w.r.t. number of iterations

growth in the number of iterations is logarithmic independently on the graph model. Similar results were reported by [6] on the convergence of PageRank.

Figures 5(a) and (b) show the ratio of the importance values that reached their final ranking with respect to the number of iterations. With the synchronous algorithm (figure 5(a)) for LC (resp. SW) only 10% (resp. 1%) of the services have converged after 30 iterations. Global convergence is achieved after about 50 iterations. This can be explained by the high connectivity of the small-world communities that leads to a large number of possible paths to be explored. On the contrary, the services in CS converge more quickly (after 10 iterations almost all the services have converged) since client services are grouped in many small communities. By comparing figure 5(a) with figure 5(b) we first note that removing synchronization allows services to converge faster. The reason is that a service sends in each importance message more information on the contribution paths than in the synchronous algorithms, the convergence being then accelerated. In other words, with the asynchronous algorithm services "learn" more quickly all their contribution paths.

We also performed several experiments to illustrate the impact of the utility function on the importance computation. In Figure 6 we see that the number of iterations decreases when the value of α increases, independently of the graph model. This was expected since smaller utility values lead to lower connectivity along with smaller paths since the quality contribution of a service on a path is reduced by the utility factor α. In Figure 7 we study the influence of the utility factor α on the ranking of the services for three graph configurations. We consider as reference the ranking obtained for $\alpha = 0.0$ (ranking obtained by considering only service contribution links without service utility). For different values of α we compute the fraction of the services that are still in the top 10 and top 50 services. We see that service utility strongly impacts the obtained ranking. For example, in the SW configuration and for $\alpha = 0.2$ only 20% of the most important contributing services still belong to the top 10 (resp. top 50) services.

Finally Table 1 illustrates the influence of a service on the importance of other services. After an initial importance computation, we removed a randomly chosen node and recomputed the importance values of all services. The table shows

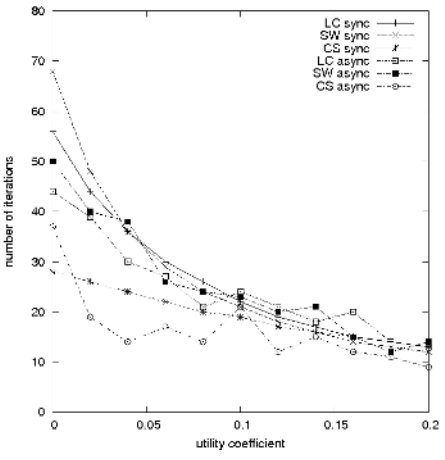


Fig. 6. Number of iteration for different utility functions

α	LC		SW		CS	
	10	50	10	50	10	50
0.0	10	50	10	50	10	50
0.04	9	46	7	39	10	47
0.08	8	45	6	30	10	44
0.12	7	42	4	26	10	42
0.16	6	39	2	17	8	38
0.2	4	25	2	10	4	27

Fig. 7. Influence of the utility on the result set

Table 1. Cost of the recomputing when 1 peer leaves

	without utility						utility: $\alpha = 0.1$					
	sync			async			sync			async		
	nodes	path	mess	nodes	path	mess	nodes	path	mess	nodes	path	mess
LC	938	18	16682	887	99	7670	938	14	12694	764	61	2979
SW	990	32	12577	450	210	16325	995	16	2223	256	85	2854
CS	896	200	230	101	31	648	898	126	200	94	20	245

the number of services involved in the importance recomputation, the number of recomputation messages that are exchanged and the length of the maximum path that is taken into consideration. We see that with the locally synchronous algorithm almost all services are involved in the importance computation, whereas with the asynchronous one only a part of services in the neighborhood of the removed node recompute their importance. For instance with $\alpha = 0.1$ and CS graph, only 94 nodes participate to the computation with the asynchronous algorithm. Note also that the path lengths and the number of exchanged messaged are smaller with a utility of 0.1. The difference in the path lengths and in the number of exchanged messages between the synchronous and the asynchronous algorithm seems to be dependent on the graph topology. For example, with the LC graph the number of messages of the synchronous algorithm are greater than the ones for the synchronous one for both values of α, whereas for SW the contrary is true.

6 Conclusion

This article presents a general framework for ranking services with respect to their global contribution to other services. Whereas the basic approach has been

illustrated in the context of a service-based news syndication system, we believe that the proposed model and algorithms are useful for many other applications like web service discovery and selection [7,21], service-based P2P data warehousing [3] and XML data integration. We are currently evaluating our model in the context of data-centric web services where each service is defined as a parameterized views on a local data repository containing the results of calls to other services [3]. The basic idea is to redefine service usage and quality contribution in terms of service call expiration, data validity and queries.

Finally, we are also aware that there are many open security issues related to our importance model and algorithms. In particular, we do not tackle the problem of services which try to cheat by increasing their importance artificially or by sending fake importance values. This phenomenon is also well-known in the context of traditional web search engines and P2P systems and we believe that existing solutions for securing the computation (like in [13]) could be adapted to our algorithms. This is also part of our future work.

References

1. S. Abiteboul, M. Preda, and G. Cobena. Adaptive On-Line Page Importance Computation. In *Proc. Intl. World Wide Web Conference (WWW)*, pages 280–290, 2003.
2. R. Albert, H. Jeong, and A.-L. Barabási. The Diameter of the World Wide Web. *Science*, 286:509–512, 1999.
3. The Active XML Project. http://activexml.net.
4. R. A. Baeza-Yates, C. Castillo, and F. Saint-Jean. Web Dynamics, Structure, and Page Quality. In *Web Dynamics*, pages 93–112. 2004.
5. D. P. Bertsekas and J. N. Tsitsiklis. Some aspects of parallel and distributed iterative algorithms-a survey. *Automatica*, 27(1):3–21, 1991.
6. M. Bianchini, M. Gori, and F. Scarselli. Inside PageRank. *ACM Transactions on Internet Technology (TOIT)*, 5(1):92–128, 2005.
7. J. Caverlee, L. Liu, and D. Rocco. Discovering and ranking web services with BASIL: a personalized approach with biased focus. In *Proc. Intl. Conf. on Service-Oriented Computing (ICSOC)*, pages 153–162, 2004.
8. J. Cho, S. Roy, and R. Adams. Page Quality: In Search of an Unbiased Web Ranking. In *Proc. ACM Symp. on the Management of Data (SIGMOD)*, 2005.
9. C. Constantin, B. Amann, and D. Gross-Amblard. A Link-based Ranking Model for Services (long version), 2006. http://www-poleia.lip6.fr/~ amann/coopis_long.pdf.
10. G. M. D. Corso, A. Gulli, and F. Romani. Ranking a Stream of News. In *Proc. Intl. World Wide Web Conference (WWW)*, pages 97–106, 2005.
11. F. Emekçi, O. D. Sahin, D. Agrawal, and A. E. Abbadi. A Peer-to-Peer Framework for Web Service Discovery with Ranking. In *Proc. Intl. Conf. on Web Services (ICWS)*, pages 192–199, 2004.
12. S. Kalepu, S. Krishnaswamy, and S. W. Loke. Reputation = f(User Ranking, Compliance, Verity). In *Proc. Intl. Conf. on Web Services (ICWS)*, pages 200–207, 2004.
13. S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina. The Eigentrust algorithm for reputation management in P2P networks. In *Proc. Intl. World Wide Web Conference (WWW)*, pages 640–651, 2003.

14. J. M. Kleinberg. Authoritative Sources in a Hyperlinked Environment. *J. ACM*, 46(5):604–632, 1999.
15. G. Kollias, E. Gallopoulos, and D. B. Szyld. Asynchronous Iterative Computations with Web Information Retrieval Structures: the PageRank Case. In *Proc. Intl. Conf. on Parallel Computing (PARCO)*, 2005.
16. R. Kumar, P. Raghavan, S. Rajagopalan, D. Sivakumar, A. Tomkins, and E. Upfal. Random Graph Models for the Web Graph. In *Proc. Intl. Symp. on Foundations of Computer Science (FOCS)*, pages 57–65, 2000.
17. Y. Liu, A. H. H. Ngu, and L. Zeng. QoS computation and policing in dynamic web service selection. In *Proc. Intl. World Wide Web Conference (WWW)*, pages 66–73, 2004.
18. L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank Citation Ranking: Bringing Order to the Web. Technical report, Stanford Digital Library Technologies Project, 1998.
19. J. X. Parreira and G. Weikum. JXP: Global Authority Scores in a P2P Network. In *Proc. Intl. Workshop on the Web and Databases (WebDB)*, pages 31–36, 2005.
20. K. Sankaralingam, S. Sethumadhavan, and J. C. Browne. Distributed Pagerank for P2P Systems. In *Proc. Intl. Symp. on High Performance Distributed Computing (HPDC)*, pages 58–69, 2003.
21. L.-H. Vu, M. Hauswirth, and K. Aberer. QoS-Based Service Selection and Ranking with Trust and Reputation Management. In *Proc. Intl. Conf. on Cooperative Information Systems (CoopIS)*, pages 466–483, 2005.
22. Y. Wang and D. J. DeWitt. Computing PageRank in a Distributed Internet Search Engine System. In *Proc. Intl. Conf. on Very Large Data Bases (VLDB)*, pages 420–431, 2004.

Quality Makes the Information Market*

B. van Gils, H.A. (Erik) Proper, P. van Bommel, and Th.P. van der Weide

Institute for Computing and Information Sciences, Radboud University Nijmegen
Toernooiveld 1, 6525 ED Nijmegen, The Netherlands, EU

Abstract. In this paper we consider information exchange via the Web to be an information market. The notion of quality plays an important role on this information market. We present a model of quality and discuss how this model can be operationalized.

This leads us to quality measurement, interpretation of measurements and the associated accuracy. An illustration in the form of a basic quality assessment system is presented.

1 Introduction

The amount of information available to us has been increasing at an explosive rate over the last few years, especially with the massive growth of the Web. Several tools and systems have been developed to help us manage the vast amount of available resources such as indexes, search engines, catalogues and so on. These tools can, to some extent, be seen as information retrieval tools.

Basic background on information retrieval is found in [17] and [16]. The traditional information retrieval paradigm is introduced in Figure 1.

A main challenge in information retrieval is the correct formulation of information requests. See the left part of Figure 1. In the middle and right part, we see brokering or matching, and characterisation.

Characterising supply: Good characterisation of resources is imperative for effective information discovery, as poor characterisations inevitably lead to the retrieval of irrelevant information, or omit relevant information.

Matching demand and supply: The selection of relevant resources for a given query is a well understood problem. The field of information retrieval has developed a number of retrieval models.

Besides traditional performance measures such as precision and recall, a notion of quality in the broader sense is essential in modern information retrieval. Relevant quality aspects are the following. What is the quality of the characterisation of resources? What qualities do resources have? What is the quality of a query? How well is a query formulated and how accurately does it describe the searchers information need? What is the quality of a search engine or match maker? What are its qualities?

* The investigations were partly supported by the Dutch Organization for Scientific Research (NWO).

R. Meersman, Z. Tari et al. (Eds.): OTM 2006, LNCS 4275, pp. 345–359, 2006.

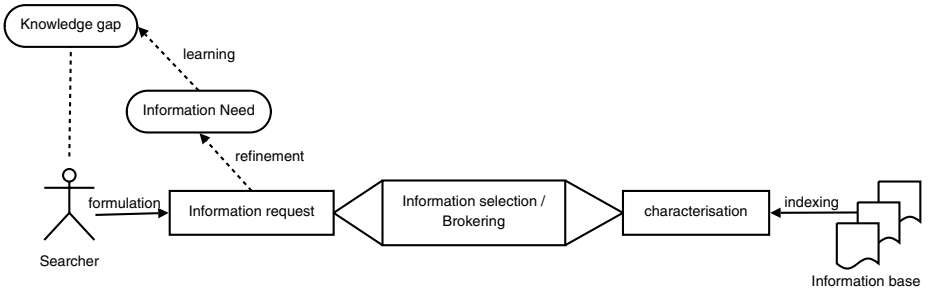


Fig. 1. The information retrieval paradigm

Quality has received a lot of attention in the general area of computing and information science. Section 2 contains examples and references to the literature. The specific domain of the Web information market still lacks a proper notion of quality, though. Our research focus is to work towards a generic model of quality of resources and to show how it can be used in the context of the Web. This research on quality is part of a larger project on information supply on the Web. This project has been introduced in [7]. More details about our approach of information supply were presented in for instance [1] and [8].

The organisation of this paper is as follows. In Section 2 we give a survey of the literature on quality in general. Then we discuss our quality model in Section 3. We discuss the operationalisation of our quality model for the Web context in Section 4. Finally, we illustrate an application in terms of an example quality assessment system in Section 5, while Section 6 gives conclusions and directions for future research.

2 Background

From the dictionary definition of quality[1] we learn that the notion of quality has two distinct interpretations: (1) a distinguishing characteristic similar to a property, and (2) inherent or intrinsic excellence, in other words how *good* some artefact is.

The term quality has a long history, for example in his work on *the philosophy of nature*, Aristotle used the notion of quality[2]. In his view, quality is the category according to which objects are said to be like or unlike. Other great philosophers such as Descartes, Bacon, Newton, and Galileo oppose to Aristotle's view on quality[3] mainly because they make a distinction between objective qualities and largely subjective qualities.

[1] We have consulted *Webster's third new international dictionary, unabridged* and *Concise Oxford Dictionary*

[2] http://www.utm.edu/research/iep/a/aristotl.htm and http://www.newadvent.org/cathen/12589c.htm

[3] http://www.ul.ie/~philos/vol1/eustac1.html

In e-commerce the notion of quality plays an important role. Two main examples in this respect are the problem of uncertainty with regards to the product/service to be traded and the lack of quality information about the traded assets, see [19] and [11] respectively. These problems are similar in the field of operations management where one is mainly concerned with key dimensions of quality such as product attributes, product performance, warranty, and service availability This is discussed in [10]. A *conformance to specification* approach is very popular in this field, but is criticized in [12] because it would focus too much on the supplier perspective, whereas the consumer perspective would focus on value for money. In [15] the focus is on the ex-post evaluation of quality of information in organizations, based on the ISO-8402 definition of quality:

> The totality of features and characteristics of a product, process or service that bear on its ability to satisfy stated or implicit goals.

Using this definition, in [15] a dual view on quality is proposed. On the one hand, the causal point of view deals with the quality of information, seen as the result of the quality of the process in which it is produced. On the other hand, in the teleological point of view the quality of information is seen as the degree to which it satisfies stated or implicit needs, derived from the situation in which it is used.

In the field of software engineering, the notion of quality plays the role of quality of software, and quality of the software engineering process. In this field the emphasis is mainly on quality attributes. See for instance [18]. Examples of these attributes are safety, security, reliability, resilience, robustness and learnability. Furthermore software quality management can be structured into three principle activities: quality assurance, quality planning, and quality control. See [3, 2, 13, 6] for further discussions on quality in the context of software engineering.

In [5] a discussion on the quality of data on the Web is presented. This discussion starts with the observation that "well-founded and practical approaches to assess or even guarantee a required degree of the quality of data are still missing". In order to overcome this defect the authors propose that a quality algebra be used for dealing with quality issues on the web. Such an algebra is particularly useful for intermediaries on the information market. See for example [7]. In [20] it is posed that user concerns about their perception of the quality of information on the Web continues to be a strong incentive for "the emergence and success of information intermediaries". They can play an important role in the trust relationship between suppliers and consumers, as well as in the control of quality versus price. Last but not least, the approach of [14] is that data quality is the measure of the agreement between the data view presented by an information system and some data in the real world.

3 A Model of Quality

Upon closer examination, the above definitions and applications of quality show that there are three main views on quality:

Property: the quality properties some artefact may poses.
Excellence: the actual quality of some artefact with regards to some property.
Desirability: the desired qualities (by some actor/buyer/user) of some artefact with regards to some property.

In computing terms, one might think of the first view as a variable, the second view the value that can be assigned to this variable after evaluating the quality of some artefact, while the third view corresponds to the value that can be assigned to this variable when considering the desires of some actor/buyer[4].

In our approach, quality has to be made specific and precise in order to be able to reason about it. We therefore provide a more formal elaboration of the notion of quality. We will do so in two steps. First we discuss quality as excellence, where we will consider both the quality properties (the variables) and the excellence (the value) of some artefact with regard to a property. Then we move on to the desirability of quality properties.

3.1 Quality as Excellence

In this subsection we introduce a model for the properties that artefacts can have. In the formalisation that follows we will use the following notation:

$\mathcal{AF}$ Artefact	$\mathcal{PT}$ Property type
$\mathcal{RO}$ Role type	$\mathcal{PD}$ property domain
$\mathcal{FL}$ Fulfillment	$\mathcal{VL}$ Value

Let $\mathcal{AF}$ be the set of all artefacts that may have certain qualities or properties, and let $\mathcal{RO}$ be the set of all roles that these artefacts can fulfill. The combination of an artefact and a role is dubbed a *fulfillment*. So a fulfillment denotes an artefact in a role. Fulfillments are captured by $\mathcal{FL}$. The artefacts and roles that participate in a fulfillment can be found using the functions $\mathsf{Artefact} : \mathcal{FL} \to \mathcal{AF}$ and $\mathsf{Role} : \mathcal{FL} \to \mathcal{RO}$ respectively. Since a fulfillment denotes an artefact in a role we know that an artefact and a role combination uniquely determines a fulfillment:

Axiom 1 (Unique fulfillment)

$$\mathsf{Artefact}(f_1) = \mathsf{Artefact}(f_2) \ \wedge \ \mathsf{Role}(f_1) = \mathsf{Role}(f_2) \quad \Longrightarrow \quad f_1 = f_2$$

For a fulfillment $f \in \mathcal{FL}$ we introduce the following notation:

$$\langle a, r \rangle \ \triangleq \ f \quad \text{such that} \quad \mathsf{Artefact}(f) = a \ \wedge \ \mathsf{Role}(f) = r$$

The following example illustrates this. Let *Mug* denoted by a be an artefact that can play two roles. It either plays the role of type *some artefact to drink from* denoted by r_1, or the role of type *art object* denoted by r_2. Both $f_1 = \langle a, r_1 \rangle$ and $f_2 = \langle a, r_2 \rangle$ are fulfillments such that:

$$\begin{array}{ll} \mathsf{Artefact}(f_1) = a & \mathsf{Role}(f_1) = r_1 \\ \mathsf{Artefact}(f_2) = a & \mathsf{Role}(f_2) = r_2 \end{array}$$

[4] We would like to thank one of the anonymous referees for suggesting this analogy.

Role types can have properties, the value of which are expressed in a property domain. For example, the role type *art object* can have the property type *color* and the values that this property can take are expressed in the domain *RGB-colors*. Let $\mathcal{PT}$ be the set of property types and $\mathcal{PD}$ be the set of property domains. The properties that can be played by a certain role type are given by the function $\mathsf{Props} : \mathcal{RO} \rightarrow \wp(\mathcal{PT})$ and the domain in which values of a property can be expressed is given by the function $\mathsf{PrDom} : \mathcal{PT} \rightarrow \mathcal{PD}$.

We continue the above mentioned example. Role type *art object* denoted by r_2 can have the property type *color* denoted by p which can be expressed in the domain *RGB-colors* denoted by d. As a consequence, we have:

$$\begin{aligned} \mathsf{Props}(r_2) &= \{p\} \\ \mathsf{PrDom}(p) &= d \end{aligned}$$

Note that property types and domains are at the typing level. We still need to assign values to entities having a certain property type. The first step to achieve this is to create a link between $\mathcal{PD}$ and the values from this domain. The set $\mathcal{VL}$ consists of sets of values for a certain domain. In other words, an element from $\mathcal{PD}$ is the *name* of a certain domain and an element of $\mathcal{VL}$ consists of its *values*. The functions $\mathsf{Value} : \mathcal{PD} \rightarrow \mathcal{VL}$ and $\mathsf{VlDom} : \mathcal{VL} \rightarrow \mathcal{PD}$ are used to find the values of a domain or the name of a set of values respectively. For example, the domain *RGB-colors* denoted by d has the values $v = \{\#000000 \ldots \#FFFFFF\}$. More specifically:

$$\begin{aligned} \mathsf{Value}(d) &= v \\ \mathsf{VlDom}(v) &= d \end{aligned}$$

The actual value assignment of a fulfillment (the level of excellence of some artefact in fulfilling a role) having a certain property is given by the function $\mathsf{ValAss} : \mathcal{FL} \times \mathcal{PT} \rightarrow \mathcal{VL}$. In the example, the fact that mug a as an art object r_2 has the color p with value *red* is expressed as follows:

$$\mathsf{ValAss}(\langle a, r_2 \rangle, p) = \#FF0000$$

We have to ensure that the observations on the instance level do not conflict with the typing level. For example, if a fulfillment is said to have a value assignment for a property, then at least one of the roles of this fulfillment must have this property. If f is a fulfillment, p is a property type and v is a value, we have:

Axiom 2 (Conformance)

$$\mathsf{ValAss}(f, p) = v \quad \Longrightarrow \quad p \in \mathsf{Props}(\mathsf{Role}(f)) \ \wedge \ \mathsf{PrDom}(p) = \mathsf{VlDom}(v)$$

In our framework of quality treatment, the axioms of conformance and unique fulfillment express basic properties of wellformed quality models.

3.2 Quality and Desirability

To be able to assess the desirable quality of an artefact for a user, the actual desires of this user must be made explicit. The question is how to do this. One

of the main problems is to choose a domain in which quality is expressed. As an example, it does not make sense to say that the quality of an artefact is 24. The notion of quality is, in that respect, similar to the notion of value as discussed in [1]. A value is an abstract notion and can be used to compare artefacts.

Quality, in the sense of desirability, depends on the desires of actors such as people. Here a distinction must be made between hard and soft desires with regard to artefacts. These can be compared, to some extent, to functional and non-functional requirements or hard goals and soft goals in requirements engineering. In requirements engineering one often tries to make soft goals hard. See for instance [4]. In our approach, goals and requirements are considered to be soft if a human opinion is needed for the value assignment. Otherwise, it is considered to be hard. In other words, hardness or softness of a requirement depends on the way of measurement. The following are examples of hard goals and soft goals:

hard goals: price below €20, contents of 25 liters, made of stainless steel.
soft goals: cheap, pretty, low, hard, strong.

Quality in the sense of desirability depends on the requirements of an individual. More specifically, these requirements have to do with value assignments. The quality of a fulfillment increases if properties have *the right value*. Putting it differently, value assignments are constrained. Consider the following example of a requirement for a fulfillment:

The price in euros may not exceed the price of a given cup.

In this example, *price* is a property type which is expressed in the domain *€'s*. Furthermore, *may not exceed the price of that cup* is a constraint involving an assignment.

Observe that all requirements involve a constraint and a property type. However, some also involve a specific value. We model this as follows. Let $\mathcal{RQ}$ be the set of all requirements and let $\mathcal{CS}$ be the set of all constraints. A requirement has a mandatory property type, a mandatory constraint and an optional expression. Expressions can either be values or fulfillments, as illustrated by the above examples. In terms of our model, we have $\mathcal{EX} \triangleq \mathcal{VL} \cup \mathsf{ValAss}$.

Let $\mathsf{Prop} : \mathcal{RQ} \to \mathcal{PT}$, $\mathsf{Constr} : \mathcal{RQ} \to \mathcal{CS}$, and $\mathsf{Expr} : \mathcal{RQ} \rightarrowtail \mathcal{EX}$. In our framework we use the following shorthand notations:

$$r_1 = \langle p, c, e \rangle \triangleq \mathsf{Prop}(r_1) = p \wedge \mathsf{Constr}(r_1) = c \wedge \mathsf{Expr}(r_1) = e$$
$$r_2 = \langle p, c \rangle \triangleq \mathsf{Prop}(r_2) = p \wedge \mathsf{Constr}(r_2) = c$$

This allows us to write $\langle \mathit{price}, <, €10 \rangle$ for the constraint *the price may not exceed €10* and $\langle \mathit{price}, \min \rangle$ for the constraint *the price must be as low as possible*. Note that a requirement with respect to a fulfillment is of a certain actor. Let $\mathcal{AC}$ be the set of actors and let $\mathsf{Req} : \mathcal{AC} \times \mathcal{FL} \to \wp(\mathcal{RQ})$ yield the requirements of an actor with regard to a fulfillment. For example, we then have:

$$\mathsf{Req}(a, f) = \{r_1, r_2\}$$

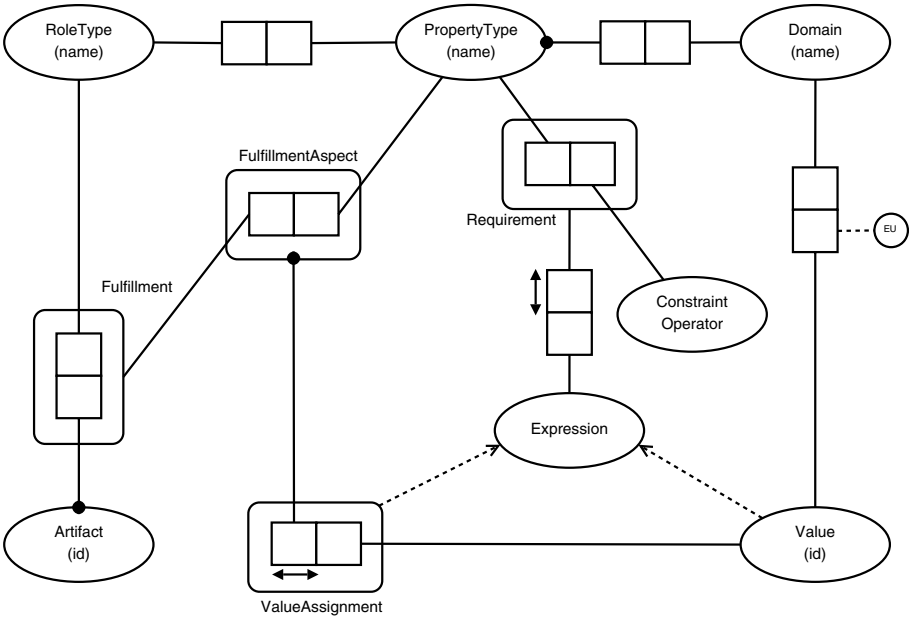


Fig. 2. Object-role model of quality

This expresses that actor a has requirements r_1 and r_2 with regard to fulfillment f. Here we conclude the discussion of our model of quality. The focus of the next section is on operationalization. In order to have a graphical representation of our quality definitions, we present an object-role model in Figure 2. See for example [9] for details on object-role models.

4 Operationalizing Quality

In the previous section we have presented our model for quality which unifies two interpretations of quality. In this section we will shift the focus to operationalizing this model in practice. The ambition of this paper is not to come up with a tool that will determine the quality of a web resource for a searcher. We will merely concentrate on determining which aspects play a role in such a process and how these aspects could be tackled.

4.1 Uncertainty in Quality Assessment

Since a quality assessment system will measure the quality of a resource for a certain actor, the system should be able to deal with uncertainty with regards to the quality. A first kind of uncertainty has to do with the observations and measurements of the system. For example, the fact that a resource has outgoing hyperlinks can be be measured with near 100% certainty. However, the language

of a resource is more difficult to measure. This kind of uncertainty is mentioned in the left part of Figure 3.

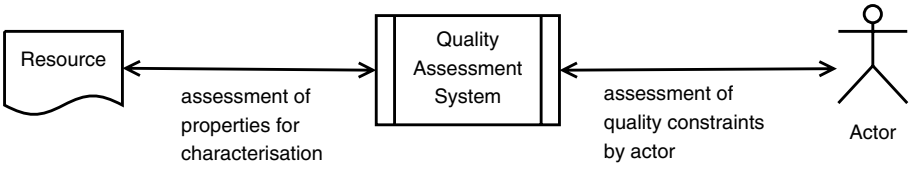


Fig. 3. Uncertainty in quality assessment

In the right part of Figure 3 a second kind of uncertainty is mentioned. This uncertainty deals with the actor for which the quality assessment is made. Consider, for example, the situation in which an actor assesses the quality of resources based on their length, in number of words. Assume that if a text is long then the quality of the resource is considered to be high. Although the number of words is an objective measure, the adjective *long* is subjective and therefore it will be difficult for the quality assessment system to assert with some level of certainty whether a resource is long or not.

4.2 Linguistic Variables

In our quality framework, the concept of a *linguistic variable* is used to describe the fuzzy assessments made by actors [22, 23, 24, 21]. A linguistic variable differs from a numerical variable in that its values are not numbers but words or sentences in some language. For example, the linguistic variable *age* might take *young*, *not young*, *old*, or *not very old* as its values.

More formally, a linguistic variable is defined by a quintuple $\langle \mathcal{X}, T(\mathcal{X}), U, G, M \rangle$ in which $\mathcal{X}$ is the name of the variable. The set $T(\mathcal{X})$ or simply T denotes the termset of $\mathcal{X}$, that is, the set of names of linguistic values being fuzzy variables ranging over U. The rule G is a syntactic rule which usually has the form of a grammar, for generating the names of the values of $\mathcal{X}$. The rule M is a semantic rule for associating with each name X its meaning $M(X)$. The fuzzy restriction is characterized by a membership function $\mu_R : U \rightarrow [0, 1]$ which represents the grade of membership with respect to the fuzzy restriction. For example, for a linguistic variable named *age* we could compute that $\mu_{\text{young}}(50) = 0.35$ which expresses how confident we are that an age of 50 years is considered to be young.

In case of resources on the Web, we need a language for expressing qualities of resources as well as domains in U in which these qualities can be expressed. As an example, assume that *importance* of a resource is the only criterion by which the quality is assessed. Using the terminology of Section 3 we then have:

– We are interested in resources which play the role of *webpage*. Let r be such a resource and let $f \in \mathcal{FL}$ with $\mathsf{Artefact}(f) = r$ and $\mathsf{Role}(f) = \mathit{webpage}$.

– We let $\mathcal{X} \in \mathcal{PT}$ be a property type for *importance*, modelled as a linguistic variable. Furthermore we define domain $\mathsf{PrDom}(\mathcal{X}) = \mathit{PageRank}$ and $\mathsf{Value}(\mathit{PageRank}) = \{\text{low}, \text{medium}, \text{high}\}$ which conforms to the termset of the fuzzy variable $\mathcal{X}$. The universe of discourse is the PageRank which can vary from 0 to 10. The membership function is illustrated by Figure 4.
– We define the requirement $R = \langle \mathcal{X}, \text{high} \rangle$ expressing that the importance property of a resource must be high if it is to be judged as having high quality. Now if $a \in \mathcal{AC}$ is the actor for which the quality assessment takes place, we have $\mathsf{Req}(a, f) = \{R\}$.

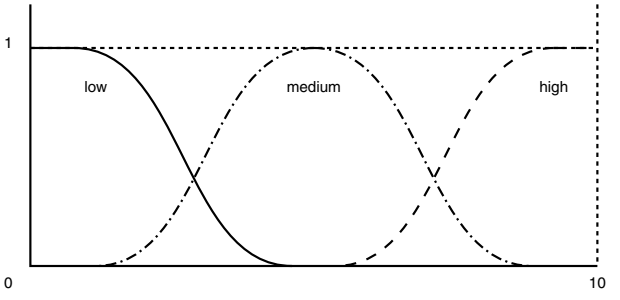


Fig. 4. Membership function for *importance* with PageRank as universe of discourse

The membership function in Figure 4 is read as follows. When the quality of a single resource is to be assessed then the actual pagerank can be used to determine whether the importance is high. An actual pagerank can be extracted from engines such as Google. For example, if the pagerank exceeds 9 then the membership function states that we are 100% sure that this resource will have a high importance. For a pagerank of approximately 7 the membership function states that the degree of membership for *high* and *medium* is approximately the same and equals roughly 0.4. This can be interpreted as being only 40% sure that the quality of the resource is, indeed, high.

4.3 Quality Measurement

Actors use requirements and constraints to determine the quality of an artefact. These requirements are often soft in the sense that they can not be measured directly. Some examples are:

– The resource must have a high pagerank.
– The resource must be recent.

In our approach these soft requirements are translated to concrete statements:

– Data resource having attribution (with value "high" AND of type "pagerank")
– Data resource having attribution (with value "recent" AND of type "modification date")

These statements are meaningful under the assumption that *high* and *recent* are fuzzy values which are mapped to their respective hard domains.

Then we define what it means that we measure some property of an artefact to have a value with some degree of certainty. Measurements depend on the situation in which they are done. Measuring the weight of an artefact depends on the location, for instance on the moon or earth. Furthermore, the measuring device is another cause for concern. For example, one thermometer may be less accurate than another. To model this we use a set $\mathcal{SI}$ describing possible relevant situations and $\mathcal{MD}$ describing measuring devices.

Two additional observations are relevant to our discussion here. Firstly, different kinds of measurements can be done:

1. One can attempt to measure the value of some property of an artefact.
2. One can attempt to verify whether the value associated to a property of an artefact satisfies a condition.

So a measurement results in a measured value or in a boolean. Let $\mathcal{MV}$ be the union of all possbible value domains. A measuring device $R \in \mathcal{MD}$ can now be modeled as a function that maps object-situation combinations into values:

$$R = [\mathcal{AF} \times \mathcal{SI}] \rightarrowtail \mathcal{MV}$$

Secondly, we denote a specific measurement with $\mathsf{M}(a, s, d) = v$ where a is the artefact under consideration, s is the present situation, d is the measuring device and v the observed value.

Example 1. Let c be a car. John is driving down the highway somewhere in Europe. Let s denote his situation, that is his current point in space and time. John drives past a police officer who uses a device d which checks the speed of cars. The observation that John is driving at a speed of $125km/h$ is expressed as $\mathsf{M}(c, s, d) = 125km/h$.

4.4 Accuracy of Measurements

In this section we consider the accuracy of quality measurements. In this context one must realize that measurements are expressed in a domain and that there are standards for expressing them. For example, speed can be measured in terms of kilometers per hour, weight can be measured in terms of grams, distances in terms of meters and so on. Standards bodies, such as a department of weights and measures, govern these standards. By comparing an actual measurement to a standard measurement one obtains a metric for determining the accuracy of a measurement device. We continue the above example as follows.

Example 2. Let d_s be an approved measuring device for speed, that is it measures exactly according the department of weights and measures. This means that a measurement executed with this device is assumed to be 100% correct. If $\mathsf{M}(c, s, d) = \mathsf{M}(c, s, d_s)$ then we know that John was indeed driving exactly at $125km/h$.

In many cases a small deviation of measurement can be allowed when comparing an actual measurement to a standard measurement. To put it differently, when determining whether an actual measurement is equal to a standard measurement one tests if they are sufficiently equal. We define $\stackrel{\circ}{=}$ to be an operator that determines whether a measurement is sufficiently equal to a standard measurement. In other words, a measurement is accurate or sufficiently equal to a standard measurement if $\mathsf{M}(c, s, d) \stackrel{\circ}{=} \mathsf{M}(c, s, d_s)$.

We relate the above discussion to the uncertainty involved in measurements. This uncertainty is caused by the accuracy of measurement devices and the many possible situations in which they are used. The following illustrates this. Let d be a measurement device and d_s be a standard measurement device for the same domain. The measurements of device d can be tested against d_s in many but not neccesarily all situations $S \subseteq \mathcal{SI}$. In our framework, the accuracy of d is defined as the average deviation of that device with respect to the situations in which it is tested:

$$\mathsf{Acc}(d) = \frac{\sum_{s \in S} \mathsf{M}(c, s, d) \stackrel{\circ}{=} \mathsf{M}(c, s, d_s)}{|S|}$$

This accuracy is the basis for defining the measurement uncertainty. That is, if we assert that a property can be measured with a degree of certainty n then we mean that measurements done with this device are correct in $n\%$ of the situations.

4.5 Interpretation of Measurements

The uncertainty involved with interpreting measurements is modeled similarly and makes use of linguistic variables. Let $\langle \mathcal{X}, T(\mathcal{X}), U, G, M \rangle$ be a linguistic variable. In the running example $\mathcal{X}$ represents the variable *volume of a mug* with termset $T(\mathcal{X}) = \{\text{big}, \text{medium}, \text{small}\}$. We interpret the membership degree for these linguistic values as the degree of certainty that we have in this specific interpretation of the actual measurement. Let $\mu_t : U \rightarrow [0 \ldots 1]$ denote the membership degree for the terms in the termset. Consider the following example.

Example 3. Linguistic variable $\mathcal{X}$ denotes volume with termset {small, medium, big}.

Domain U represents volume in *cc*'s. The following is an example of a linear membership function for the linguistic value *big*:

$$\mu_b(u) = \begin{cases} 0 & u \leq 15 \\ \frac{1}{15}u - 1 & \text{otherwise} \\ 1 & u \geq 30 \end{cases}$$

Now we consider the following question. If the volume of a mug is measured to be $25cc$, what are the odds that this mug is considered to be big?

The answer to this question depends on the accuracy of measurements as previously described, but also on the interpretation of the linguistic value *big*. In our approach we interpret the membership degree as certainty of interpretation.

This is based on a conversion of the membership degree function to a probability distribution.

5 Example Quality Assessment System

In this section we will illustrate our quality framework by means of an example quality assessment system. This system is assigned the task to assess the quality of the newsletter of an online news site. The role of this site is informative medium. In terms of our formalism $n \in \mathcal{AF}$ denotes the newsletter and $r \in \mathcal{RO}$ denotes the role played by this site. Furthermore $f = \langle n, r \rangle$ is the fulfillment for this newsletter.

The assessment has to take place for a certain actor $a \in \mathcal{AC}$. Suppose that the actor has three requirements $\mathsf{Req}(f) = \{r_1, r_2, r_3\}$ verbalized as follows:

r_1: Data resource involved in Representation of type "newsletter"
r_2: Data resource of type "Pdf"
r_3: Data resouce having attribution (with value "high" AND of type "importance")

These requirements are embedded in the quality framework as follows:

$r_1 = \langle p_1, c_1, e_1 \rangle$ where p_1 is the property type *representation type*, c_1 is an equality constraint and e_1 is value expression *newsletter*.

$r_2 = \langle p_2, c_2, e_2 \rangle$ where p_2 is the property type *data resource type*, c_2 is an quality constraint and e_2 is value expression *Pdf* which is a data resource type.

$r_3 = \langle p_3, c_3, e_3 \rangle$ where p_3 is the property type *importance*, c_3 is an equality constraint and e_3 value *high*. Note that in this case the system uses a linguistic variable to represent this constraint since *high* is a soft value. The underlying hard domain for importance is chosen to be the pagerank metric.

To be able to make a quality assessment the system uses three measuring devices $d_1, d_2, d_3 \in \mathcal{MD}$, one for each constraint. The three measurements will be done in parallel in one situation $s \in \mathcal{SI}$. Suppose that, based on previous experiences the system knows the following.

d_1 is software tool that is designed with the sole purpose of determining whether a given artefact is a newsletter or not. $\mathsf{Acc}(d_1) = 0.95$ which means that the system is able to correctly judge whether a given artefact is actually a newsletter in 95% of the cases.

d_2 is a tool that checks the data resource type of artefacts. This tool has been trained extensively on all known types and therefore $\mathsf{Acc}(d_2) = 1$.

d_3 is a highly complex tool. It assumes that the PageRank is a good measure for importances of artefacts but knows that this need not always be a 100% correct assumption. Hence suppose $\mathsf{Acc}(d_3) = 0.9$.

The system uses linguistic variables to express the values of constraints. For r_1 and r_2 the membership function is 1 if the condition is met and 0 if it is

not. However, for r_3 the situation is a little more complex. The termset for this variable is $\{low, average, high\}$ and the underlying domain $U = [0 \ldots 10]$ is the domain for expressing pagerank. After careful consideration of the user profile of actor a the system chooses the following membership function for linguistic value *high*:

$$\mu_{\text{high}}(u) = \begin{cases} 0 & 0 \leq u \leq 6 \\ \frac{1}{4}u - 1\frac{1}{2} & 6 < u \leq 10 \end{cases}$$

In this example situation s the system makes the following measurements:

$\mathsf{M}(n, s, d_1) = true$	means that the system suggests that s is indeed a newsletter. So membership degree is 1.
$\mathsf{M}(n, s, d_2) = Pdf$	means that the system suggests that s is a *Pdf* file. So the membership degree is 1.
$\mathsf{M}(n, s, d_3) = 9$	means that the observed pagerank for n is 9. The membership degree then is 0.75.

Now the system computes the certainty of the assertion that n is of high quality to actor a as follows:

- $P_{r_1} = 0.95 \times 1 = 0.95$
- $P_{r_2} = 1 \times 1 = 1$
- $P_{r_3} = 0.9 \times 0.75 = 0.675$

Finally the total quality is the multiplication of these three certainties which results in 0.64. The interpretation is that the system is able to assert with 64% certainty that newsletter n is of high quality to actor a.

We are aware of the fact that the example quality assessment system sketched in this section gives a basic illustration of the possibilities of our quality framework. This is sufficient for the purpose of this paper. More complex case studies will be part of future research.

6 Conclusions and Future Research

The notion of quality plays an important role on the Web, as we rely more and more on information gathered on the Web to perform our day to day tasks. This is why the focus of our project is on aptness-based search rather than topic-based search. Not only topic, but other factors should be taken into account as well when searching the Web. In the current paper we have focused on aptness of Web resources in general, and on the notion of quality in particular.

The paper gives an overview of how the quality notion is used in different fields. Also, we have presented a model which explains what quality is and how the quality of an asset for a certain actor can be measured. This model is sufficiently expressive but still needs more work. The fuzzy-logic approach using linguistic variables provides a straightforward way to deal with quality on the Web. We elaborated our approach in an example quality assessment system.

In future research we aim at more complex case studies. On the one hand, our quality framework needs a more extensive validation. On the other hand, we plan to apply the framework in different domains, such as scientific search, medical information management, geographic applications, and bioinformatics. In the area of technology we see the application of XML-based quality management as a challenge to overcome the heterogeneity on the Web.

References

1. P. van Bommel, B. van Gils, H.A. Proper, M. van Vliet, and Th.P. van der Weide. The information market – its basic concepts and its challenges. In *Web information systems engineering (WISE), New York*, volume 3806 of *Lecture Notes in Computer Science*, pages 577–583. Springer-Verlag, November 2005.
2. F.P. Brooks Jr. No silver bullet: essence and accidents of software engineering. *IEEE Computer*, 20(4):10–19, April 1987.
3. G.B. Davis and M.H. Olson. *Management Information Systems: Conceptual Foundations, Structure and Development*. McGraw–Hill, New York, USA, 1985.
4. P Donzelli and B. Bresciani. Improving requirements engineering by quality modelling – a quality-based requirements engineering framework. *Journal of Research and Practice in Information Technology*, 36(4), November 2004.
5. Michael Gertz, M. Tamer Özsu, Gunter Saake, and Kai-Uwe Sattler. Report on the dagstuhl seminar: data quality on the web. *SIGMOD Rec.*, 33(1):127–132, 2004.
6. T. Gilb. *Principles of software engineering management*. Addison Wesley, Reading, Massachusetts, USA, 1988.
7. B. van Gils, H.A. (Erik) Proper, and P. van Bommel. A conceptual model of information supply. *Data & Knowledge Engineering*, 51:189–222, 2004.
8. B. van Gils, H.A. (Erik) Proper, P. van Bommel, and Th.P. van der Weide. Transformations in information supply. In *Workshop of the 16th Conference on Advanced Information Systems Engineering (CAiSE), Riga*, pages 60–78, June 2004.
9. T.A. Halpin. *Information Modeling and Relational Databases, From Conceptual Analysis to Logical Design*. Morgan Kaufmann, San Mateo, California, USA, 2001.
10. M. Harrison. *Principles of operations management*. Pitman, London, United Kingdom, EU, 1996.
11. V. Lala, A. Arnold, S.G. Sutten, and L. Guan. The impact of relative information quality of e-commerce assurance seals on internet purchasing behavior. *International Journal of Accounting Information Systems*, 3(4):237–253, December 2002.
12. K.C. Laudon and J.P. Laudon. *Management Information Systems, International Edition*. Prentice-Hall, Englewood Cliffs, New Jersey, USA, 1996.
13. Steve McConnell. *Code complete, a practical handbook of software construction*. Microsoft Press, Redmond, Washington, USA, 2 edition, 2004.
14. Ken Orr. Data quality and systems theory. *Commun. ACM*, 41(2):66–71, 1998.
15. G. John van der Pijl. Quality of information and the goals and targets of the organization. In *Computer personnel research conference on reinventing IS, Alexandria, USA*, pages 165–172, 1994.
16. C.J. van Rijsbergen. *Information Retrieval*. Butterworths, London, United Kingdom, EU, 1975.
17. G.E Salton and M.J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, New York, New York, USA, 1983.

18. I. Sommerville. *Software Engineering*. Addison Wesley, Reading, Massachusetts, USA, 1989.
19. E. Turban, J. Lee, D. King, and H.M. Chung. *Electronic Commerce, a managerial perspective*. Prentice-Hall, Englewood Cliffs, New Jersey, USA, 1999.
20. C. Vishik and A.B. Whinston. Knowledge sharing, quality, and intermediation. In *Conference on work activities coordination and collaboration, San Francisco*, pages 157–166, 1999.
21. L. Zadeh. From computing with numbers to computing with words - from manipulation of measurements to manipulation of perceptions. *International Journal of Applied Mathematics and Computer Science*, 12:307–324, 2002.
22. L.A. Zadeh. The concept of a linguistic variable and its application to approximate reasoning – i. *Information Science*, 8:199–249, 1975.
23. L.A. Zadeh. The concept of a linguistic variable and its application to approximate reasoning – ii. *Information Science*, 8:301–357, 1975.
24. L.A. Zadeh. The concept of a linguistic variable and its application to approximate reasoning – iii. *Information Science*, 9:301–357, 1975.

Bid-Based Approach for Pricing Web Service

Inbal Yahav[1], Avigdor Gal[1], and Nathan Larson[2]

[1] Technion - Israel Institute of Technology
Haifa 32000, Israel
[2] University of Maryland, College Park
MD 20742 U.S.A.

Abstract. We consider a problem of Web service resource allocation in an economic setting. We assume that different requestors have different valuations for services and a deadline for executing a service, after which it is no longer required. We formally show an optimal offline allocation that maximizes the total welfare, denoted as the total benefit of the requestors. We then propose a bid-based approach to resource allocation and pricing for Web services. Using a detailed simulation, we analyze its behavior and performance compared to other known algorithms. We empirically show that flexibility in service price benefits both the provider in terms of profit and the requestors in terms of welfare.

Our problem motivation stems from the expanding use of Service-Oriented Architecture (SOA) for outsourcing enterprize activities. While the most common method for pricing a Web service nowadays is a fixed-price policy (with a price of 0 in many cases), A Service-Oriented Architecture will increasingly generate competition among providers, underlying the importance of finding methodologies for pricing Web service execution.

1 Introduction

A Web service (WS) is an autonomous unit of application logic, hosted by a service provider, which provides either some business functionality or database information to service customers through the World Wide Web. A Service-Oriented Architecture enables organizations to outsource software activities to individual Web service providers, some of which may provide similar functionality. Like any other profit making organization, Web service providers use limited resources such as CPU and bandwidth to provide services to requestors in return for benefits. A Service-Oriented Architecture may, therefore, generate *competition* among providers, underlying the importance of finding methodologies for pricing a Web service.

The most common method for pricing a Web service nowadays is a fixed-price policy (possibly at a 0 cost). Less common methods deal with dynamic pricing as price-negotiation, logrolling - negotiation on the trade-off between quality of service (QoS) and cost of service (CoS) [1] or usage-based-cost [2]. Yet, in all these cases the provider determines the service cost. Service scheduling methods, resulting from limited resource availability, are typically first-in-first-out (FIFO) or a round-robin (RR) policy. Both sustain fairness, yet fail to support varying client priority conditions.

In this paper we consider Service-Oriented Architecture in an economic setting. We assume that different requestors have different valuations for services. We also assume

R. Meersman, Z. Tari et al. (Eds.): OTM 2006, LNCS 4275, pp. 360–376, 2006.

a hard deadline model where each requestor has a deadline for executing a service after which it is no longer required. We refer to the evaluation of value and deadline of a request as a *private* knowledge, *i.e.*, known to the requestor alone. We study two types of problems: profit maximization and welfare maximization. While in a profit maximization model the provider goal is to maximize its own benefit without considering clients utility, a welfare problem deals with the overall benefit of both providers and requestors. The latter can be measured as the fraction of the processed job requests (*throughput*) or their total valuation (*weighted throughput*).

We model the problem of pricing Web services as an online scheduling problem. Requestors with different urgencies (deadlines) and priorities request services over time. Due to resource constraints the provider cannot process all the requests simultaneously. It therefore has to prioritize them. We introduce a bid-based approach for pricing Web service execution and compare it to a fixed-price model under an economic setting. We provide two main results:

- A bid-based method that heuristically optimizes the *weighted throughput*, yielding similar performance as the best-performer algorithm, known so far in the literature (RMix, [3]). The latter assumes public knowledge of request evaluation.
- In a server competition setting, customers have an incentive to choose the most price-wise flexible server. Due to deadline constraints, high-value requestors are willing to bid a higher value to the bid-based provider then the price required by the fixed-price service. The result is higher server profit and higher throughput and weighted throughput to the bid-based service provider.

1.1 Related Work

We discuss related work in two dimensions. We first summarize the research in the area of pricing Web services. Then, following our modeling as a job scheduling problem, we discuss online scheduling research in general and in economic setting in particular.

Pricing Web Services. Recently, grid computing is emerging as a new direction of Internet computing with the combined power of individual computers and the Internet. In grid computing, a QoS requirement involves computing constraints as CPU and bandwidth. A Web service is subject to the same QoS requirement: optimizing request allocation under limited resources [4].

QoS research on economics-based network resource allocation mechanisms supports usage-based Web service pricing [2,4,5,6], where each customer pays relative to the percentage of provided service. The main idea is to use market mechanisms to suppress low-value data traffic. One common method is dynamic pricing, denoted as GSW priority pricing model, for network resource allocation [5]. The queuing schedule method that is derived from GSW is FIFO or RR (Round Robin).

Another method for pricing a Web service, called *logrolling*, is suggested in [1,7]. The logrolling method supports sophisticated business models, in which the Web service provider can provide a list of trade-off alternatives between the QoS they offer and the CoS (Cost of Service) they use. In some models [8], both parties have to evaluate the list of QoS and CoS alternatives for obtaining a combination that suits their needs.

This commercial negotiation is generally concerned with buying and selling of goods or services and also with the associated topics of quality, specification, delivery, and service [9].

In this paper we refer to each Web service as an atomic unit that can either be performed as a whole or not at all. Therefore, usage-base models do not apply in our case. We introduce a bid-based approach for pricing Web service and compare it to a fixed-price method.

Online Scheduling. The weighted throughput of an online scheduler has many variations and implementations in the literature. For a non-preemptive model of unit-length tasks on a single processor, some restricted versions of the problem are proposed and studied [3,10,11,12,13]. The performance of an online scheduling algorithm is measured with a competitive ratio, first defined by Graham [14], where the algorithm is compared against the optimal algorithm.

In a non-strategic setting, a naïve greedy algorithm that schedules the heaviest task or the earliest deadline first is known to be 2-competitive [11,15]. For the deterministic case Hajek [11] proves a lower bound of $\theta \approx 1.618$. The RMix randomized algorithm was introduced with a lower bound of $e/(e-1) \approx 1.582$ [3]. Both algorithms assume public knowledge of job evaluation.

The online scheduling with private information is introduced and motivated in [16,17]. In this setting each task is owned by a separate, self-interested agent. The agent in this setting can manipulate the algorithm by artificially changing its private parameters. To the best of our knowledge, there is no work done for the simple one-unit tasks in the strategic case, which we handled in our case.

In this paper we introduce a bid-based algorithm Higher-Bid-First (HBF) that is subject to private knowledge. We empirically show that the HBF algorithms matches the lower bound of the random algorithm RMix [3].

The remaining of the paper is organized as followed. in Section 2 we provide a formal definition of the online scheduling problem in an economic setting. In Section 3 we study the online and offline bounds of the throughput and weighted throughput maximization problem. In Section 4 we introduce a bid-based approach for pricing a Web service. Additionally, we describe a scheduler that maximizes the total profit in a fixed-price model. We study the empirical performance of the two algorithms separately and in competition in Section 5. We conclude in Section 6.

2 Model and Problem Definition

We consider a server serving k requests at most at each time slot. We assume an identical expected processing time for each request. A server needs to *schedule* requests to be served at each time slot t. For ease of exposition, we assume from now on a capacity of 1 and expected processing time of 1 time slot.

A scheduling problem of size n is an assignment problem of n independent requests to units of processing time under pre-defined constraints such as CPU and processing rate [18]. A scheduler is called an *online scheduler* if it has no prior knowledge of

future tasks, whereas an *offline scheduler* or a *clairvoyant scheduler* has full a-priori information of the arrival of request. A performance of a scheduler is measured by either *throughput* or *weighted throughput*.

Traditionally, scheduling mechanisms implicitly assume public knowledge of request evaluation. However, with the emerge of the Internet as a major platform of computing, this assumption can no longer be taken for granted. We consider a Service-Oriented Architecture (SOA) that is designed to combine services of different companies with possibly contradictory needs and goals. Therefore, each company will likely try to manipulate the mechanism to increase its own benefit. We model this situation using *autonomous customers*. Our goal is to design a Web service scheduling mechanism under the conjecture of private information, which performs at least as good as algorithms that utilize public information.

We use T = $\{t_i\}$ to denote the set of service requests, with t_i being the i^{th} arriving request. We identify each request by the triple (a_i, d_i, v_i), representing respectively its request time, deadline, after which it is not longer needed, and value. We define a *span* (s_i for the i^{th} job) of a request to be the difference between the deadline and the request time. Time is divided into discrete units, represented by U = $\{1, 2, 3, \ldots\}$. We use v_{ij} to represent the value gained from processing request t_i at time j, where:

$$v_{ij} = \begin{cases} v_i \text{ if } a_i \leq j \leq d_i \\ 0 \text{ otherwise} \end{cases}$$

We define x_{ij} to be a Boolean variable to denote whether t_i was processed at time j.

We study the scheduling mechanism of Web service requests from two different aspects. The first aspect examines the general *welfare* of the Web service scheduling algorithm. That is, checking the fraction of processed Web service requests (*throughput*), given by Eq. 1, and the value of processed requests *weighted throughput*, given by Eq. 2. Our goal is to maximize one or two of the measurements.

$$Throughput = \max \sum_{t_i \in T, j \in U} x_{ij} \tag{1}$$

$$WeightedThroughput = \max \sum_{t_i \in T, j \in U} v_{ij} x_{ij} \tag{2}$$

The second aspect is the Web service provider *profit*. Formally, assume each customer i pays p_i if $\exists j \in U | x_{ij} = 1$, *i.e.*, t_i is processed. The provider's total profit maximization problem is therefore given by:

$$Profit = \max \sum_{t_i \in T, j \in U} p_i x_{ij} \tag{3}$$

3 Upper Bounds

In assessing the performance of a scheduling algorithm, scheduling the processing of Web service requests, we would like to set upper bounds on the best attainable performance by any allocation scheme under the constraints of online scheduling and private information.

The traditional approach to examine a performance of an online scheduling algorithm is by formally comparing the worst case of the ratio between the cost incurred by an online algorithm and the base case cost. This approach is called *competitive ratio*.

However, since our results are empirical, a formal optimal solution for the offline problem is required for a comparison purpose. In Section 3.1 we use a graph representation to solve the offline problems of throughput and weighted throughput maximization. We show that the given problem does not have an optimal solution in an online setting, and therefore there is a need to apply a heuristic solution becomes apparent. Additionally, we prove competitive ratio of 1 for the online throughput maximization problem, that is, showing it to be optimal.

3.1 Studying the Offline Problem

Since each request requires exactly one unit of processing time, we can restate the requests scheduling problem using a bipartite graph. Specifically, consider the set of requests to be one set of nodes and the set of time slots to be another. Since a schedule is a one-to-one mapping between requests and time slots, we may represent it as a bipartite graph with an edge between request t and time slot i if t can be scheduled at i (that is, if i is later than the request time and prior to its deadline). The weight on this edge equals to the requestor's value v_i.

The bipartite graph representation of the scheduling problem enables us to restate the offline throughput optimization problem (Eq. 1) as a maximum-matching problem, to be solved in $O(|V||E|)$ time [19]:

$$\begin{aligned}
&\max \sum_{t_i \in T, j \in U} x_{ij} \\
&\text{s.t.} \\
&\sum_{t_i \in T} x_{ij} \leq 1, \forall j \in U \\
&\sum_{j \in U} x_{ij} \leq 1, \forall t_i \in T \\
&x_{ij} \in \{0,1\}
\end{aligned} \tag{4}$$

The weighted-throughput optimization problem is equivalent to determining the bipartite matching with the greatest sum of edge weights (maximum-weighted-matching), with complexity of $O(|V|^{2.5} log(|E||V|))$ [20]:

$$\begin{aligned}
&\max \sum_{t_i \in T, j \in U} v_i x_{ij} \\
&\text{s.t.} \\
&\sum_{t_i \in T} x_{ij} \leq 1, \forall j \in U \\
&\sum_{j \in U} x_{ij} \leq 1, \forall t_i \in T \\
&x_{ij} \in \{0,1\}
\end{aligned} \tag{5}$$

Table 1. Example scenario of Web service requests

Request	Request time	Deadline	Value
a	1	2	7
b	1	1	3
c	2	3	8
d	3	3	2

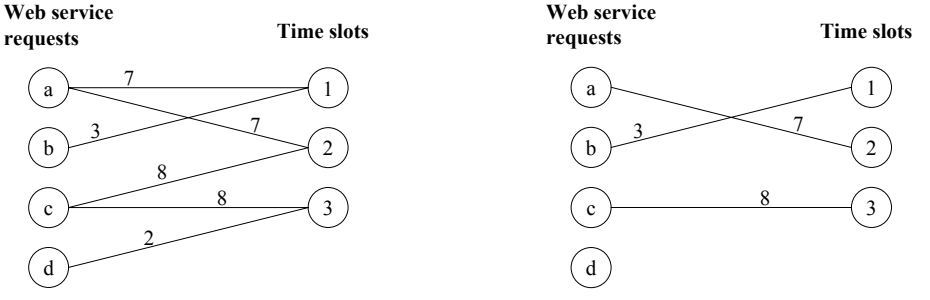


Fig. 1. Left: graph representation of the example scenario; Right: maximum-weighted-matching on the graph

For example, consider the Web service requests scenario presented in Table 1. The corresponding graph representation and maximum-weighted-matching solution in given in Figure 1.

Theorem 1

$$\max \sum_{t_i \in T, j \in U} v_{ij} x_{ij} \rightarrow \max \sum_{t_i \in T, j \in U} x_{ij}$$

Theorem 1 gives the existence of an offline algorithm that maximizes both the throughput and the weighted throughput.

We refrain from presenting the proof of this and other theorems in this paper for space consideration. All proofs are given in [21].

3.2 Studying the Online Problem

An online Early-Deadline-First (EDF) scheduler is a greedy scheduler that processes at each time slot the most urgent request in the queue. Formally, given a set X of pending requests that arrive (a_i) before time slot t ($\forall t_i \in X : a_i \leq t$), then the *earliest* request t_0 is defined by: $\forall t_i \neq t_0 \in X : d_0 < d_i$ or $[d_0 = di$ and $a_0 < a_i]$. The earliest-deadline-request is uniquely defined.

Theorem 2. *EDF under hard deadline model has a competitive-ratio of 1 in terms of throughput.*

Theorem 3. *For each online scheduling policy there exists a request-arrival scenario for which its weighted throughput is not optimal.*

Table 2. Upper bounds summary

	Throughput	**Weighted Throughput**
Offline	Max-Weighted-Matching	Max-Weighted-Matching Max-Matching
Online	EDF	*Does not Exist*

The heuristic online scheduling algorithm RMix, presented in [3] is claimed to have the best weighted throughput competitive ratio ($e/(e-1) \approx 1.582$-competitive), under public information assumption.

3.3 Upper Bounds Summary

Table 2 summarizes the bounds for the welfare problems (throughput and weighted throughput) in offline and online settings. Based on Theorem 3 the weighted throughput online maximization problem can only be solved heuristically. In the next section we describe our heuristic scheduling, based on bidding.

4 Online Scheduling with Private Information

We consider the online scheduling in an economic setting, where each service request is owned by a unique, independent customer. We assume that customers are self-interested and therefore reveal only information that benefits them. We examine two types of mechanisms: the first is a bidding mechanism that prioritizes jobs in decreasing order of bids; the second is a variation on Early-Deadline-First [22] that prioritizes job requests according to an increased deadline as reported by the customer (*strategic deadline*).

Our setting includes one server and N independent unit-length service requests that arrive online. We assume that the distribution of request deadline and value is a public knowledge, whereas the evaluation of a deadline and value of a specific request is known only to the customer.

4.1 Bid-Based Algorithm HBF

We model our mechanism as follows (see Figure 2). Each Web service customer presents a request to the broker, along with a one dimension bid, defined as $b_i = F : (s_i, v_i) \rightarrow R^+$. The broker then forwards it to a single service provider. The provider processes the requests, giving higher priority to higher bids (Higher-Bid-First policy). Over-due requests leave the queue (or alternatively withdrawn by their owners. In addition, the broker publishes a success rate of serviced requests, based on request bids and spans, allowing the customers to maximize their personal gain, given by:

$$Gain = \begin{cases} v_i - b_i, & \text{if } \exists j : x_{ij} = 1 \\ 0, & \text{otherwise} \end{cases}$$

The customer's goal in the mechanism is to maximize its expected utility, given by its personal gain multiplied by the probability of getting processed. Algorithm 1 provides this decision making process.

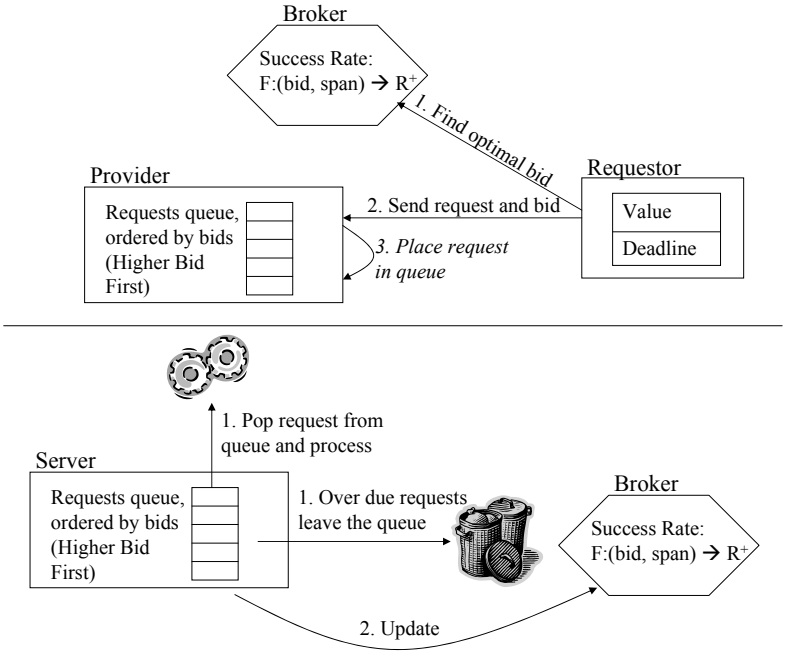


Fig. 2. Schematic representation of the bidding mechanism. Top: Sending bids to the server; Bottom: Tasks processing.

Algorithm 1. Customer Decision Making Process

On New Service Arrival:
Input: $\{v_i, a_i, d_i\}$
Output: $\{b_i\}$
$u_i = 0$
$b_i = 0$
$s_i = d_i - a_i$
for all $b \in$ [Bids Range] $\leq v_i$ **do**
  $u \leftarrow (v_i - b) * P(\text{getting processed } |\{s_i, b\})$
  if $u > u_i$ **then**
    $b_i \leftarrow b$
    $u_i \leftarrow u$
  end if
end for

4.2 Early Strategic Deadline First Algorithm (EsDF)

For comparison we now present an online scheduler that prices its services using a fixed-price policy. It is easy to show that maximum throughput entails maximum profit in a fixed-price model. We therefore look for a scheduler that maximizes throughput. We subject this alternative scheduler to the self-interested customers assumption as well.

In Theorem 2 we showed that the EDF algorithm maximizes throughput of an online scheduling. Yet, EDF assumes that customers do not manipulate the server by providing a false deadline. Since the scheduler prioritizes service requests by their deadline, giving higher priority to more urgent requests, a rational client has an incentive to declare a deadline that is earlier then its actual deadline. In fact, the incentive is to declare a deadline that equals its time of request.

To avoid such a scenario we define the following variation on EDF, called Early-Strategic-Deadline-First (EsDF). Customers present requests to the server along with a *strategic deadline* that maximizes their probability of getting processed. The server then processes the requests, giving higher priority to more urgent-declared jobs. Overdue requests, as declared by the customer deadline, are removed from the queue. In addition, the server publishes a success rate of processed requests, as a function of the declared span, allowing the customers to base their strategic deadline on processing history.

Intuitively, if a customer declares a deadline that is smaller than its real deadline evaluation, it gets higher priority in the queue, but at the same time it reaches its declared deadline earlier. Therefore it might leave the queue without being processed before its actual deadline is due. Obviously, customers have little incentive to declare a deadline that is greater than their true deadline. We keep the customer and server algorithms as before.

5 Experiments and Results

In this section we report our empirical results. We first describe in Section 5.1 our simulation model, along with a detailed description of the experiment setup. We then present the empirical behavior of HBF and EsDF policies and analyze their performance in Section 5.2.

5.1 Simulation Setting

We implemented five online scheduling algorithms. The first two are HBF and EsDF that were described in Section 4. The other three serve for comparison. We use EDF, the optimal algorithm in terms of throughput, the randomized algorithm RMix, presented in [3], and a FIFO algorithm. Additionally, we implemented a bipartite graph generator that creates offline schedules. We used a VS.Net 2004 platform for implementing the simulator and ILOG version 3.6.1 to solve the offline optimal solution.

Model. We simulate online scenarios to examine the performance of HBF and EsDF algorithms. The HBF server is mainly composed of the components described below. EsDF server is implemented similarly.

Requests. Each request in our mechanism has a private valuation and a request completion deadline. A request that misses its deadline is dropped from the system. We assume an identical processing time (1-unit) for all the requests. When a new request is created, the customer calculates the best bid strategy according to a public table of past success rates for each combination of span and bid.

Table 3. Summary of customer setting

Parameter	Distribution	Range	Notes
Value (v_i)	Uniform	[1, 2 ... 10]	
Span (s_i)	Uniform	[0, 1 ... 8]	
Strategic Bid (b_i)	-	$[0, 10] \leq v_i$	99% of the customers
Random Bid (b_i)	Uniform	[0, 10]	1% of the customers

Servers. Customers submit bids to a broker, which then forwards them to the server. The server places the requests in its queue, ranking them by their bids. Once in every y units of time, the server publishes a statistics table of process success rates at its queue, allowing the customers to calculate the best utility, based on history. The calculation of the success rate for window $[t_x, t_{x+y}]$ is given by:
$S[x,x+y][b][s]$ - The number of requests that arrived in time window $[t_x, t_{x+y}]$, with a span of s and a bid of b, that were successfully processed.
$T[x,x+y][b][s]$ - The total number of requests that arrived in time window $[t_x, t_{x+y}]$, with a span of s and a bid of b.
In each re-computation, the new probability table uses a smoothing function combining new and old success rates, as followed:

$$\begin{aligned} P_{t_{x+y}} \ & (\text{getting processed} \mid \text{bid, span}) = \\ & \alpha * (S[x,x+y][b][s]/T[x,x+y][b][s]) \\ & (1-\alpha) * P_{t_x}(\text{getting processed} \mid \text{bid, span}) \end{aligned}$$

Different server configurations are distinguished by server service rate k, statistics table publication frequency y and the smoothing coefficient α.

Experiment Setting. We have run experiments with HBF and EsDF algorithms separately and in competition under various settings to evaluate their behavior and performance.

We model the arrival rate of jobs as a Poisson process with an intensity parameter λ. We assume a uniform distribution (uniform [0, 10]) for values and spans. Bidding are modeled as discrete values in the range [0, 10]. We allow a small percentage (1%) of random bids to simulate 'noise' in the system. Table 3 summarizes the customer setting in our experiments.

A service rate of a server is set to be $k = 1$, meaning it can process one request at most at any time slot. Every $y = 1000$ time slots the server refreshes its success rate table. We use $\alpha = 0.9$ for the smoothing function. The smoothing function by its nature gives higher priority to recent history. Table 4 summarizes the server setting in our experiments.

Evaluation. We evaluate the behavior of HBF using the following metrics:

– Stability: We examine the smoothness of the success rate table over time. We say that the algorithm is *stable* if for each combination of span and bid the Δ between two sequential success rates is negligible:

$$\begin{aligned} & \forall t \in U, s \in Span, b \in Bid : \\ & |P_t(\text{getting processed} \mid s,b) - P_{t-y}(\text{getting processed} \mid s,b)| \rightarrow 0 \end{aligned} \quad (6)$$

Table 4. Summary of server setting

Parameter	Value
Service Rate (k)	1 request per time slot
Smoothing coefficient (α)	0.9
Success table refresh rate (y)	1000 sec

– Convergence: We study the empirical equilibrium of the algorithm in the long run. We say that the algorithm *converges* if after finite initialization time j the changes in the success rate table are negligible:

$$\exists u \in U \forall j > u \forall s \in Span, b \in Bid :$$
$$\lim_{t \to \infty} |P_t(\text{getting processed} \mid s,b) - P_j(\text{getting processed}|s,b) \mid \to 0 \qquad (7)$$

We evaluate the correctness of EsDF by studying the *strategic* span, declared by customers, compared to request *true* span.

Next, we evaluate the performance of HBF by studying its throughput (see Eq. 1) compared to the optimal algorithm EDF. We study its weighted throughput (see Eq. 2) and compare HBF to EDF, FIFO, RMix and the optimal offline solutions.

In the competition setting, we run EsDF with fixed price against HBF. We study the type, defined as $type_i = \{v_i, s_i\}$ and number of requestors that *choose* each server. We examine additionally the profit gained by each server. Formally, we define c_{ik} to be a boolean variable that denotes whether customer i chooses server k; x_{ijk} is a boolean variable that denotes whether customer i was processed at time slot j by server k; and p_i is the amount paid by customer i. Then, the number of requestors that choose server k is given by Eq. 8 and its profit by Eq. 9.

$$\sum_{t_i \in T} c_{ik} \qquad (8)$$

$$\sum_{t_i \in T} c_{ik} x_{ijk} p_i \qquad (9)$$

5.2 Results

HBF. We study the behavior of HBF as described in Section 5.1. First, we examine the *stability* of the algorithm. Figure 3 presents the change in the success rate table as a function of span and bid. For readability reasons, we chose to present representative bids-curve only. We show that $\forall s \in Span, b \in Bid : \max |P_t(\text{getting processed} \mid s,b) - P_{\text{getting processed}|t-y}(s,b)| < 0.1$, meaning that the success rate table changes smoothly and the algorithm is stable. To further examine the stability of HBF algorithm we study the standard deviation of the success rate table as a function of span for each bid. Figure 4 presents representative curves of the eventual decrease deviation after an initialization period of $20,000$ time slots, giving that the algorithm is stable.

Next, we examine the algorithm in terms of *convergence*. Figure 5 presents for each bid, the $\max \Delta_p(t) = \max(Prob_n(\text{getting processed} \mid s,b) - Prob_t(\text{getting processed} \mid$

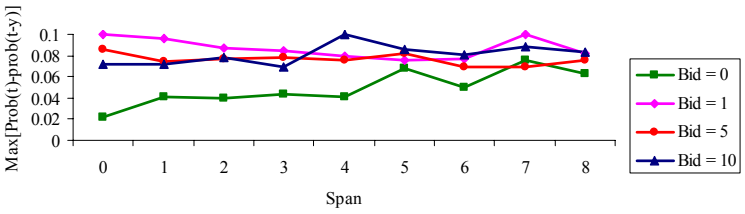


Fig. 3. Stability of HBF success rate table

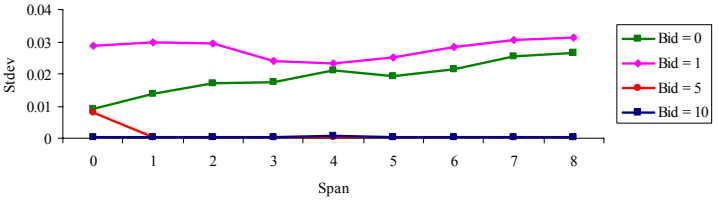


Fig. 4. Standard deviation of the success rate table

$s,b)$) given by some span s as a function of time, where n is the last time slot. It is shown that the algorithm converges after approximately $20,000$ time slots. The empirical stability and convergence of HBF allow us to study the performance of the algorithm, *i.e.*, throughput and weighted throughput, in a short finite time and apply these results to the long run behavior of the algorithm.

Finally, we examine the uniqueness of the algorithm empirical equilibrium. In Figure 6 we show the throughput of HBF under different success rate table initializations as a function of value and span, where at each run we give higher priority to different types of requests: in the *symmetric* case the initial probability for each combination of bid and span is exactly 1; in the *High Bid* initialization case, the initial probability for each span and bid, where bid is greater than the average value, is a random value in $[\frac{2}{3}, 1]$. The initial probability for each span and bid, where bid is lower than or equal to the average value, is a random value in $[0, \frac{1}{3}]$; The *Low Bid*, *Not Urgent Tasks* and *Urgent Tasks* cases are initialized similarly. The results for each initialization are similar, indicating that the algorithm converges to a single equilibrium.

After studying the behavior of our mechanism we examine its performance with respect to *throughput* and *weighted throughput*. We test our method under different request loads, comparing it to the bounds described in Section 3. In Figure 7 we compare HBF throughput to the optimal algorithm EDF as a function of requests load. We show that the throughput of HBF is larger than 95% of the optimal solution. In Figure 8 we compare HBF weighted throughput to the optimal offline solution as a function of requests load. We show a performance ratio of more than 92%. For both experiments, we can see that worst performance is achieved whenever the arrival rate is approximately the same as the server service rate. Which, in our case, equals 1. For much smaller arrival rates the light load allows the server to process all tasks, independently on the scheduling model, processing each task on its release time. At the other extreme, when

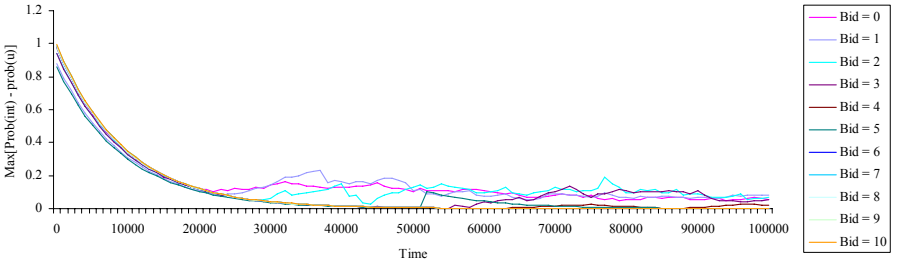


Fig. 5. $\max \Delta_p(t)$

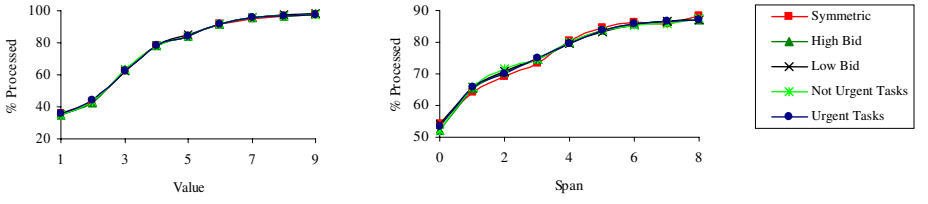


Fig. 6. Empirical equilibrium under Different Success Rate Initialization

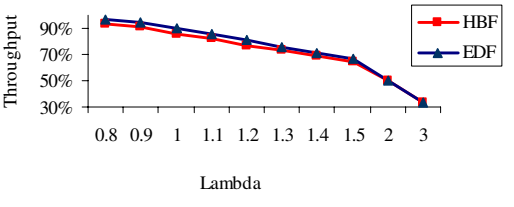


Fig. 7. Throughput comparison

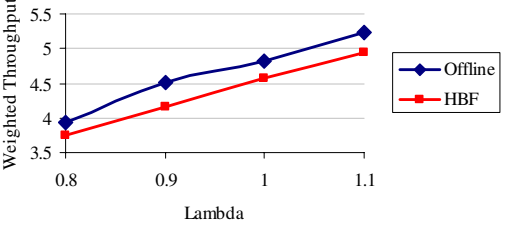


Fig. 8. Weighted throughput comparison

the load is very high the queue length is bigger then 1 and the server always have a task in the queue to process. The scheduling therefore becomes most interesting when delaying a task may cause loosing it before another task arrives to the queue. This occurs when the tasks arrival rate is close to the server service rate.

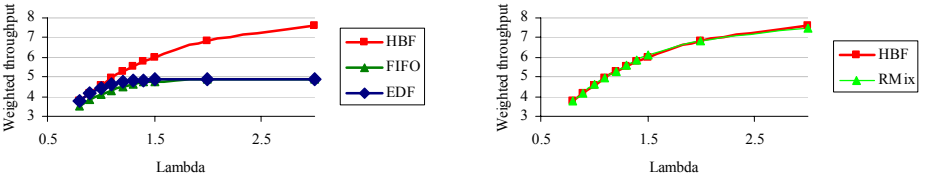


Fig. 9. Throughput comparison

Next, we measure the *weighted throughput* of our bidding by comparing it to known mechanisms in the literature. In Figure 9 we present the performance of HBF compared to EDF and to FIFO, both are known to be 2-competitive [11]. We show that the weighted throughput of HBF increases with load whereas FIFO and EDF converge to the average value, 5 (See Table 3). We also present a comparison to the RMix algorithm, that is known to be 1.582-competitive. We show that in spite of the restriction of private value and deadline, the weighted throughput gained by HBF is approximately 99% of RMix's weighted throughput.

EsDF. In the first set of EsDF experiments (Figure 10) we examine the behavior of requestors by studying the ratio between tasks true deadline and declared deadline under different loads. We show that the average declared deadline in monotonic in deadline. The monotonicity implies that the EsDF scheduler performance in terms of throughput should be very similar to the optimal EDF scheduler. Next, we compare the two schedulers throughput to strengthen our conclusion. Figure 11 shows that EsDF processes more than 98% requests compared to EDF, for all λ (arrival rate) values.

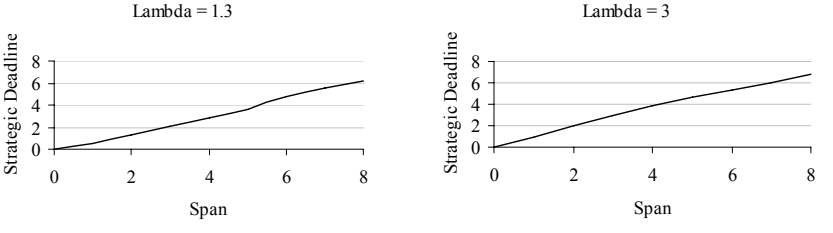


Fig. 10. Average strategic span as a function of the true span

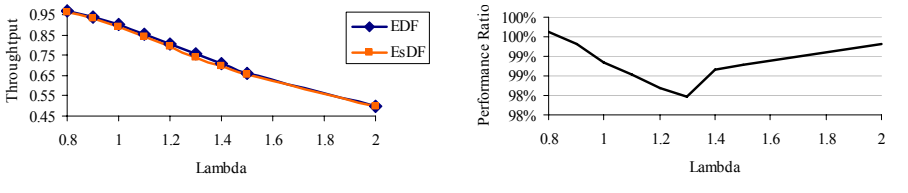


Fig. 11. Throughput: EsDF Vs. EDF. Left: throughput comparison. Right: ratio between EsDF throughput and EDF.

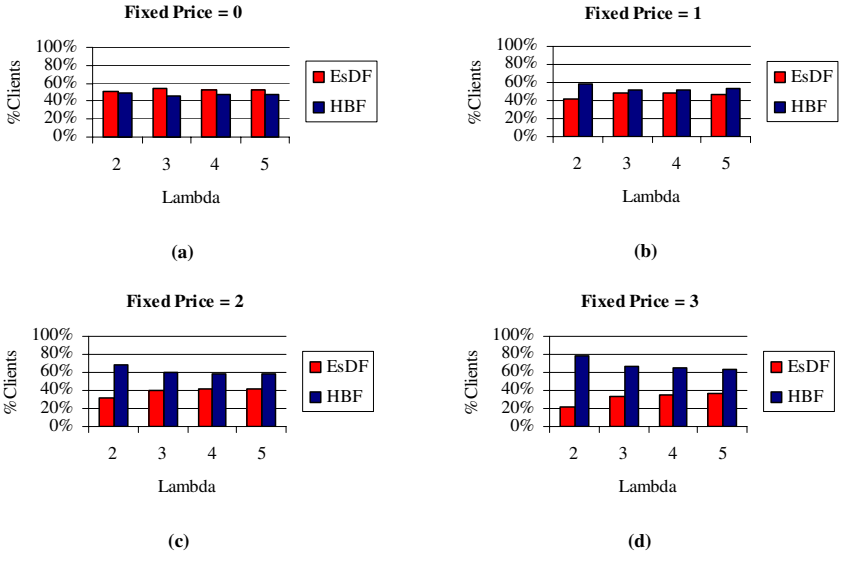


Fig. 12. Fraction of customers that chooses each server as a function of arrival rate

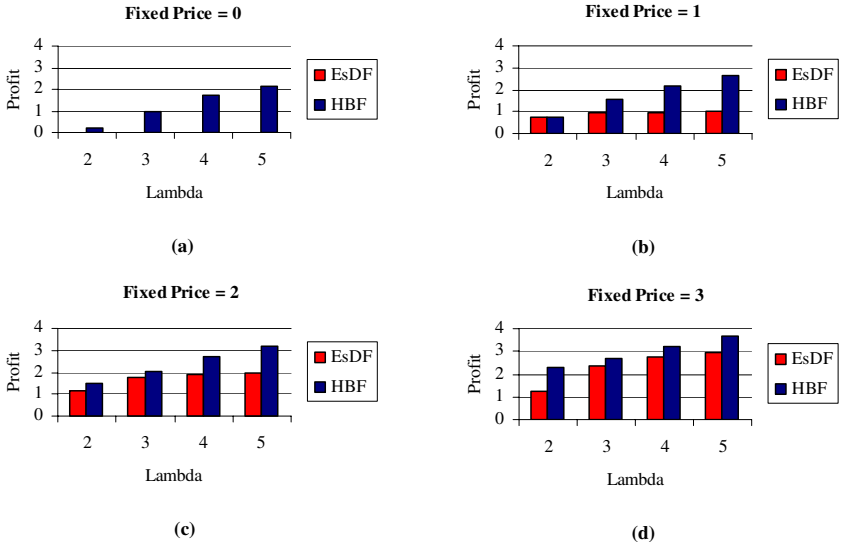


Fig. 13. Servers profit as a function of arrival rate

Server Competition. We simulate a competition between a HBF server and EsDF server, with a fixed price. We allow bids in [0, 1 ... 10]. We vary the fixed price of the EsDF server, in the range [0, 1, 2, 3]. Each Web service customer chooses the server that maximizes his utilitiy.

In the first experiment (Figure 12) we examine the fraction of customers that choose each server as a function of arrival rate. The graphs differ by the EsDF fixed price. Interestingly, even when EsDF server does not charge any price at all, around 50% of the clients choose the HBF server (Figure 12a). We show that the higher the fixed price is, the higher is the percentage of customers that choose the HBF server.

We then compare the profit per time slot gained by each server (Figure 13). We show in Figure 13a that even when the EsDF server offers a free-of-charge service, the profit of the HBF server is larger then 0, giving that some customers, presumably high-value customers, prefer to pay for the service in order to increase their probability of getting processed. Figures 13b, 13c and 13d show that increasing the fixed price positively affects both servers, yet the HBF server constantly gains more profit.

6 Conclusion

In this paper we examined a problem of Web service provision subject to a QoS model. We focused on requests allocation under capacity limitations. We presented a bid-based algorithm, Higher-Bid-First. In this method a customer sends a request to the server, along with a bid. The client maximizes its personal gain by basing its bid on history success-rate table as revealed by a broker. We studied the algorithm in terms of *welfare* and *profit*. We showed that under economic setting, the proposed bid-based algorithm increases the benefits of service providers and customers.

The model we suggested in this paper is simple and subject to many limitations; we assumed an atomic, equal in length Web service provided by each provider; we considered one server with capacity of 1; we also assumed same usage for all requestors.

In future work we plan to expand our model to support a multi-provider architecture, where each provider offers a variety of atomic services. Apart from deadline and value, we will allow requestors to base their bids on number and type of required services and a CPU requirement. We will study a bid-per-service setting, where a customer can submit each service request to a different provider, and a bid-per-server setting, where a customer submits all of its requests to a single server.

Additionally, we plan to re-implement our simulation model using Maui scheduler (available at http://mauischeduler.sourceforge.net/), which is designed to simulate intricate grid architectures. The following will enable us to easily compare our methods to other methods in a grid setting.

Acknowledgement

We thank Prof. Louiqa Raschid from the University of Maryland, College Park for her useful comments.

References

1. Hung, P.C., Li, H.: Web services discovery based on the trade-off between quality and cost of service: A tokenbased approach. ACM SIGecom Exchanges **4** (2003)
2. Lin, Z., Zhao, H., Ramanathan, S.: Pricing web services for optimizing resource allocation an implementation scheme. Web2003, Seattle, WA (2003)

3. Bartal1, Y., Chin, F.Y., Chrobak, M., Fung, S.P., Jawor, W., Lavi1, R., Sgall, J., Tichy, T.: Online competitive algorithms for maximizing weighted throughput of unit jobs. In: 21st STACS, LNCS 2996, Springer (2004) 187–198
4. Li, D., Lin, Z., Stahl, D.O., Whinston, A.B.: Bridging agent-based simulations and direct experiments - an experimental system for internet traffic pricing. AMCIS'01, Boston (2001)
5. Gupta, A., Stahl, D.O., Whinston, A.B.: A stochastic equilibrium model of internet pricing. Journal of Economic Dynamics and Control (1997) 697–722
6. Lin, Z., Ow, P., Stahl, D.O., Whinston, A.B.: Exp loring traffic pricing for the virtual private network. In: WITS. (1999)
7. Stratmann, T.: Logrolling, perspectives on public choice: A handbook. Cambridge University Press, Cambridge, United Kingdom (1997)
8. Mani, A., Nagarajan, A.: Understanding quality of service for web services: Improving the performance of your web services. http://www-106.ibm.com/developerworks/library/ws-quality.html (2002)
9. Fowler, A.: Effective negotiation. Institute of Personnel Management (1986)
10. Chrobak, M., Jawor, W., Sgall, J., Tichy, T.: Improved online algorithm for buffer management in qos switches. ESA (2004)
11. Hajek, B.: On the competitiveness of online scheduling of unit-length packets with hard deadlines in slotted time. conference in Information Sciences and Systems (2001) 434–438
12. Li, F., Sethuraman, J., Stein, C.: An optimal online algorithm for packet scheduling with agreeable deadlines. In: 16th ACM-SIAM SODA. (2005) 460–469
13. Sgall, J.: Online algorithms for scheduling unit jobs. In: 7th MAPSP. (2005)
14. Graham, R.: Bounds for certain multiprocessing anomalies. Bell Sys. Tech. Journal **45** (1966) 1563–1581
15. Kessekman, A., Lother, Z., Mansour, Y., Patt-Shamir, B., Schieber, B., Sviridenko, M.: Buffer overflow management in qos switches. SIAM J. Comput **33** (2004) 563–583
16. Buyya1, R., Abramson, D., Giddy, J., Stockinger, H.: Economic models for resources management and scheduling in grid computation. The Journal on Concurrency and Computation: Practice and Experience 14 (2002) 1507–1542
17. Porter, R.: Mechanism design for online real-time scheduling. In: ACM Conference on Electronic Cemmerce. (2004)
18. Lawler, E., Lenstra, J., Kan, A.R., Shmoys, D.: Sequencing and scheduling: Algorithms and complexity, Logistics of Production and Inventory. Volume 4: Logistics of Production and Inventory. S.C. Graves, A.H.G. Rinnooy Kan and P.H. Zipkin, eds (1990)
19. Edmonds, J.: Paths, trees, flowers. Canadian Journal of Mathematisc (1965) 449–467
20. Güntzer, U., Balke, W.T., Kießling, W.: Optimizing multi-feature queries in image databases. In: Twenty Sixth Very Large Databases (VLDB) Conference. (2001) 419–428
21. Yahav, I.: Bid-based online scheduling of unit-length tasks with hard deadlines. Master's thesis, Thechnion - Israel Institute of Technology (2006)
22. Liu, C., Wayland, J.W.: Scheduling algorithms for multiprogramming in a hard real time environment. Journal of ACM (1973) 4661

Customizable-Resources Description, Selection, and Composition: A Feature Logic Based Approach

Yacine Sam, François-Marie Colonna, and Omar Boucelma

Aix-Marseille Universités, UMR CNRS 6168,
Avenue Escadrille Normandie-Niemen, 13397 Marseille Cedex 20, France
{yacine.sam, colonnaf, omar.boucelma}@lsis.org

Abstract. Users preferences heterogeneity in distributed systems often forces resources suppliers to offer customizable-resources in order to fulfill different customer needs. We present in this paper a Feature Logic based approach to customizable-resources description, selection, and composition. In our approach, resources and requests are both specified in a logical framework by feature terms. The feature terms unification technique allows reasoning on these specifications in order to select and possibly compose the resources that are candidate to satisfy a client request.

1 Introduction

Internet and grid technologies [6] development was accompanied by an exponential amount of resources that a customer (user, program, community) may have access to. However, the great heterogeneity that characterizes customer preferences and needs often forces suppliers to offer their resources under different versions/variants in order to satisfy different customer categories.

In order to better characterize the resources' customization problem, we need to provide an adequate theoretical framework for the description and selection of customizable-resources, i.e., resources being able to be offered under different variants. We believe that existing approaches are not adequate for resources being able to be offered under different variants. These approaches are generally founded on the matchmaking of a "attribute:value" pair, consisting of expressions describing the resources on one side and the clients' requests on the other side (see for instance [10,11,16,21]). Indeed, a framework for customizable-resources description and selection should satisfy two main requirements: (1) the specification of alternative properties for the resources, and (2) the management of the set of all variants of a given resource as a whole; these two characteristics are not provided by the majority of existing resources descriptions frameworks. The first one makes it possible to model different variants of a given resource, while the second allows to publish each resource only once, with no obligation to publish separately all its variants.

Feature logic [18] is a knowledge representation formalism that captures the two characteristics mentioned above. This is why we will adopt it in this article

R. Meersman, Z. Tari et al. (Eds.): OTM 2006, LNCS 4275, pp. 377–390, 2006.

Service	Hotel = $[service : hotel, city : Paris, currency : \{euro, usd, yen\}]$
Variants	$\text{Hotel}_1 = [service : hotel, city : Paris, currency : euro]$
	$\text{Hotel}_2 = [service : hotel, city : Paris, currency : usd]$
	$\text{Hotel}_3 = [service : hotel, city : Paris, currency : yen]$

Fig. 1. Web service providing information on Parisian hotels prices

for a customizable-resources description, selection, and composition framework. Indeed, the feature logic syntax, based on feature terms, allows the expression of alternative properties and the handling of the set of variants of an object (a resource) like only one item. As an example, the first line of Fig. 1 illustrates a feature term representing an informational Web service – a Web service being a kind of resource – on hotels prices in Paris, where the prices are expressed in three different currencies: *euro*, *dollars*, and *yen*. The second line represents the feature terms of its three different variants.

The work described in this paper is the continuation of a previous work on Web services customization [17]. However, in this paper, the term "resource" may refer, besides a Web service, a physical device, a data source, or any other entity able to be supplied in different variants and for which the discovery and selection procedures are required. Our interest in customizable-resources is greatly justified by the strong tendency to the customization of resources: a new paradigm in resources, products, and services supply, known as "Mass Customization Paradigm" (see [14]).

The remainder of this paper is organized as follows: in Section 2 we provide the customizable-resource definition and motivate our choice of a feature logic based language for the specification of a customizable-resource. In Section 3, we introduce the syntax and the semantics of our resource description language as well as a formal calculus that allows the selection and composition of resources. Related works are discussed in Section 4, while conclusions and future directions are in Section 5.

2 Motivation

We start this section in describing our motivating example, which consists of a customizable Web service[1] described by a feature term. We perform reasoning on abstract representations of Web services: a customizable Web service is considered as a conjunction of constraints describing the set of all its variants – or as a class whose instances are the set of its variants. Each request is however a conjunction of constraints asking for a particular variant of a given Web service. For instance, let us describe the informational Web service on hotel prices by means of three features: a *service* (its name); a *city* (city of the required hotel); and a *currency* (the currencies used by the service), where the currency feature may be assigned a value coming from different currency units (see Fig. 1).

[1] We are however convinced that our approach is largely generalizable for other kinds of resources.

The representation adopted for the service and its three variants in Fig. 1 follows a Feature Logic syntax. Our choice for this logic is motivated by the adequacy of its concepts with the need to describe customizable-resources. These concepts, namely *set*, *feature*, and *feature unification*, are detailed in the sequel, in emphasizing their role in the customizable-resources description, selection and composition processes.

2.1 Notion of Set

Feature Logic is a knowledge representation formalism which models set of objects by their properties and provides elementary operations for their manipulation. As an example, the following feature logic syntax describes a set of three Web services.

$$\begin{array}{l}\text{Hotel} = [service : hotel,\\ \qquad\qquad\qquad\quad city : Paris,\\ \qquad\qquad\qquad\quad currency : \{euro, usd, yen\}]\end{array}$$

The three features of this set are "service", "city" and "currency", and their respective values are "hotel", "Paris" and "{euro, usd, yen}". This last value means an alternative on three different values. It is thus possible to specify three different Web services. This notion of set is very adequate for the resources we are dealing with in this paper, i.e., resources being supplied under different variants. Indeed, such resources can be seen as classes and the different variants under which they can be offered as their instances.

2.2 Notion of Feature

In a Feature Logic formalism, set of objects are described by feature expressions. The adoption of this formalism for customizable-resources description implies resources identification, selection and composition schemes based on features. Feature propagation may be used as a technique for customizable-resources composition when the answer to a request requires the combination of several resources.

In other words, Feature Logic allows the specification of resources compositional constraints together with resources descriptions. This characteristic enables the composition in a consistent way of two variants of two different resources without axiomatizing their composition rules. This kind of resources composition is possible thanks to the fact that the compositional constraints are included in the resources descriptions. As an example, let us consider two informational Web services: the first one provides the trip fares from a given French city to Paris, the second one, already described in Fig. 1, provides the prices for Parisian hotels. A query asking prices for a stay in Paris (trip + hotel), could answered by the aggregation of these two services. However, this aggregation needs to be consistent: it should be generated from two compatible variants of two simple Web services. In this example, the variants to be composed should have the same currency unit.

With the feature terms propagation technique [19], a consistent composite Web service variant is obtained by the selection and the composition of two simple Web services variants. The inconsistent composite variants are eliminated by means of feature terms unification [19]. We will formally clarify the feature terms propagation in section 4. However one can note that in the example above, the Web services composition was carried out without using any external information for the descriptions of the two simple Web services involved in the composition.

2.3 Notion of Feature Unification

The customizable-resource selection process consists in determining the resource variants whose features are consistent with those of a given "environment": a request in the case of a simple resource selection and an already selected resource in the case of resources composition. The resources are typically specified by conjunctions of "feature:value" pairs, and the environment by a feature expression. Resources selection is then based on the unification of the feature terms describing these two specifications.

The characteristics presented in this section are the three main motivations of our choice for Feature Logic as a customizable-resource specification formalism. In the following section, we formally present the syntax and the semantics of Feature Logic as well as its inherent inference techniques. We illustrate particularly its application like a customizable-resources description and selection formalism.

3 Framework Description

In this section, we respectively describe (1) the syntax and the semantics of Feature Logic (See [22] for more details), (2) the foundations of our customizable-resources description language and the techniques allowing (3) their selection and (4) their composition during query processing.

3.1 Feature Logic: Syntax and Semantics

As already mentioned, Feature Logic is a knowledge representation formalism based on feature terms (*ft*). A *ft* indicates a set of objects characterized by some features. A feature is a functional property or a characteristic of an abstract object. In its simplest form, a *ft* consists on a conjunction of "feature:value" pairs named *slots*, where each feature represents an object characteristic – an object being a resource in our case. A feature value may include literals, variables, and embedded *ft*. The *ft* syntax is summarized in Fig. 2 where variables are denoted by x, y, z, attributes by f, g, h, constants by a, b, c, and *ft* by S,T. Complex *ft* can moreover recursively be built on elementary *ft* in using well known boolean operators such as intersection, union, and complement. These operators are used to specify logical constraints on features terms and represent the set of objects that satisfies them.

Notation	Name	Interpretation
$\top$ (also [ ])	Top	Universe
$\bot$ (also { })	Bottom	Empty set, Inconsistency
a	Atom	Singleton set containing a
x	Variable	—
f:S	Selection	The value of f is in S
f:$\top$	Existence	f is defined
f:$\uparrow$	Divergence	f is undefined
$f{\downarrow}g$	Agreement	f and g have the same value
$f{\uparrow}g$	Disagreement	f and g have the same value
$\sim S$	Complement	S does not hold
$S{\sqcap}T$ (also [S,T])	Intersection	S and T hold
$S{\sqcup}T$ (also {S,T})	Union	S and T holds
$S{\rightarrow}T$	Implication	If S holds then T holds
$S{\leftrightarrow}T$	Equivalence	S holds if and only if T holds
$\exists x(S)$	Quantification	There is an x such that S holds

Fig. 2. Feature terms syntax and semantics

As an example, if $S = [f : a]$ is the set of objects whose feature f has value a and $T = [g : b]$ is the set whose feature g has value b, the set $S \sqcap T = [f : a, g : b]$ is the intersection of S and T. In other words, it is the set of objects whose feature f has value a and feature g has value b. Similarly, $S \sqcup T = [f : a, g : b]$ is the union of $S = [f : a]$ and $T = [g : b]$; it represents the set whose feature f has value a or the feature g has value b. More generally, *ft* form a boolean algebra and all the boolean transformations such as distribution and Morgan laws are applicable to *ft*.

One of the fundamental characteristics of Feature Logic (shared by all terminological logics) is the possibility to specify incomplete knowledge. As an example, it is possible to specify that a feature exists (the feature is defined.) in a *ft* without giving its value, this is indicated by $f : \top$. It is also possible to specify that an attribute does not exist in a *ft*, this is indicated by $\sim f : \top$ (abbreviation, $f : \uparrow$).

Note 1. The specification of incomplete knowledge is not allowed in the existing approaches to resource description and selection because they are based on matchmaking, not on unification.

Logical complement is available in Feature Logic, which allows a great expressiveness. As an example, the term $\sim$[*currency*:*euro*] in the example of section 2 indicates all the services whose feature *currency* is undefined or having a value other than *euro*. The term [*currency*:$\sim$*euro*] indicates all the Web services whose feature *currency* is defined but with a value other than *euro*.

There are also *ft* representing a single object without any feature (e.g., euro, yen, etc.). Consequently, the equivalences $a \sqcap b = \bot$ and $a \sqcap f : \top = \bot$ are valid for any atom a and b and for any feature f. This led to a simple concept of consistency: since Feature Logic supposes that any feature cannot have more

than one value at the same time, terms $a \sqcap b = \bot$ and $a \sqcap f : \top = \bot$ are equivalent to $\bot$, the empty set. More formally:

$$\begin{aligned}[currency : euro, currency : yen] &= [currency : [euro,\ yen]] \\ &= [currency : \bot] \\ &= \bot\end{aligned}$$

The terms equivalent to $\bot$ denote empty sets and are known as inconsistent terms.

The verification of *ft* consistency is done by feature unification [19]. Feature term unification is a constraints resolution technique which evaluates the consistency of arbitrary *ft*. For terms with neither union nor complement, feature unification behaves like first order logic terms unification. The only difference is that the sub-terms are not identified by their position (as in Prolog), but by feature names. In presence of union, the unification process calculates the union of all the unifiers. However, the complement is usually handled by constraint resolution, similar to the negation by failure.

$$\begin{aligned}\text{Hotel} \sqcap \text{S} &= (\text{Hotel}_1 \sqcup \text{Hotel}_2 \sqcup \text{Hotel}_3) \sqcap \text{S} \\ &= (\text{Hotel}_1 \sqcap \text{S}) \sqcup (\text{Hotel}_2 \sqcap \text{S}) \sqcup (\text{Hotel}_3 \sqcap \text{S}) \\ &= ([service : hotel, city : Paris, currency : euro] \sqcap [currency : euro]) \sqcup \\ &\quad ([service : hotel, city : Paris, currency : usd] \sqcap [currency : euro]) \sqcup \\ &\quad ([service : hotel, city : Paris, currency : yen] \sqcap [currency : euro]) \\ &= ([service : hotel, city : Paris, currency : [euro, euro]]) \sqcup \\ &\quad ([service : hotel, city : Paris, currency : [usd, euro]]) \sqcup \\ &\quad ([service : hotel, city : Paris, currency : [yen, euro]]) \\ &= ([service : hotel, city : Paris, currency : euro]) \sqcup \\ &\quad ([service : hotel, city : Paris, currency : \bot]) \sqcup \\ &\quad ([service : hotel, city : Paris, currency : \bot]) \\ &= \text{Hotel}_1 \sqcup \bot \sqcup \bot \\ &= \text{Hotel}_1\end{aligned}$$

Fig. 3. Customizable-resources selection by feature terms unification

3.2 Customizable-Resources Description

In our customizable-resources description approach, *ft* are used for the specification of the resources' providers as well as for the clients' requests. Thus, resources selection is done by unification of the resources' and the requests' feature terms.

For specifying different resources variants, no constraint is imposed on the existence or the meaning of their features. However, to associate different variants of a given resource, i.e., to describe a class of a given resource by its different variants, one must have at least one common feature value for all the variants – the name of the resource for example. One assumes that each resources variants set can be identified in a single way through an identifier feature named *resource*. This feature represents the resource name (class name), and the name of all its variants.

The universe of all resources[2] is thus denoted [resource : $\top$], which is the set of the sets of resources variants. In other words, it is the set of all resources classes (simple or composite resources). In the sequel,we formally define the concepts of: variants set of a given resource, variant of a given resource, variants class of a given resource, variant of a variants class of a given resource.

Definition 1. *A resource variants set is any set V such as $V \sqsubseteq [resource : \top]$ ($[resource : \top]$ is the universe of all resources).*

Definition 2. *A variant of a variants set of a given resource is a singleton variants set of such a resource i.e., a set V such as $V \sqsubseteq [resource : \top]$ and $\mid V \mid = 1$.*

Definition 3. *A given resource variants class is a set $k \sqsubseteq [resource : k]$ where k is a feature identifying in an unique way all the resources variants.*

Definition 4. *A variant of a variants class of a given resource is a set k such as $k \sqsubseteq [resource : k]$ and $\mid k \mid = 1$.*

Note 2. The difference between a set of variants and a variants class of a given resource is in the type of the considered resource (simple or composite). The first definition can relate to a composition of two or several resources. However, in the second, it relates to only the variants set of a simple resource.

The features of a given resource variants class are modeled as alternatives on the features of each variant. Thus, a resource S supplied under n different variants $V_1, V_2, ..., V_n$, is the union of all its variants.

$$\begin{aligned} S &= V_1 \sqcup V_2 \sqcup \cdots \sqcup V_n \\ &= \sqcup V_i \ (1 \leq i \leq n) \end{aligned}$$

As stated previously, for each resource variants class, at least one of the features must have the same value for all its variants in order for those variants to be grouped together. Thus, the features F allowing the identification of the resources variants class w.r.t other resources variants classes can be factorized as follows:

$$\begin{aligned} S &= (F \sqcap V_1) \sqcup (F \sqcap V_2) \sqcup \cdots \sqcup (F \sqcap V_n) \\ &= F \sqcap (V_1 \sqcup V_2 \sqcup \cdots \sqcup V_n) \end{aligned}$$

For example, the informational Web service for Parisian hotels prices is available in three different variants,

$$\begin{aligned} \text{Hotel}_1 &= [service : hotel, city : Paris, currency : euro] \\ \text{Hotel}_2 &= [service : hotel, city : Paris, currency : usd] \\ \text{Hotel}_3 &= [service : hotel, city : Paris, currency : yen] \end{aligned}$$

and may be described as follows:

[2] By resource, we mean here a class, i.e., the set of all the variants of a given resource.

$$\begin{aligned}\text{Hotel} &= \text{Hotel}_1 \sqcup \text{Hotel}_2 \sqcup \text{Hotel}_3 \\ &= [service : hotel, city : Paris, \\ &\qquad currency : \{euro, usd, yen\}]\end{aligned}$$

3.3 Customizable-Resources Selection

In order to find a particular variant of a given resource, the resource requester specifies a selection term S defining the features of the desired variant. Thus, for each selection term S and a set of variants T, one can select the resource variants that satisfies S by evaluating the expression $T' = T \sqcap S$. T' is the set of variants being both a subset of S and T. If $T' = \bot$, the selection fails because it does not indicate any available resource variant.

In our example, the selection of $S = \{currency : euro\}$ from our informational Hotel prices service returns Hotel_1 because this is the only variant that is compatible with the selection term S. More formally:

$$\begin{aligned}\text{Hotel} \sqcap \text{S} &= (\text{Hotel}_1 \sqcup \text{Hotel}_2 \sqcup \text{Hotel}_3) \sqcap \text{S} \\ &= (\text{Hotel}_1 \sqcap \text{S}) \sqcup (\text{Hotel}_2 \sqcap \text{S}) \sqcup (\text{Hotel}_3 \sqcap \text{S}) \\ &= \text{Hotel}_1 \sqcup \bot \sqcup \bot \\ &= \text{Hotel}_1\end{aligned}$$

$\text{Hotel}_2 \sqcap \text{S} = \bot$ et $\text{Hotel}_3 \sqcap \text{S} = \bot$ are valid since the feature "currency" cannot have more than one value at the same time (see Fig. 3).

When the resource description has more than one feature being able to be specified under several alternatives, T' could be another variants set. Thus one specifies a new selection term S' and selects T''=$T' \sqcap S'$, and so on. This selection procedure narrows the choice set in an iterative way until a singleton set is selected, this latter corresponds to the desired resource variant.

3.4 Customizable-Resources Composition

The problem of resources selection in distributed systems is often complicated due to the fact that, in many situations, the response to a given request cannot be satisfied by one resource only, but involves the combination of several ones. This situation leads to the need for introducing compositional techniques into the resources management system.

Within our customizable-resources specification framework, the resources composition is obtained by the intersection of the feature terms describing them. We say thus that the composite resource inherits its feature terms from the resources composing it. Thus, if K_1, K_2, ...,K_n are the feature terms describing n resources being part of a composition, the composite resource is described by the feature term C such that:

$$\begin{aligned}C &= K_1 \sqcap K_2 \sqcap \cdots \sqcap K_n \\ &= \sqcap K_i (1 \leq i \leq n)\end{aligned}$$

However, the consistency of the composite customizable-resource must be checked when making the intersection of the feature terms specifying the different resources variants involved in the composition. In other words, the composition of two resources variants is not consistent only when they do not share any feature or when the values of the shared features are the same. If the resources to be composed share at least one common feature, the problem of resource composition consists in selecting a consistent configuration (a variant of each resource while ensuring mutual compatibility between them) through the feature propagation mechanism from one resource variants class to another. The resources composition process proceeds as follows: one selects initially a variant $R1$ of a first resource variants class R, then a compatible variant $S1$ of a second resource variants class S. More generally, the composition of several resources implies the selection and the composition of different resources variants classes while ensuring the consistency of the resulting composite resources.

The Feature Logic propagation technique allows resources composition without requiring any preliminary external knowledge. Indeed, knowledge governing the composition is described in the resources themselves (second fundamental characteristic of Feature Logic: the notion of feature, see Section 2). In our Web services variants classes example, the composition constraints can be specified in the selection term, but also in the features describing the Web services. For this reason, resources heritage and consistency are carried out by modeling composite Web services like the intersection of the features describing the simple Web services variants classes constituting them, while excluding inconsistent combinations. One cannot for example compose two Web services variants described by a feature *currency* in euro [currency:euro] and in yen [currency:yen] respectively since a feature term cannot have more than one value at the same time. More formally:

$$\begin{aligned}[currency : euro,\ currency : yen] &= [currency : [euro, yen]] \\ &= [currency : \bot] \\ &= \bot\end{aligned}$$

Thanks to this feature unification technique, only consistent composite resources variants are selected during the customizable-resources composition process. Nevertheless, this technique should not be applied to features identifying the resources since those features must be aggregated: this is illustrated by the example below.

Fig. 4 illustrates two Web services: the Parisian hotels prices service, and the Rome-to-Paris trip fares service. If one wants to compose these two Web services in order to offer an informational service on both hotels prices in Paris and trip fares from Rome to Paris, the composite service name must be described by both "hotel" and "trip". Thus, the feature *service* representing the name of a service should not be unified during the Web services composition process.

The need for aggregating features identifying the resources descriptions means that the intersection of feature terms is not well adapted for resource composition. Hence we define an operator equivalent to the intersection "$\sqcap$" but with

Service 1	Hotel $= [service : hotel, city : Paris, currency : \{euro, yen\}]$
Service 1 Variants	$\text{Hotel}_1 = [service : hotel, city : Paris, currency : euro]$ $\text{Hotel}_2 = [service : hotel, city : Paris, currency : yen]$
Service 2	Trip $= [service : trip, departure : Rome, city : Paris,$ $currency : \{euro, yen\}]$
Service 2 Variants	$\text{Trip}_1 = [service : trip, departure : Rome, city : Paris,$ $currency : euro]$ $\text{Trip}_2 = [service : trip, departure : Rome, city : Paris,$ $currency : yen]$

Fig. 4. An example of two Web services offered in two different variants

the particularity of aggregating instead of unifying the "independent" features: this operator is denoted "∇". By independent feature we denote any feature related to the resource itself, i.e., a feature that does not impact the features of other resources during the resources composition processes. In addition to the resources identifiers such as *service*, several other features may be considered as independent. Let us consider for example a feature describing the owner of a given resource. In this example, two resources belonging to two different providers can be aggregated in only one resource whose feature *owner* value is the aggregation of the two resources composing its feature *owner* values.

Since each resource variants class K_i is a set of variants, there can be more than one consistent composite resource. For example, the set of consistent composite Web services variants of the two services mentioned above (Parisian hotels prices and Rome-to-Paris trip fares) corresponds to lines 2 and 5 in Fig. 5 below.

However, the consistency of the composition to be realized from several resources variants classes implies the inclusion at the same time of only one variant of each resource variants class participating in the composition. In the composition example of Fig. 5, if we start with the selection of the variant in *euro* Rome-to-Paris trip fares, it is impossible to select the variant in *yen* for the Parisian hotels. Indeed, this makes the composition inconsistent because of the two different values of the currency feature in the two services (see lines 3 and 4 in Fig. 5.

Now that we have described our customizable-resources language and its selection and composition techniques, we survey, in Section 4, existing approaches to resources description and selection and we discuss their inadequacy for the specification of customizable-resources.

4 Related Work

The problem of resources selection was tackled under several contexts, and many approaches, based mainly on resources matchmaking were proposed in the literature. We present in what follows the most known approaches.

In the information systems area, much efforts were devoted to develop systems allowing the publication, the interrogation and the aggregation of resources

$$
\begin{array}{|l|}
\hline
\text{Hotel} \nabla \text{ Trip} = [service : hotel, city : Paris, currency : \{euro, yen\}] \nabla \\
\quad [service : trip, departure : Rome, city : Paris, currency : \{euro, yen\}] \\
\quad = ([service : hotel, city : Paris, currency : euro] \sqcup \quad (1) \\
\quad [service : hotel, city : Paris, currency : yen]) \nabla \quad (2) \\
\quad ([service : trip, departure : Rome, city : Paris, currency : euro] \sqcup \quad (3) \\
\quad [service : trip, departure : Rome, city : Paris, currency : yen]) \quad (4) \\
\hline
(1) \nabla (3) = [service : hotel, city : Paris, currency : euro] \nabla \\
\quad [service : trip, departure : Rome, city : Paris, currency : euro] \\
\quad = [service : [hotel, trip], city : Paris, currency : euro, departure : Rome, \\
\quad city : Paris, currency : euro] \\
\quad = [service : [hotel, trip], city : [Paris, Paris], currency : [euro, euro], \\
\quad departure : Rome] \\
\quad = [service : [hotel, trip], city : Paris, currency : euro, departure : Rome] \\
\quad \neq \bot \\
\hline
(1) \nabla (4) = [service : hotel, city : Paris, currency : euro] \nabla \\
\quad [service : trip, departure : Rome, city : Paris, currency : yen] \\
\quad = [service : [hotel, trip], city : Paris, currency : euro, departure : Rome, \\
\quad city : Paris, currency : yen] \\
\quad = [service : [hotel, trip], city : [Paris, Paris], currency : [euro, yen], \\
\quad departure : Rome] \\
\quad = [service : [hotel, trip], city : Paris, currency : \bot, departure : Rome] \\
\quad = \bot \\
\hline
(2) \nabla (3) = [service : hotel, city : Paris, currency : yen] \nabla \\
\quad [service : trip, departure : Rome, city : Paris, currency : euro] \\
\quad = [service : [hotel, trip], city : Paris, currency : yen, departure : Rome, \\
\quad city : Paris, currency : euro] \\
\quad = [service : [hotel, trip], city : [Paris, Paris], currency : [yen, euro], \\
\quad departure : Rome] \\
\quad = [service : [hotel, trip], city : Paris, currency : \bot, departure : Rome] \\
\quad = \bot \\
\hline
(2) \nabla (4) = [service : hotel, city : Paris, currency : yen] \nabla \\
\quad [service : trip, departure : Rome, city : Paris, currency : yen] \\
\quad = [service : [hotel, trip], city : Paris, currency : yen, departure : Rome, \\
\quad city : Paris, currency : euro] \\
\quad = [service : [hotel, trip], city : [Paris, Paris], currency : [yen, yen], \\
\quad departure : Rome] \\
\quad = [service : [hotel, trip], city : Paris, currency : yen, departure : Rome] \\
\quad \neq \bot \\
\hline
\end{array}
$$

Fig. 5. Customizable-resources composition

collections (SNMP [20], LDAP [8], MDS [4], UDDI [12]). Such systems differ from each other in many aspects: their data model (e.g., MIBs [20], relations [7], objects LDAP [8]), and their query languages (e.g., SQL [7], XQuery [3], LDAP query [8]). However they all share the asymmetrical mode of interaction between the resources providers and requesters.

In the asymmetrical interaction, the providers describe and publish their resources in order to be identified by the requesters before any request generation. Requesting procedures are then carried out via simple requests or procedural programs based on performance models for example [2,5]. In these approaches,

the access control to the resources systems is managed manually by an administrator through Accounts/Identifiers. However because of scalability constraints, such management is not convenient. To overcome such situation, a resources *symmetrical* selection method has been proposed.

Symmetrical selection has been introduced for the first time in the Condor matchmaking system [9]. In Condor, requests and resources descriptions are specified in the same language (syntax add-class: advertising class) [10]. Resources and requests properties are (attribute:expression) pairs and the constraints on them are expressed in the form of Boolean expressions. Matchmaking succeeds if there is a resource that proposes attributes values being able to satisfy those of the client's request. The Condor system maintains a set of classes describing the resources (add-class) and starts the matchmaking procedure between the client's request and all the published resources in a resources directory.

Other systems adopting symmetrical matchmaking include *Jini lookup* service [1], DAML-S *matchmaker* [13], LARKS *matchmaker* [21], and HP E-trade *matchmaker* [15]. These systems use different syntaxes (Java objects, services profiles, RDF classes, Description Logics concepts) and different matchmaking levels (syntactic, semantic, or both), while sharing the symmetrical evaluation principle and the fact that the resources and the requests are described by the same syntax. However, none of them, including those previously mentioned, takes into account the possibility of specifying a resource under several variants or the possibility to specify a request on a particular variant of a given resource. This need to specify knowledge on customizable-resources justifies the introduction of our Feature Logic resources specification framework.

Our resources specification language is close to the LARKS one [21]. LARKS is indeed a language for the specification of "attribute:value" pairs describing the role of Web services as well as their constraints usage (constraints on the "attribute:value" pairs). In a previous work [17], we presented an approach adapting LARKS for the Web services customization. However, in such approach, resources selection procedure was founded on the resources and requests matchmaking, and the resource customization was carried out by dynamic Web services composition of other intermediate Web services, which is not always not convenient. The framework presented in this paper is based on the feature terms unification [19], that is a Feature Logic reasoning technique. To the best of our knowledge, no approach based on Feature Logic has already been proposed for the resources description and selection in general, and for the customizable-resources in particular.

Finally, for the customizable-resources description, the advantage of the unification compared to matchmaking is twofold. On one hand, the unification makes it possible to compare resources descriptions even in the presence of incomplete information. On the other hand, when reasoning on resources described by feature terms, the resources customization becomes obvious, this is a direct consequence of the intrinsic characteristics of Feature Logic: (1) the possibility to express alternative properties and (2) the handling of a set of variants of a given resource like only one item.

5 Conclusion and Future Work

Users' profiles heterogeneity in distributed systems forces resources suppliers to provide customizable-resources in order to satisfy various customers needs and categories. Existing approaches for the description and selection of resources are not adequate for the specification of resources that might be supplied under several variants. This paper proposes a Feature Logic based framework for the description, selection, and composition of customizable-resources. Feature Logic is a terminological logic that allows the specification of alternative properties on objects, thus fulfilling the requirements for customizable-resources specification. Feature Logic terms are used to describe clients' requests and providers' customizable-resources, while the feature terms unification allows reasoning on those descriptions in order to select, and possibly compose, the resources that satisfy different clients' requests.

In this paper, soundness and applicability of a Feature Logic approach to the customization of web resources have been highlighted. However, this work will remain incomplete if an illustrating implementation is not provided. This will result in a tool that should be compared to existing tools and formalisms for customization and composite Web Services.

References

1. K. Arnold, B. Osullivan, R. W. Scheifler, J. Waldo, A. Wollrath, B. O'Sullivan, and R. Scheifler. *The Jini(TM) Specification*. Addison-Wesley, MA, USA, 1999.
2. F. Berman and R. Wolski. *The AppLeS project : A status report. Proceedings of the 8 th NEC Research Symposium*. Berlin, Germany, 1997.
3. S. Boag, D. Chamberlin, M. F. Fernández, D. Florescu, J. Robie, and J. Siméon. *XQuery 1.0: An XML Query Language*. W3C, 2002.
4. K. Czajkowski, C. Kesselman, S. Fitzgerald, and I. T. Foster. Grid information services for distributed resource sharing. In *HPDC*, pages 181–194, 2001.
5. H. Dail. *A Modular Framework For Adaptive Schedulling in Grid Application Development Environments*. Computer Science, University of California, San Diego, 2002.
6. I. Foster and C. Kesselman. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers, San Francisco, 1999, xxiv, 667 pp, 2004.
7. H. Garcia-Molina, J. Ullman, and J. Widom. *Database Systems: The Complete Book*. Printice Hall, Upper saddle River, N1, xxvii, 1119pp, 2002.
8. T. Howes, M. C. Smith, G. S. Good, T. A. Howes, and M. Smith. *Understanding and Deploying Ldap Directory Services*. Addison-Wesley Professional, 608 pp, 2003.
9. M. J. Litzkow, M. Livny, and M. W. Mutka. Condor - a hunter of idle workstations. In *ICDCS*, pages 104–111, 1988.
10. C. Liu and I. T. Foster. A constraint language approach to grid ressource selection. In *niversite Of Chicago. Chicago.*, 2003.
11. C. Liu and I. T. Foster. A constraint language approach to matchmaking. In *RIDE*, pages 7–14, 2004.
12. E. Newcomer. *Understanding Web Services: Xml, Wsdl, Soap, and Uddi*. Addison-Wesley, Boston, xxviii, 332 pp, 2002.

13. T. R. Payne, M. Paolucci, and K. Sycara. *Advertising and Matching DAML-S Service Descriptions*. International Semantic Web Symposium (SWWS), Standford University, California, USA, 2001.
14. B. J. Pine and S. Davis. *Mass Customization: The New Frontier in Business Competition*. Harvard Business School Press, 1993.
15. C. Preist. *Agent Mediated Electronic Commerce at HP Labs Bristol*. Hewlett-Packard Labs, Bristol, 2001.
16. R. Raman, M. Livny, and M. H. Solomon. Matchmaking: Distributed resource management for high throughput computing. In *HPDC*, pages 140–, 1998.
17. Y. Sam, O. Boucelma, and M. S. Hacid. *Web Services Customization : A composition-based approach, In ICWE'06*. ACM Press, July 2006, to appear.
18. G. Smolka. Feature-logik. In *GWAI*, pages 477–478, 1989.
19. G. Smolka. Feature-constraint logics for unification grammars. *J. Log. Program.*, 12(1&2):51–87, 1992.
20. W. Stallings. *SNMP, SNMPv2, SNMPv3, and RMON 1 and 2 (3rd Edition)*. Addison-Wesley, Reading, Mass, xv, 619 pp, 1999.
21. K. P. Sycara, S. Widoff, M. Klusch, and J. Lu. Larks: Dynamic matchmaking among heterogeneous software agents in cyberspace. *Autonomous Agents and Multi-Agent Systems*, 5(2):173–203, 2002.
22. A. Zeller and G. Snelting. Unified versioning through feature logic. *ACM Trans. Softw. Eng. Methodol.*, 6(4):398–441, 1997.

Defining and Modelling Service-Based Coordinated Systems

Thi-Huong-Giang Vu, Christine Collet, and Genoveva Vargas-Solar

LSR-IMAG Laboratory, BP 72,
38402 Saint Martin d'Hères, France
{Thi-Huong-Giang.Vu, Christine.Collet, Genoveva.Vargas}@imag.fr

Abstract. This paper introduces *MEO* - a model for securing service-based coordinated systems. The model uses constraints for expressing the application logic of a coordinated system and its required security strategies. Coordination activities are the key concepts used for controlling the execution of participating services. Constraints are specified as pre and post conditions of these coordination activities.

1 Introduction

The democratization of Internet along with recent advances in information technologies has made the global networked marketplace vision a reality. In such an environment, companies form alliances for building information systems that aggregate their services. Effective service sharing and integration is a critical step towards developing next generation of information systems for supporting the new online economy. Given the time-to-market, rapid development and deployment requirements, information systems are made up of the services of different providers that are accessible through networks, e.g., Internet. Such information systems are called *coordinated systems*. A service provider is an autonomous organism that keeps control on the service execution with respect to some non-functional aspects such as security. It predefines instructions and descriptions for using its services (e.g., where and when functions of these services can be accessed). Using a service implies invoking a method and (possibly) waiting for execution results.

Numerous systems, models and languages have been proposed for supporting services coordination, i.e., the way services invocations are orchestrated according to the application logic of a given coordinated system. Existing solutions such as workflow models [5,8] or Petri nets [18] tackle the specification and enactment of services coordination. Using a workflow model, the execution of a coordinated system is controlled by a data flow and a control flow. The data flow specifies data exchange among participating services. The control flow describes their dependencies and it is expressed by ordering operators such as sequence, selection (OR-split, OR-joint) and synchronization (AND-split, AND-joint). Using a Petri net, the execution of a coordinated system is expressed by rules applied on data delivered to or consumed by participating services (i.e., places). It defines (i)

R. Meersman, Z. Tari et al. (Eds.): OTM 2006, LNCS 4275, pp. 391–407, 2006.

rules for abstracting the structure of exchanged data (i.e., tokens) among services and (ii) rules for scheduling and sending input and output data that can fire some service execution (i.e., transitions). The interaction among services has been facilitated by current technologies that either adopt approaches based on interoperation [9,6] and based on intercommunication [10].

While particular attention has been devoted to services coordination, non-functional aspects such as security have been poorly addressed by existing coordination models, languages and execution engines. It is hard to accurately specify what a coordinated system has to do under specific security requirements such as authentication, reliability, non repudiation and messages integrity. It is also often difficult to consider in advance the coordination of participating services under a large set of interactions and interdependencies among them. A loose specification of an application logic can lead to a wrong order of interactions among services. We can also mistreat real situations during the coordination execution, e.g., invoked service is undesirably replaced by another. Another example is that information exchanged among services in the coordination context is altered without our knowledge. Managing secure coordination at execution time implies:

- Verifying messages integrity (i.e., those exchanged among services) in order to avoid their unauthorised alteration.
- Authentication of the services that participate in a coordination process (i.e., identify the invoked service and the service that provides results after an invocation).
- Ensuring non repudiation of coordination: post-check the validity of coordinated system execution and prevent a participating service from denying previous actions.

The challenges are to avoid security vulnerabilities of services coordination and provide strategies for ensuring security at run-time. Moreover, the proposed strategies should not contradict the facility of the coordinated system construction and services flexibility. It should be possible to adapt coordination and security aspects of coordinated systems according to different topologies, scenarios, delegation requirements and security configurations.

We focus on the functional safety of coordination. We propose to enrich and adapt a coordination specification in a way that unintended behaviours can be detected and rectified. We propose to model such a control as constraints associated to so called coordination activities. Our model, called MEO (constraint-based Model for sEcure cOordination) offers concepts to describe the coordination of services as coordination activities and their associated constraints. These constraints are used to express coordination and security strategies for a coordinated system.

The remainder of this paper is organized as follows. Section 2 introduces the MEO model. Section 3 and 4 describe the way coordination and security strategies are expressed in our model. Section 5 explains how to specify coordination and security aspects of a coordinated system. Section 6 compares our work with

existing ones. Finally, section 7 concludes the paper and discusses further research directions.

2 The Main Concepts of MEO

This section details the main concepts of *MEO*, the proposed constraint-based Model for sEcure cOordination.

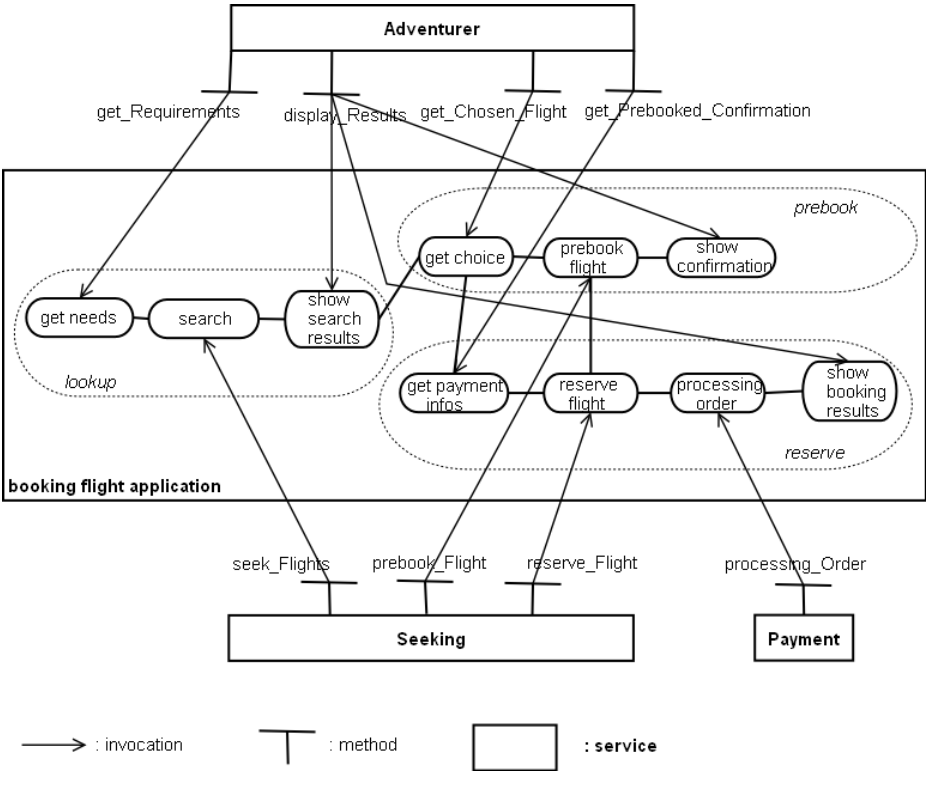


Fig. 1. Booking flight application

Let us consider a flight booking application expressed as a *coordinated system* in Figure 1 (called *coordinated system*). The system is built from three existing services (called *participating services*).

- *Adventurer* manages clients that are interested in booking flights.
- *Payment* executes online payment transactions on given client accounts.
- *Seeking* looks for available seats and performs flight pre-booking operations on a flight database.

By invoking methods of these services, the flight booking application performs the following functions:

(i) Lookup an available flight list according to given client's needs.
(ii) Book a flight by choosing it from the returned available flights list.
(iii) Buy a tickets.

As shown in Figure 2, using *MEO*, such a coordinated system is modelled through three fundamental concepts: coordination activity, coordination scenario and constraint. The application logic of this coordinated system is specified as a set of constraints on coordination activities within a given coordination scenario. Similarly, constraints expressed on security properties provided by services are coupled with reliability constraints used to express control flow among coordination activities in order to specify security strategies.

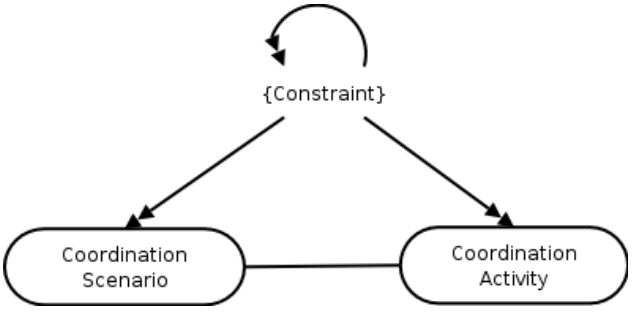


Fig. 2. Main concepts of MEO

2.1 Coordination Activity

A *coordination activity* specifies an interaction between two services, where one invokes a function provided by another and (possibly) waits for its execution results. Generally speaking a coordination activity (CA) is characterized as CA = (P, F, S, I, O, A), where:

- P is the profile that identifies the coordination activity by a unique name.
- F denotes a specific function of a participating service.
- S is the current execution state of the coordination activity.
- I contains a set of parameters related to the input data of the coordination activity.
- O contains a set of parameters related to the output data of the coordination activity.
- A is a set of attribute. Every attribute represents a specific property of the coordination activity and a possible value.

For instance, the reservation function in our example is described as follows: In case of buying a booked flight, the booking confirmation is transferred to

the *Payment* service. These information is redirected to the *Seeking* service to validate the reservation by invoking the method reserve_Flight. We define the following coordination activities:

- CA_7, where the payment information is collected:
  P_7 = getPaymentInfos; F_7 = *Adventurer*.get_Booked_Confirmation.
- CA_8, for booking the flight according to the information received as input:
  P_8 = reserveFlights; F_8 = *Seeking*.reserve_Flights.
- CA_9, for processing a client's order according to the information received as input: P_9 = processingOrder; F_9 = *Payment*.processing_Order.
- CA_{10}, for displaying the booking result:
  P_{10} = showBookingResult; F_{10} = *Adventurer*.display_Results.

2.2 Coordination Scenario

A *coordination scenario* is the history of the execution of a set of coordination activities. A coordination scenario (CS) is characterized by CS = (N, L, {(ER, IT, OT)}, CT), where:

- N is a name identifying the scenario.
- L is a log that registers the execution status of a coordination.
- ER is an execution reference of an identified coordination activity.
- IT (incoming trace), contains the values of the input parameters of an identified coordination activity.
- OT (outgoing trace) contains the values of the output parameters of an identified coordination activity.
- CT (context trace) contains the values describing the execution context of an identified coordination activity.

For every coordination activity CA_j belonging to the scenario CS_i, the following information is traced: information about the instant in which a participating service starts or finishes a function (i.e., $ER_{i,j}$). Information about message producers and consumers (e.g., services) is logged (CT_i) with a time stamp and a signature, i.e., $IT_{i,j}$, $OT_{i,j}$. All coordination activities of CS_i result in a log that contains information about successful or unsuccessful execution of the coordination scenario CS_i. We can then analyze these traces to extract information that can be used to diagnose coordination failures. Through the log provided by the scenario, the coordination can be monitored and evaluated.

Considering our example, we have three coordination scenarios CS_1, CS_2, CS_3, which are respectively the execution information of the lookup, book and purchase functions. For instance, the coordination scenario CS_3 is defined as follows:

$CS_3 = (N_3, L_3, \{(ER_{3,j}, IT_{3,j}, OT_{3,j})\}, CT_3)$, where N_3 = "*reserve*" and $(ER_{3,7}, IT_{3,7}, OT_{3,7})$, $(ER_{3,8}, IT_{3,8}, OT_{3,8})$, $(ER_{3,9}, IT_{3,9}, OT_{3,9})$, $(ER_{3,10}, IT_{3,10}, OT_{3,10})$ are respectively the traces of coordination activities CA_7, CA_8, CA_9, CA_{10} of the purchase function.

2.3 Constraint

Constraint is the basic concept used to manage and control coordination. A constraint defines the behaviour, the data, the characteristic or the interface that is selected as a requirement for a coordination activity or to refer to coordination scenarios. Constraints can be enabled, validated, observed and verified. Constraints are defined using set operators ($\cup, \cap, -$), logic operators ($\vee, \wedge, \neg, \rightarrow$), relation operators ($\in, \notin, \subset, \subseteq, \supset, \supseteq, \neq$) and functions. A constraint (C) is characterized as $\mathsf{C} = (\mathsf{P}, \mathsf{SC}, \mathsf{T}, \mathsf{ST})$, where:

- P is a predicate defining the condition to be evaluated.
- SC defines the scope of the constraint. It may be a coordination activity and a coordination scenario.
- T is the occurrence time of the constraint verification. It can be before, during or after (i) the execution of a coordination activity, (ii) the activation of a coordination context and (iii) the validation of another constraint. So, constraints are of three types: pre-condition, post-condition and invariants.
- ST is the structure that characterizes possibilities for composing constraints. It has three values: atomic, composite and nested.

Given a set of coordination activities needing to be orchestrated according to specific functional (e.g. coordination) and non functional aspects (e.g. security), constraints express these requirements. So for each coordination activity, constraints describe what is considered, permitted, obligatory and forbidden. The following sections detail the constraints designed for our coordinated system example to ensure its functional safety.

3 Coordination Strategy

A coordination strategy corresponds to a specific logic property of a coordinated system. Properties are expressed by *functional constraints* considering the exchanged data and the temporal relationships among a set of coordination activities. Such constraints are added to coordination activities and are evaluated on coordination scenarios. We identify the following functional constraint types: ordering, firing and data dependency. The following sections define these constraint types.

3.1 Ordering

An *ordering constraint* represents temporal relationships between two coordination activities. Let CA_i, CA_j be two coordination activities to be orchestrated, and st_i, st_j and et_i, et_j their start and termination times respectively. Three ordering constraints can be associated to CA_i in order to determine its relationship with CA_j.

- Before(CA_j): $TRUE$ if CA_i starts before CA_j, $FALSE$ otherwise.
- After(CA_j): $TRUE$ if CA_i terminates after CA_j, $FALSE$ otherwise.
- Simultaneous(CA_j): $TRUE$ if CA_i starts and terminates simultaneously with CA_j, $FALSE$ otherwise.

3.2 Firing

A firing constraint represents the situation where the execution of a coordination activity cannot be fired until the execution of another coordination activity has terminated in a particular state.

Let CA be a coordination activity to be orchestrated, and $\mathsf{STATE} = \{\mathsf{S}_1,\mathsf{S}_2,..,\mathsf{S}_n\}$ be the set of possible termination states for this coordination activity. The following constraints can be defined on each state $\mathsf{S}_i \in \mathsf{STATE}$ of CA:

- Consider(S_i): $TRUE$ if S_i is *considered* to fire the execution of the coordination activity CA, $FALSE$ otherwise.
- Permit(S_i): $TRUE$ if S_i is *permitted* in order to decide whether to fire the execution of the coordination activity CA, $FALSE$ otherwise.
- Obligate(S_i): $TRUE$ if S_i is *obligatory* to fire the execution of the coordination activity CA, $FALSE$ otherwise.
- Forbid(S_i): $TRUE$ if S_i must not happen if the execution of the coordination activity CA must be fired, $FALSE$ otherwise.

3.3 Data Dependency

A data dependency constraint indicates the situation where a coordination activity consumes data produced by another coordination activity.

Let CA_i, CA_j be two coordination activities to be orchestrated, I_i, I_j, and O_i, O_j their input and output data respectively. There are three possibilities for the input data I_i of the coordination activity CA_i in relation with the output data O_j of the coordination activity CA_j:

- Contain(O_j): $TRUE$ if I_i contains O_j, $FALSE$ otherwise.
- Belong(O_j): $TRUE$ if I_i belongs O_j, $FALSE$ otherwise.
- Match(O_j): $TRUE$ if I_i matches O_j, $FALSE$ otherwise.

4 Security Strategy

We use *non functional constraints* to define security aspects such as integrity, authentication, authorisation, non repudiation and the final objective is the functional safety of coordination. Constraints are added to coordination activities to define authentication, authorisation, non repudiation and the integrity of exchanged messages.

4.1 Integrity

An integrity constraint provides a way for ensuring that authorized changes made to an application logic or a coordination activity do not violate their consistency. The consistency of a coordinated system is defined by the properties of coordination activities (e.g., profile), a coordination scenario (e.g., execution state) and constraints (e.g., scope). An integrity constraint associated to a specific property restricts the values that can be assigned to it.

Let us denote such a property by Ch and the set of its possible valid values by $V = \{v_1, v_2, .., v_n\}$. The integrity Ch is determined by the following constraints:

- Unique(Ch): $TRUE$ if the value v to be assigned to Ch is unique among all possible values $v_i \in V$, $FALSE$ otherwise.
- Range(Ch, [v_x, v_y]): $TRUE$ if the value v to be assigned to Ch belongs to an interval [v_x, v_y], where v_x and $v_y \in V$, $FALSE$ otherwise.
- Free(Ch): $TRUE$ if the value v is free to be assigned to Ch, $FALSE$ otherwise.
- Invariant(Ch): $TRUE$ if the value v is not altered after having been assigned to Ch, $FALSE$ otherwise.

For a given tuple of characteristics (Ch_1, Ch_2,.., Ch_n) the following constraints can be defined in relation with their real assigned values:

- Nul(Ch_i): $TRUE$ if no value is assigned to Ch_i, $FALSE$ otherwise.
- Exist(Ch_i): $TRUE$ if there is a Ch_i with a valid assigned value, $FALSE$ otherwise.
- ForAll(Ch_i): $TRUE$ all Ch_i have valid assigned values, $FALSE$ otherwise.

The next paragraphs describe how to associate these constraints on coordination activities and an application logic.

For coordination activity: Given a set of coordination activities CA_set = $\{CA_1, CA_2, .., CA_n\}$, where $CA_i = \{P_i, F_i, S_i, I_i, O_i, A_i\}$, the following integrity constraints can be defined:

- Unique(P_i): the profile (P) must have an unique combination of values for the name and the description.
- Unique({F_i, provider, context}): (F) invoked within the coordination context, must refer to a unique function provided by a unique service.
- Range(S_i, [R_1, R_2, R_3]): the execution state (S) of a coordination activity can have one of the following values R_1, R_2, R_3.
  - In the preparation step there are three values to be assigned to the state of a coordination activity CA_i: R_1 = {UNDERCONSIDERATION, WAIT, READY}.
  - In the treatment step there is one value to be assigned to the state of coordination activity CA_i: R_2 = ACTIVE.
  - In the termination step there are three values to be assigned to the state of coordination activity CA_i: R_3 = {ABORT, COMMIT, COMPENSATION}.
- ForAll(I_i): all input parameters (I) must have associated valid values.
- Invariant((O_i)): output parameters (O), must be consistent when they are exchanged.
- Exist(A_i): attributes (A) must have some valid associated values.

For application logic: Such a constraint is coupled with ordering constraints to verify exchanged data among a sender and a recipient.

4.2 Authentication

An authentication constraint represents the situation where an invocation in a coordination activity occurs only until its sender and/ or its recipient have been identified. Typically, authentication constraints merely ensure that the invocation sender is the coordinated system and/ or the invocation recipient is a participating service. The authentication card is associated with a coordination activity as its authentication attributes, e.g. digital certificates. It serves to the mutual authentication.

Let $\mathsf{CA\_set} = \{\mathsf{CA}_1,\mathsf{CA}_2,..,\mathsf{CA}_n\}$ be a set of coordination activities to be orchestrated, $\mathsf{CA}_i = \{\mathsf{P}_i, \mathsf{F}_i, \mathsf{S}_i, \mathsf{I}_i, \mathsf{O}_i, \mathsf{A}_i\}$, a set of possible names used to identify these coordination activities as $\mathsf{NAME\_set} = \{\mathsf{N}_1,\mathsf{N}_2,..,\mathsf{N}_n\}$, $\mathsf{N}_i \in \mathsf{P}_i$, and $\mathsf{DEMO\_set} = \{\mathsf{D}_1,\mathsf{D}_2,..,\mathsf{D}_n\}$, $\mathsf{D}_i \in \mathsf{A}_i$ a set of possible cards used to authenticate these coordination activities. The following authentication constraints are defined on the identity and the identity card:

- $\mathsf{NotConsidered}(\mathsf{N}_i)$: $TRUE$ if the name N_i is not considered as an authentication identity of the coordination activity CA_i, i.e., there is no need for authenticating CA_i, $FALSE$ otherwise.
- $\mathsf{Approved}(\mathsf{N}_i)$: $TRUE$ if the name N_i is in the list of approuved identities of the coordinated system, $FALSE$ otherwise.
- $\mathsf{Coupled}(\mathsf{N}_i, \mathsf{D}_i)$: $TRUE$ if the authentication card D_i of the coordination activity CA_i is correctly demonstrated for its identity N_i, $FALSE$ otherwise.

At the execution level, once sender's and recipient's identities are explicitly represented, authentication constraints are validated by the relationships between authentication demonstration and the implicit sender and recipient of a coordination activity. Authentication constraints will be then applied on the function dimension (F) of a coordination activity.

4.3 Authorisation

Based on the identity of a specific coordination activity, an authorisation constraint represents the situation where the invocation of a function is granted. Logically, authorisation constraints are checked after authentication constraints because authorisation solvers use the results of authentication solvers as their inputs. Authorisation constraints are defined by predicates that are similar to those defining firing constraints.

Let us denote by $\mathsf{FUNCTIONALITY} = \{\mathsf{f}_1,\mathsf{f}_2,..,\mathsf{f}_n\}$ a set of possible functions to be invoked. Let $\mathsf{CA} = (\mathsf{P},\mathsf{F},\mathsf{S},\mathsf{I},\mathsf{O},\mathsf{A})$ be a coordination activity needing to be orchestrated and $\mathsf{N} \in \mathsf{P}$ the approuved identity of CA produced by authentication solvers. There are four authorisation constraints that define the relation between an approuved identity $\mathsf{ID}_i \in \mathsf{IDENTITY}$ and an authorized function invocation $\mathsf{f}_i \in \mathsf{FUNCTIONALITY}$ in the scope of CA.

- $\mathsf{Consider}(\mathsf{N}, \mathsf{f}_i)$: $TRUE$ if N is *under consideration* to invoke the function f_i in the scope of CA, $FALSE$ otherwise.

- Permit(N, f_i): $TRUE$ if N is *authorised* to invoke the function f_i in the scope of CA, $FALSE$ otherwise.
- Obligate(N, f_i): $TRUE$ if N must *obligatory* invoke the function f_i in the scope of CA, $FALSE$ otherwise.
- Forbid(N, f_i): $TRUE$ if it is *forbidden* for N to invoke the function f_i in the scope of CA, $FALSE$ otherwise.

4.4 Non Repudiation

A non repudiation constraint represents the situation where we can post-check the validity of coordination activities and prevent a participating service from denying previous behaviours that are related to its input or output data. Let $\{CA_1, CA_2, .., CA_n\}$ be the set of coordination activities to be orchestrated. The following non repudiation constrains can be associated to coordination scenarios:

- Ordered($\{CA_i\}$): $TRUE$ if it is possible to proof from the previous evaluations of constraints the real execution order of coordination activities, $FALSE$ otherwise.
- Approval(CA_i): $TRUE$ if from the coordination scenario we can proof who is responsible for approving the exchange of data when a function was invoked within the scope of a given coordination activity, $FALSE$ otherwise.
- Sent(CA_i, [invocation, result]): $TRUE$ if from the coordination scenario we can proof who sent an invocation and who sent the corresponding result within the scope of a given coordination activity, $FALSE$ otherwise.
- Original(CA_i): $TRUE$ if both Approval(CA_i) and Sent(CA_i) are $TRUE$, $FALSE$ otherwise.
- Received(CA_i, [invocation, result]): $TRUE$ if from the coordination scenario we can proof that the recipient received an invocation or a result, $FALSE$ otherwise.
- Known(CA_i, [invocation, result]): $TRUE$ if from the coordination scenario we can proof that the recipient recognised the content of a received invocation or result, $FALSE$ otherwise.
- Delivered(CA_i): $TRUE$ if both Received(CA_i) and Known(CA_i) proof that the recipient received and recognised the content of an invocation or a result, $FALSE$ otherwise.
- Submitted(CA_i): $TRUE$ if from the coordination scenario we can proof that a delivery authority has accepted the transmission of an invocation, $FALSE$ otherwise.
- Transported(CA_i): $TRUE$ if from the coordination scenario we can proof that a delivery authority has given the invocation or the result of a specific sender to an intended recipient, $FALSE$ otherwise.

4.5 Safety

These constraints serve to diagnose following unintended behaviours:

- a required coordination activity is not performed;
- an incorrect or unsafe coordination activity is performed;

- a required coordination activity is performed at the wrong time or at the wrong crosslink;
- a correct coordination activity is not stopped in prescribed order.

We suggest additional constraints for reinforcing the described constraints in order to ensure a correct specification of an application logic. There are intra-safety constraints of coordination activities and inter-safety constraints of coordination activities. These constraints can be used as safeguards to detect and handle unintended behaviours at execution time.

Intra-safety constraints diagnose if (i) a required functional constraint is not considered or (ii) a non required functional constraint is considered in the application logic. These constraints serve to compare and post check the execution state of coordination activities. They are verified on L, CT, OT of coordination scenarios.

Inter-safety constraints diagnose if a required functional constraint is considered at the wrong time. In this case, they serve to check the validity of a constraint corresponding with a given activity. Also, inter-safety constraints prevent the dependencies that exist between constraints of the same type. They serve to avoid the redundance and the contradiction of constraints imposed to a coordination activity.

5 Execution Policies

As shown in Figure 3, the execution of every coordination activity is done in three steps: preparation, treatment, termination. Constraints are handled as pre-conditions, post-conditions and invariants of the corresponding invocation according to the following rules:

- *Rule 1.* Preconditions must be verified and treated before firing the execution of invoked functions of a participating service. For a given coordination activity, preconditions are specified by firing, authentication and authorisation constraints. For a coordination activity (i.e. the current coordination activity) in relation with another coordination activity (i.e. next coordination activity), preconditions of the next coordination activity are ordering and data dependency constraints (see (1), (2)).
- *Rule 2.* Postconditions must be verified and treated after having the result of an invoked function. For a coordination activity post conditions are specified by non repudiation constraints. For a coordination activity (i.e., the current coordination activity) in relation with another coordination activity (i.e., next coordination activity), the postcondition of current coordination activity is specified by integrity constraints (see (1), (3)).
- *Rule 3.* Invariants are always translated as preconditions and postconditions. They must be verified and treated respectively as part of the treatment of preconditions and post conditions.

```
(1) coordination_activity := treatment |
    ( preparation ^ treatment ^ termination );
(2) preparation := precondition-checking |
    ( precondition-checking ^ exception-treatment ) |
    UNDER_CONSIDERATION | WAIT | READY;
(3) treatment := invoked_functionality-execution |
    ( invoked_functionality-execution ^ faillure-treatment ) | ACTIVE;
(4) termination := postcondition-checking |
    ( postcondition-checking ^ exception-treatment );
(5) precondition-checking := TRUE | FALSE;
(6) postcondition-checking := TRUE | FALSE;
(7) faillure-treatment := STOP | ( STOP ^ exception-treatment )|
    CONTINUE | COMPENSABLE;
(8) exception-treatment := ABORT | COMMIT | COMPENSATION;
```

Fig. 3. Handling rules of a coordination activity

- *Rule 4.* During the condition treatment process, the intra-safety and inter-safety constraints must be ensured.
- *Rule 5.* If the result of an invoked function notifies and failure, failure treatment must be executed as part of the treatment of post conditions (see (7), (8)).
- *Rule 6.* The validity of a constraint is either $TRUE$ or $FALSE$ (see (5), (6)).
- *Rule 7.* Constraints evaluation to FALSE is captured as the occurrence of an exception (see (8)).
- *Rule 8.* The matching of coordination and security strategies to the execution of the coordinated system are validated by post-checking the information extracted from coordination scenarios.

Figure 4 illustrates the state automation of a coordination activity during execution.

To glue together a set of coordination activities {CA_1, CA_2, .., CA_n}, we predefine their associated constraints as constructors. The following constructors have been identified: sequence, selection (OR-split, OR-joint), iteration, synchronisation (AND-split, AND-joint) and free (unordering).

- sequence(CA_1,CA_2) expresses a sequential order among coordination activities, i.e. the fact that the execution of one coordination activity (CA_2) is carried out after another (CA_1).
- OR-split(CA_1,{CA_2,..,CA_n}), or sequence(CA_1,CA_2) $\vee$ .. $\vee$ sequence(CA_1,CA_n), is used to select one coordination activity from a set of coordination activities CA_2,.., CA_n. CA_1 and only one of them ensure a sequential order.
- OR-joint({CA_1,..,CA_{n-1}},CA_n), or sequence(CA_1,CA_n) $\vee$ .. $\vee$ sequence (CA_{n-1},CA_n), is used to select one coordination activity from a set of coordination activities CA_1,.., CA_{n-1}. Only one of them and CA_n ensure a sequential order.

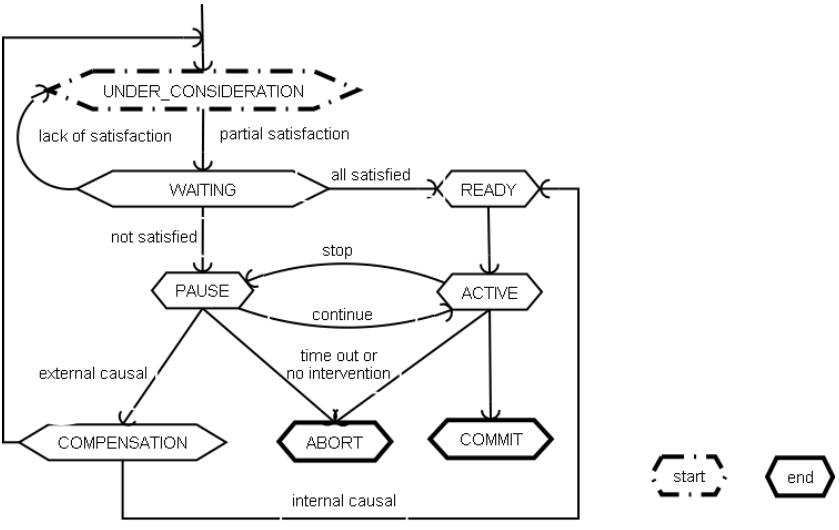


Fig. 4. State automation of coordination activity

- AND-split(CA_1,{CA_2,..,CA_n}), or sequence(CA_1,CA_2) $\wedge$ .. $\wedge$ sequence(CA_1,CA_n), expresses the fact that two or more coordination activities are executed at the same time, after the execution of another. This is an effect of synchronisation among coordination activities {CA_2,..,CA_n}. CA_1 and all of them ensure a sequential order.
- AND-joint({CA_1,..,CA_{n-1}},CA_n), or sequence(CA_1,CA_n) $\wedge$ .. $\wedge$ sequence (CA_{n-1},CA_n), expresses the fact that two or more coordination activities CA_1,.., CA_{n-1} are executed at the same time. All of them and CA_n ensure n-1 sequential orderings.
- iteration(CA_1) expresses the fact that the execution of a coordination activity CA_1 is repeated.
- unordering(CA_1, .., CA_n) expresses the fact that no coordination constraints described above are met. In other words, there is no dependency among coordination activities. Such activities can be executed at the same time or according to any order.

For example, consider the three coordination activities implementing the look up function. The flight booking application first interacts with *Adventurer* service to get information about a client and her/his needs by invoking the method get_Requirements (CA_1). This information is used by the method seek_Flights of the *Seeking* service for looking for available flights (CA_2). This service returns a list of possible flights that are displayed by the method display_Results of the *Adventurer* service (CA_3). A sequential constructor is expressed as a set of pre and post conditions of these coordination activities. The following constraints specify coordination and security aspects associated to CA_2 as pre conditions:

- Obligate(S_1 = COMMIT): once the execution status of CA_1 (i.e., S_1) is succesful, the method seek_Flights of the Seeking service can be invoked.

- Match(O_1): information about customer's needs (i.e., (O_1)) produced by the method get_Requirements of the Adventurer service is used as input data of the method seek_Flights provided by the Seeking service.
- After(CA_1): the end of the execution of the method get_Requirements must precede the beginning of the execution of the method seek_Flights.
- Approuved(searchFlights): the identity of the service providing the invoked method (seek_Flights) must belong to the list of valid names of the coordinated system.

Similarly, the following post conditions of CA_2 must hold:

- Permit(S_3 = READY): CA_3 can fire the invocation to a method of the Adventurer service. This constraint plays also the role of an authorisation constraint for CA_3.
- Invariant(O_2): the flight search result cannot be altered until it is delivered to the Adventurer service.
- Received(searchFlights, invocation) $\wedge$ Sent(searchFlights, result): it ensures that the invocation and the transmission of results are done within the same execution scope. In the example, the invocation of the method seek_Flights is received and its results are sent within the scope of the coordination activity CA_2. This constraint is used for avoiding non-repudiation.

6 Related Work

Coordination is described and managed as constraints on interactions among services. From this point of view, we consider existing related works according to the service interaction mechanisms they support.

The first category of mechanisms is based on sharing data space, naturally a common accessible data structure for all of participating services. Their communication is realized by the way they control or process common data values in sharing space. In this category, the target coordinated system configuration is supported by suitable coordination languages at run-time, i.e. the Linda family [16,17] and other varieties. [19] combines the Linda language with certain standards of W3C (XML, XSL) to build a workflow system management on the Internet. Polished [4] is a coordination language, which provides basic descriptions and automatic analysis for mobile agent system architectures. The security solution is to build a secured sharing space [7,13] corresponding with a given coordination model in a concrete context.

The second category is based on data passing among participating services. A participating service is then considered as a black box process that produces or consumes data via well defined interfaces, called ports. These boxes communicate directly in several ways: establishing connections among participating services ports for exchanging data, diffusing control events or messages among processes. The coordinated system is configured at runtime with the support of integration tools or integration environments according to specified software architecture.

This category of mechanisms can be used to manage complex distributed systems [12], especially multi-agent systems [3]. ToolBUS [15] and coordination orchestration [13] are used to facilitate and to control potential interactions between system components. A coordination protocol for distributed applications (e.g. Web services) is specified in [11]. A coordination tool based on local data space of participating Web services is also studied in [1].

Nevertheless, at the abstraction level of coordination, these works do not consider the security by the way coordination is abstracted and specified. Security and coordination are differently modelled. If we wish to consider security requirements using the concepts and terminologies of a coordination model, there is no way to explicitly determine the relation between such a coordination model and a specific security model. Security management relies upon the architecture components that implement coordinated systems.

At the execution level, existing security solutions are strictly proposed as tools for securing the execution of specific components that carry out the coordination. With regard to the first category of coordination, the execution architecture of coordination is the sharing space. So, the main security solution is to build a secured sharing space [7,13] corresponding with a given coordination model deployed in a concrete environment. Following this idea, security techniques for controlling access (authorization, control privilege, etc.) and for identifying participating services are focused. With regard to the second category, the execution architecture of coordination is a tool that supports interconnection and communication among participating services. Security measures are then applied to these tools. Efforts like WS-Policy and WS-Secure-Conversation combined with WS-Security and WS-Trust are going in this direction. [7] presents an approach for building a secure mobile agent environment. Moreover, some methodologies used to evaluate the security quality of participating services are quoted. [2] presents a specification to secure exchanged messages among Web services by using SOAP protocols. In [14], a component characterization diagram for component-based system is proposed. Based on this diagram, a formal security model to identify and quantify security properties of component functionalities is also presented. The purpose is to protect user data by evaluating and certifying the components and their composition (if they are re-used by another).

Security solutions that are defined at the abstraction level can be mapped and reused as security solutions at the execution level. For example, at the abstraction level, we use constraints to define an authentication strategy of coordination activities. At the execution level, this strategy is mapped and implemented as one component that performs the coordination. It can be reused to define an authentication strategy of participating services. In this case, it is implemented as a tool for securing the execution of components that perform the coordination.

7 Conclusion

This paper presented *MEO* - a secure coordination model that enables the specification of secure service-based coordinated systems. Both functional and non

functional requirements of such coordinated systems are specified by constraints for coordination activities and that refer to coordination scenarios.

The originality of our solution is that security requirements are considered at both abstraction and execution level. In the *MEO* model, security solutions (e.g., authentication strategy) are constructed in the same way in which coordination solutions (e.g., application logic) are constructed. For this reason, the first essential point making our proposition different from other current solutions is that we can consider the functional safety requirements early on the coordination abstraction level, while other solutions can only consider it at the execution level.

Our *MEO* model takes some advantages for defining a secure coordinated system. First, *MEO* enables the definition of security strategies without redundancy by distinguishing different levels of coordination and security. Moreover, these strategies can be independently applied to different objectives at both abstraction and execution level. Second, with *MEO* coordination and security requirements of the coordinated system are specified in the same way, so there is no need to consider the relation of terminologies and concepts amongst different coordination and security models.

Further research focuses on the construction of a secure coordination framework. It will support the building elements for coordinating services. Executing a coordination in a secure way means defining tools that control and validate the constraints specified at abstraction level. That means also supporting tools for ensuring other security objectives at execution level: exchanged message integrity, service authentication and non repudiation of coordination activities. It implies also the capability to enable the coordinated system to automatically adapt the coordination and security strategies.

References

1. P. Alvarez, J. A. Banares, P. R. Muro-Medrano, J. Nogueras, and F. J. Zarazaga. A java coordination tool for web-service architectures: The location-based service context. In *FIDJI '01: Revised Papers from the International Workshop on Scientific Engineering for Distributed Java Applications*, pages 1–14, London, UK, 2003. Springer-Verlag.
2. Khalid Belhajjame, Genoveva Vargas-Solar, and Christine Collet. Defining and coordinating open-services using workflows. In *Proceedings of the Eleventh International Conference on Cooperative Information Systems (COOPiS03)*, number 2519, Catania Sicily-Italy, Novembre 2003. Lecture Notes in Computer Science.
3. Ciaran Bryce, Manuel Oriol, and Jan Vitek. A coordination model agents based on secure spaces. In *COORDINATION '99: Proceedings of the Third International Conference on Coordination Languages and Models*, pages 4–20, London, UK, 1999. Springer-Verlag.
4. P. Ciancarini, F. Franze, and C. Mascolo. Using a coordination language to specify and analyze systems containing mobile components. *ACM Trans. Softw. Eng. Methodol.*, 9(2):167–198, 2000.
5. Workflow Management Coalition. Workflow management coalition: Terminology and glossary, 1996.

6. Microsoft Corporation. http://msdn.microsoft.com/webservices /building /interop/, 2003.
7. Marco Cremonini, Andrea Omicini, and Franco Zambonelli. Coordination in context: Authentication, authorisation and topology in mobile agent applications. In *COORDINATION '99: Proceedings of the Third International Conference on Coordination Languages and Models*, page 416, London, UK, 1999. Springer-Verlag.
8. Dimitrios Georgakopoulos, Mark F. Hornick, and Amit P. Sheth. An overview of workflow management: From process modeling to workflow automation infrastructure. *Distributed and Parallel Databases*, 3(2):119–153, 1995.
9. Object Management Group. http://www.corba.org/, 2002.
10. IBM. http://www-306.ibm.com/software/htp/cics/, 1999.
11. IBM, Microsoft, and BEA. Web services coordination. Technical report, 2003.
12. Paola Inverardi and Henry Muccini. Coordination models and software architectures in a unified software development process. In *COORDINATION '00: Proceedings of the 4th International Conference on Coordination Languages and Models*, pages 323–328, London, UK, 2000. Springer-Verlag.
13. Valerie Issarny, Christophe Bidan, and Titos Saridakis. Characterizing coordination architectures according to their non-functional execution properties. In *HICSS '98: Proceedings of the Thirty-First Annual Hawaii International Conference on System Sciences-Volume 7*, page 275, Washington, DC, USA, 1998. IEEE Computer Society.
14. K. Khan, J. Han, and Y. Zheng. Characterising user data protection of software components. In *Proceedings of the 2000 Australian Software Engineering Conference*, page 255, WCanberra, Australia, 2000. IEEE Computer Society.
15. Paul Klint and P. Olivier. The TOOLBUS coordination architecture - a demonstration. In *Algebraic Methodology and Software Technology*, pages 575–578, 1996.
16. Thomas W. Malone and Kevin Crowston. What is coordination theory and how can it help design cooperative work systems? In *CSCW '90: Proceedings of the 1990 ACM conference on Computer-supported cooperative work*, pages 357–370, New York, NY, USA, 1990. ACM Press.
17. George A. Papadopoulos and Farhad Arbab. Coordination models and languages. Technical report, Amsterdam, The Netherlands, The Netherlands, 1998.
18. James Lyle Peterson. *Petri Net Theory and the Modeling of Systems.* Prentice Hall PTR, Upper Saddle River, NJ, USA, 1981.
19. Robert Tolksdorf. Coordination technology for workflows on the web: Workspaces. In *COORDINATION '00: Proceedings of the 4th International Conference on Coordination Languages and Models*, pages 36–50, London, UK, 2000. Springer-Verlag.

Retracted: Web Service Mining and Verification of Properties: An Approach Based on Event Calculus

Mohsen Rouached, Walid Gaaloul, Wil M.P. van der Aalst, Sami Bhiri, and Claude Godart

LORIA-INRIA-UMR 7503
BP 239, F-54506 Vandœuvre-les-Nancy Cedex, France
{rouached, gaaloul, bhiri, godart}@loria.fr
Department of Technology Management, Eindhoven University of Technology
P.O. Box 513, NL-5600 MB, Eindhoven, The Netherlands
w.m.p.v.d.aalst@tm.tue.nl

Abstract. Web services are becoming more and more complexly involving numerous interacting business objects within complex distributed processes. In order to fully explore Web service business opportunities, while ensuring a correct and reliable execution, analysing and tracking Web services interactions will enable them to be well understood and controlled. The work described in this paper is a contribution to these issues for Web services based process applications.

This article describes a novel way of applying process mining techniques to Web services logs in order to enable "Web service intelligence". Our work attempts to apply Web service log-based analysis and process mining techniques in order to provide semantical knowledge about the context of and the reasons for discrepancies between process models and related instances.

1 Introduction

With the ever growing importance of the service-oriented paradigm in system architectures, more and more (business) processes will be executed using service-oriented systems. Indeed, Service Oriented Architectures (SOA) seems to be a key architecture to support BPM (Business Process Management). In particular, with SOA, an application can be now considered as a composition of services, Workflow Management Systems (WfMSs), and legacy applications. Thus, a business process becomes a set of composed services that are shared across business units, organizations, or outsourced to partners.

Currently many products that offer modeling, analysis, and simulation facilities for such business processes exist. However, one of the great advantages offered by the coupling of BPM and SOA is that designers can not only model, analyze, simulate, but they can also use the result directly for deployment, using WSBPEL for instance. Functions at the modeling layer can be linked to required services at the architecture level, and engines/systems can now manage

R. Meersman, Z. Tari et al. (Eds.): OTM 2006, LNCS 4275, pp. 408–425, 2006.

the overall business process. This is a great improvement, and it clearly shows that BPM over SOA can add value over traditional WfMSs for instance.

However, there are many challenges for trully realizing BPM over SOA. A first challenge deals with the ability to adapt and to offer self-management of the designed processes [33]. This is an important topic since these processes are quite complex and dynamic, and deviations from the expected behavior may be highly desirable. In order to fully explore business opportunities while ensuring a correct and reliable behavior, the transition from vast amounts of Internet servers and Web Services transactions data to actionable modelling intelligence would be one of the major challenges in Web Services research field. The second challenge is the need for being able to check the consistency of the business process. This can be done both statically, i.e. at design time, or dynamically, i.e. at runtime. For the dynamic part of the verification, the business process should be auditable. For that, we can use process mining techniques, because processes (and their associated services) leave many traces of their behavior in the underlying systems they used to be executed.

Obviously, the practical benefit of mining techniques depends on the quality of the available log data. Though process execution logs can be used to reveal errors and bottlenecks, they do not provide any semantical information about the reasons for the observed discrepancies. In respect to the optimization of the process models these logs therefore provide only limited information to process engineers. Consequently, the PMS (Process Management System) is unaware of the applied deviations and thus unable to log information about them. The lack of traceability of process instance changes significantly limits the benefits of process mining and data mining approaches especially in dynamic and unpredictable environment such as in web service composition.

In this paper, unlike most process mining approaches, the emphasis is not on discovery. Instead we focus on verification, i.e., given an event log we want to verify certain specified properties, to provide knowledge about the context of and the reasons for discrepancies between process models and related instances. We focus on reasoning about event logs to capture the semantics of complex Web service combinations and checking their consistency. Since services behaviors and composition specifications we are modelling are event driven, our approach uses the Event Calculus (EC) of Kowalski and Sergot [14], and an extension proposed by Mueller on Discrete EC [18], to declaratively model event based requirements specifications. Compared to other formalisms, the choice of EC is motivated by both practical and formal needs, and gives several advantages. First, in contrast to pure state-transition representations, the EC ontology includes an explicit time structure that is independent of any (sequence of) events to model event-based interactions where a number of input events may occur simultaneously, and where the system behavior may in some circumstances be non-deterministic. Second, the EC ontology is close enough to existing types of event-based requirements specifications to allow them to be mapped automatically into the logical representation. More specifically, the EC ontology is close enough to the WSBPEL specification to allow it to be mapped automatically

into the logical representation. This allows us to use the same logical foundation for verification at both design time and runtime.

The remainder of the paper is structured as follows. Section 2 describes a running example used to illustrate our ideas. The log based verification is introduced in Section 3. Then, the existing web logging techniques are discussed and the used web logger is explained. Section 5 shows how the behavioral properties can be verified using the EC formalism. The related work is discussed in section 6. Finally, Section 7 concludes the paper and outlines some future directions.

2 Running Example

Throughout this article, we will illustrate our ideas using a running example of Web services composition. We consider a car rental scenario that involves four services. A Car Broker Service (CBS) acts as a broker offering its customers the ability to rent cars provided by different car rental companies directly from car parks at different locations. CBS is implemented as a service composition process which interacts with Car Information Services (CIS), and Customer Management Service (CMS). CIS services are provided by different car rental companies and maintain databases of cars, check their availability and allocate cars to customers as requested by CBS. CMS maintains the database of the customers and authenticates customers as requested by CBS. Each Car Park (CP) also provides a Car Sensor Service (CSS) that senses cars as they are driven in or out of car parks and inform CBS accordingly. The end user can access CBS through a User Interaction Service (UIS). Typically, CBS receives car rental requests from UIS services, authorizes customers contacting CMS and checks for the availability of cars by contacting CIS services, and gets car movement information from CSS services. However, many complications may arise. For example, CBS can accept a car rental request and allocate a specific car to it if, due to the malfunctioning of a CSS service, the departure of the relevant car from a car park has not been reported and as a consequence, the car is considered to be available by the UIS service. Through this example, we aim to demonstrate how Web services logs can be specified and formalized in a way that enable the checking of some behavioral properties with respect to the composition process.

For the running example, we assume that we can log events such as the ones shown in Figure 1. Variables l_i, v_i, and c_i represent respectively the park number, the car number, and the customer identifier.

3 Log-Based Verification

3.1 Formal Specification of Composition Properties

Our framework assumes service composition processes expressed in WSBPEL [2] and uses the Event Calculus [14] to specify the properties to be monitored. In this paper, the monitorable properties may include behavioural properties of the composition process and/or assumptions that service providers can specify

activity id	service originator	timestamp
...	...	...
Enter(v1,l1)	CSS	2006-03-13 10:40:39
RelKey(v1,c1,l1)	UIS	2006-03-13 10:40:44
Available(v1,l1)	CIS	2006-03-13 10:40:48
RetKey(v1,l1)	UIS	2006-03-13 10:40:54
Enter(v2,l2)	CSS	2006-03-13 10:40:57
...	...	...
CarRequest(c1,l2)	UIS	2006-03-13 10:41:01
FindAvailable(l2,veh)	CIS	2006-03-13 10:41:02
CarHire(c1,l2,v2)	UIS	2006-03-13 10:41:03
RetKey(v2,l2)	UIS	2006-03-13 10:41:09
...	...	...

Fig. 1. A fragment of the event log

in terms of events extracted from this specification. The behavioural properties are specified in terms of: (i) events which signify the invocation of operations in different services or the composition process and responses generated at the completion of these executions, (ii) the effects that these events may have on state variables of the composition (e.g., assignment of values), and (iii) conditions about the values of state variables at different time instances. The events, effects and state variable conditions are restricted to those which can be observed during the execution of the composition process. Assumptions are additional constraints about the behaviour of individual services in the execution environment. These constraints are specified by system providers and must be expressed in terms of events, effects and state variable conditions which are used in the behavioural properties directly or indirectly.

The behavioural properties of individual web services are extracted automatically from their WSDL descriptions and the WSBPEL specification of their composition process. Following the extraction of such properties, assumptions are specified by system providers in terms of event and state condition literals that have been extracted from the WSBPEL specification and, therefore, their truth-value can be established during the execution of the composition process. The extraction of behavioural formulas from WSBPEL specifications was done in a previous work BPEL2EC developed in [28]. The BPEL2EC is a tool that takes as input the specification of a web service composition expressed in WSBPEL and produces as output a possible behavioural specification of this composition in Event Calculus. This specification can be amended by service providers, who can also use the atomic formulas of the extracted specification to additional assumptions about the composition requirements if appropriate.

Once both behavioral properties and additional assumptions are formalized, we move to annotate the execution log with semantical information to enable reasoning on recorded events for checking the consistency of the above properties and gathering reasons about deviations that may arise. This means that given an

event log and an EC property, we want to check whether the observed behavior matches the (un)expected/(un)desirable behavior.

3.2 Formulating Properties: The EC Language

Assuming that the information system left a "footprint" in some event log, it is interesting to check whether certain properties hold or not. Before being able to check such properties, a concrete language for formulating dynamic properties is needed [34]. Given the fact that we consider behavioral properties where ordering and timing are relevant and we adopt an event driven reasoning, the Event Calculus (EC) [14] seems to be a solid basis to start from.

EC is a logic-based formalism for representing and reasoning about dynamic systems. We adapts a simple classical logic form of the EC, whose ontology consists of (i) a set of time-points isomorphic to the non-negative integers, (ii) a set of time-varying properties called fluents, and (iii) a set of event types (or actions). The logic is correspondingly sorted, and includes the predicates $Happens$, $Initiates$, $Terminates$ and $HoldsAt$, as well as some auxiliary predicates defined in terms of these. $Happens(a, t)$ indicates that event (or action) a actually occurs at time-point t. $Initiates(a, f, t)$ (resp. $Terminates(a, f, t)$) means that if event a were to occur at t it would cause fluent f to be $true$ (resp. $false$) immediately afterwards. $HoldsAt(f, t)$ indicates that fluent f is true at t. The auxiliary predicate $Clipped(t1, f, t2)$ expresses whether a fluent f was terminated during a time interval $[t1, t2]$. Similarly, the auxiliary predicate $Declipped(t1, f, t2)$ expresses if a fluent f was initiated during a time interval $[t1, t2]$. For using the previous predicates, we distinguish 4 different types of events in the context of web services:

1. The **invocation** of an operation by the composition process in one of its partner services. These events are represented by terms of the form: $ic.Service.OperationName(parameters)$.
2. The **return** from the execution of an operation invoked by the composition process to a partner service. These events are represented by terms of the form: $ir.Service.OperationName(parameters)$.
3. The **reply** following the execution of an operation that was invoked by a partner service in the composition process. These events are represented by terms of the form: $re.Service.OperationName(parameters)$.
4. The **assignment** of a value to a variable. These events are represented by terms of the form $as.AssignmentName(assignmentId)$.

The above types are assumed to have instantaneous duration. We also use fluents to signify the binding of specific variables of the composition process to specific values. Thus and by using the EC ontology, some behavioral properties of the services involved in the running example are specified as shown in Figure 2. This figure shows five properties expressed in EC. These are described below.

Property P1 is about the behavior of CSS services. The variable t_m refers to the minimum time between the occurrence of two events. Then, according to this property, if a car vID is sensed to enter a car park $pID1$ at time $t1$ and

Property	EC specification
P1	$(\forall t1,t2)Happens(rc.CSS.Enter(oID1),t1)\wedge Initiates(rc.CSS.Enter(oID1),equalTo(v1,vID),t1)\wedge Initiates(rc.CSS.Enter(oID1),equalTo(p1,pID1),t1)\wedge Happens(rc.CSS.Enter(oID2),t2)\wedge(t1+t_m\leq t2)\wedge Initiates(rc.CSS.Enter(oID2),equalTo(v2,vID),t2)\wedge Initiates(rc.CSS.Enter(oID2),equalTo(p2,pID2),t2)\Longrightarrow(\exists t3)Happens(rc.CSS.Depart(oID3),t3)\wedge(t1+t_m\leq t3\leq t2-t_m)\wedge Initiates(rc.CSS.Depart(oID3),equalTo(v3,vID),t3)\wedge Initiates(rc.CSS.Depart(oID3),equalTo(p3,pID1),t3)$
P2	$(\forall t1,t2)Happens(ic.CIS.FindAvailable(oID,pID),t1)\wedge Happens(ic.CIS.FindAvailable(oID),t2)\wedge(t1\leq t2)\wedge HoldsAt(equalTo(availability(vID1),\text{"not avail"}),t2-t_m)\Longrightarrow\neg Initiates(ic.CIS.FindAvailable(oID),equalTo(vID2,vID1),t2)$
P3	$(\forall t1,t2,t3)Happens(ir.UIS.RelKey(oID1,vID),t1)\wedge Happens(ir.UIS.RelKey(oID1),t2)\wedge(t1\leq t2)\wedge Happens(ir.UIS.RetKey(oID2),t3)\wedge(t2\leq t3)\wedge Initiates(ir.UIS.RetKey(oID2),equalTo(v,vID),t3)\Longrightarrow(\forall t4)(t1<t4)\wedge(t4<t3)HoldsAt(equalTo(available(vID),\text{"not avail"}),t4)$
P4	$(\forall t1)Happens(ir.UIS.RelKey(v,c,l),t1)\wedge(\exists t2)(Happens(rc.CSS.Depart(v,l),t2)\wedge(t1\leq t2\leq t1+d*t_m))\Longrightarrow\neg(\exists t3)Happens(ic.CIS.Available(v,l)),t3)\wedge(t1+d*t_m\leq t3\leq t1+d*t_m))$
P5	$Happens(rc.UIS.CarRequest(c,l),t1)\wedge Happens(ic.UIS.FindAvailable(l,v3),t2)\wedge(t1\leq t2\leq t1+t_m)\wedge Initiates(ic.UIS.FindAvailable(l,v),equalTo(v,v3),t2)\Longrightarrow(\exists t3)Happens(ir.UIS.CarHire(c,l,v),t3)\wedge(t2\leq t3\leq t2+t_m)$

Fig. 2. Behavioural properties and assumptions of the running example

later at time $t2$ the same car is selected to enter the same or a different car park, then a *Depart* event signifying the departure of vID from $pID1$ must have also occurred between the two *enter* events. The *Happens* predicates in P1 represent the invocation of the operations *Enter* and *Depart* in CBS by CSS following the entrance and departure of cars in car parks. The *Initiates* predicates in the same formula initiate events that represent the specific value bindings of the input parameters vi and pi (i=1,2,3) of the operations *Enter* and *Depart*.

Property P2, which is about the behavior of CIS services, indicates that the operation *FindAvailable*, which is provided by the CIS service and searches for available cars at specific car parks should not return the identifier of a car to CBS unless this car is available.

The property P3 states that whilst a customer has the key of a car, this car cannot be available for rental.

The property P4 specifies that when CBS receives the event $RelKey(v,c,l)$ that signifies the release of a car key to a customer, it waits for an event signifying the exit of the car from the car park for d time units (this message is to be sent by CSS). If the latter event occurs, CRS invokes the operation $Available(v,l)$ in CIS to mark the relevant car as unavailable.

4 Web Service Logging

4.1 Web Service Collecting Solutions and Web Mining Log Structure

In this section we examine and formalize the logging possibilities in service oriented architectures which is a requirement to enable the approach described in this paper. Thus, the first step in the Web Service mining process consists of gathering the relevant Web data, which will be analyzed to provide useful information about the Web Service behaviour. We discuss how these log records could be obtained by using existing tools or specifying additional solutions. Then, we show that the mining abilities is tightly related to what of information provided in web service log and depend strongly on its richness.

Existing logging solutions provide a set of tools to capture web services logs. These solutions remain quite "poor" to mine advanced web service behaviours. That is why **advanced logging solutions** should propose a set of developed techniques that allows us to record the needed information to mine more advanced behaviour. This additional information is needed in order to be able to distinguish between web services composition instances.

4.2 Existing Logging Solutions

There are two main sources of data for Web log collecting, corresponding to the interacting two software systems: data on the Web server side and data on the client side. The existing techniques are commonly achieved by enabling the respective Web servers logging facilities. There already exist many investigations and proposals on Web server log and associated analysis techniques. Actually, papers on Web Usage Mining (WUM) [25] describe the most weel-known means of web log collection. Basically, server logs are either stored in the *Common Log Format* [1] or the more recent *Combined Log Format* [2]. They consist primarily of various types of logs generated by the Web server. Most of the Web servers support as a default option the *Common Log Format*, which is a fairly basic form of Web server logging.

However, the emerging paradigm of Web services requires richer information in order to fully capture business interactions and customer electronic behavior in this new Web environment. Since the Web server log is derived from requests resulting from users accessing pages, it is not tailored to capture service composition or orchestration. That is why, we propose in the following a set of advanced logging techniques that allows to record the additional information to mine more advanced behavior.

4.3 Advanced Logging Solutions

Identifying web service composition instance: Successful mining for advanced architectures in Web Services models requires composition (choreography/

[1] http://httpd.apache.org/docs/logs.html
[2] http://www.w3.org/TR/WD-logfile.html

orchestration) information in the log record. Such information is not available in conventional Web server logs. Therefore, the advanced logging solutions must provide for both a choreography or orchestration identifier and a case identifier in each interaction that is logged.

A known method for debugging, is to insert logging statements into the source code of each service in order to call another service or component, responsible for logging. However, this solution has a main disadvantage: we do not have ownership over third parties code and we cannot guarantee they are willing to change it on someone else behalf. Furthermore, modifying existing applications may be time consuming and error prone.

Since all interactions between Web Services happen through the exchange of SOAP message (over HTTP), an other alternative is to use SOAP headers that provides additional information on the message's content concerning **choreography**. Basically, we modify SOAP headers to include and gather the additional needed information capturing **choreography** details. Those data are stored in the special <WSHeaders>. This tag encapsulates headers attributes like: choreographyprotocol, choreographyname, choreographyinstancecase and any other tag inserted by the service to record optional information; for example, the <soapenv:choreographyprotocol> tag, may be used to register that the service was called by $WS-CDL$ choreography protocol. The SOAP message header may look as shown in Figure 3. Then, we use SOAP intermediaries [3] which are an application located between a client and a service provider. These intermediaries are capable of both receiving and forwarding SOAP messages. They are located on web services provider and they intercept SOAP request messages from either a Web service server or captures SOAP response messages from either a Web service provider. On Web service client-side, this remote agent can be implemented to intercept these messages and extract the needed information. The implementation of client-side data collection methods requires user cooperation, either in enabling the functionality of the remote agent, or to voluntarily use and process the modified SOAP headers but without changing the Web service implementation itself (the disadvantage of the previous solution).

Concerning **orchestration** log collecting, since the most web services orchestration are using a WSBPEL engine, which coordinates the various orchestration's web services, interprets and executes the grammar describing the control logic, we can extend this engine with a sniffer that captures orchestration information, i.e., the orchestration-ID and its instance-ID. This solution provides is centralized, but less constrained than the previous one which collects choreography information.

Using these advanced logging facilities, we aim at taking into account web services' neighbors in the mining process. The term neighbors refers to other Web services that the examined Web Service interacts with. The concerned levels deal with mining web service choreography interface (abstract process) through which it communicates with others web services to accomplish a choreography, or discovering the set of interactions exchanged within the context of a given choreography or composition.

$$
\begin{array}{l}
< soapenv : Header > \\
\quad < soapenv : choreographyprotocol \\
\qquad soapenv : mustUnderstand = \text{"0"} \\
\qquad xsi : type = \text{"}xsd : string\text{"} > WS - CDL \\
\quad < /soapenv : choreographyprotocol > \\
\quad < soapenv : choreographyname \\
\qquad soapenv : mustUnderstand = \text{"0"} \\
\qquad xsi : type = \text{"}xsd : string\text{"} > OTA \\
\quad < /soapenv : choreographyname > \\
\quad < soapenv : choreographycase \\
\qquad soapenv : mustUnderstand = \text{"0"} \\
\qquad xsi : type = \text{"}xsd : int\text{"} > 123 \\
\quad < /soapenv : choreographycase > \\
< /soapenv : Header >
\end{array}
$$

Fig. 3. The SOAP message header

Collecting Web service composition Instance: The focus in this section is on collecting and analysing **single** web service composition instance. The issue of identifying several instances has been discussed in the previous section. The exact structure of the web logs or the event collector depends on the web service execution engine that is used. In our experiments, where we have used the engine bpws4j [3] uses log4j[4] to generate logging events. Log4j is an OpenSource logging API developed under the Jakarta Apache project. It provides a robust, reliable, fully configurable, easily extendible, and easy to implement framework for logging Java applications for debugging and monitoring purposes. The event collector (which is implemented as a remote log4j server) sets some log4j properties of the bpws4j engine to specify level of event reporting (INFO, DEBUG etc.), and the destination details of the logged events. At runtime bpws4j generates events according to the log4j properties set by the event collector. Below we show some example log4j 'logging event' generated by bpws4j engine.

```
2006-03-13 10:40:39,[illegible]  [Thread-35]  INFO   bpws.runtime - Outgoing res-
   ponse: [WSIFResponse:serviceID = '{http://tempuri.org/services/Custom-
   erReg}CustomerRegServicefb0b0-fbc5965758--8000'operationName = ''
   isFault = 'false'   outgoingMessage = 'org.apache.wsif.base.WSIFDefau-
   ltMessage@1df3d59 name:null  parts[0]:[JROMBoolean: : true]'
   faultMessage = 'null'  contextMessage = 'null']
2006-03-13 10:40:39,634  [Thread-35]  DEBUG  bpws.runtime.bus - Response
   for external invoke is[WSIFResponse:serviceID='{http://tempuri.org/se-
   rvices /CustomerReg}CustomerRegServicefb0b0-fbc5965758--8000'
     operationName = 'authenticate'  isFault = 'false'  outgoingMessage =
   org.apache.wsif.base.WSIFDefaultMessage@1df3d59 name:null  parts[0]:
   [JROMBoolean: : true]'faultMessage = 'null'  contextMessage = 'null']
2006-03-13 10:40:39,634  [Thread-35]  DEBUG  bpws.runtime.bus - Waiting
   for request
```

[3] http://alphaworks.ibm.com/tech/bpws4j
[4] http://logging.apache.org/log4j

The event extractor captures logging event and converts it to EC events by applying regular expressions. These expressions are described below.

Regular expressions for capturing events. The following rules and assumptions are used in the regular expressions

R1 *$LogString* is the string representation of the 'logging event' received by the event receiver from the bpws4j engine

R2 *SubString($string, $substring)* extracts *$substring* at the beginning of *$string* where *$substring* can be a regular expression.

R3 *Matches($string, $substring)* returns true if *$substring* appears in *$string*, else returns false. *$substring* can be a regular expression.

R4 *StripEnds($string)* removes the first and the last character from *$string* and returns the *$string*.

R5 *$string1 + $string2* performs the concatenation of *$string1* and *$string2*.

Below, we precise how we use these rules for extracting EC formulas for two basic activities from WSBPEL:

Regular Expression for receive activity

$Happens(rc.ServiceName.operation(vID, var1, var2), t)$ is an EC predicate that corresponds to a *receive* activity. A logging event from bpws4j that corresponds to a *receive* activity looks as follows

```
2006-03-13 11:41:59,714[Thread-34]
   DEBUG  bpws.runtime.bus Invoking external service with[WSIFRequest:se
   rviceID='{http://tempuri.org/services/CarReg}CarRegServicefb0b0-fbc59
   65758-8000'operationName='isAvailable'incomingMessage='org.apache.wsi
   f.base.WSIFDefaultMessage@1[...]423name:nullparts[0]:[JROMString:loc:One
   ]'contextMessage='null']
```

In this event [http8080-Processor25] is the unique ID assigned by the bpws4j engine to this instance of the *receive* activity and its corresponding *reply* activity. Applying the above assumptions and rules the event extractor generates the EC event as follows

```
if Matches($LoString, "http[0-9][0-9][0-9][0-9]-Processor[0-9]*") and
Matches($LogString, "operation") and Matches($LogString, "var1")
and Matches($LogString, "var2") then return
"Happens(rc.operation(" +  SubString($LogString, "http[0-9][0-9]
[0-9][0-9]-Processor[0-9]*") + StripEnds(SubString(SubString($LogString,
"var1: [0-9A-Za-z]*]"), " [0-9A-Za-z]*]")) + StripEnds(SubString(
SubString($LogString, "var2: [0-9A-Za-z]*]"), " [0-9A-Za-z]*])) + ")"
+ SubString($LogString, "[0-9][0-9][0-9][0-9]-([0][0-9]|[1][0-2])-[0-3]
[0-9] ([0-1][0-9]|[2][0-4]):[0-5][0-9]:[0-5][0-9],[0-9][0-9][0-9]) +
"R(SubString($LogString, "[0-9][0-9][0-9][0-9]-([0][0-9]|[1][0-2])-[0-3]
[0-9] ([0-1][0-9]|[2][0-4]):[0-5][0-9]:[0-5][0-9],[0-9][0-9][0-9]),
SubString($LogString,"[0-9][0-9][0-9][0-9]-([0][0-9]|[1][0-2])-[0-3]
[0-9]([0-1][0-9]|[2][0-4]):[0-5][0-9]:[0-5][0-9],[0-9][0-9][0-9]))" + ")"
```

The rest of the extraction schemes are analogous to those of *receive* activity. By doing this, the event log fragment 2 can be represented in formalised manner as shown in Figure 4.

L1 : $Happens(rc.CSS.Enter(v1, l1), 1)$
L2 : $Happens(rc.UIS.RelKey(v1, c1, l1), 5)$
L3 : $Happens(ic.CIS.Available(v1, l1), 9)$
L4 : $Happens(rc.UIS.RetKey(v1, l1), 15)$
L5 : $Happens(rc.CSS.Enter(v2, l2), 18)$
L6 : $Happens(rc.UIS.RetKey(v2, l2), 23)$
L7 : $Happens(ic.CIS.Available(v2, l2), 26)$
L8 : $Happens(rc.CSS.Enter(v1, l1), 27)$
L9 : $Happens(rc.UIS.RelKey(v2, c2, l2), 29)$
L10 : $Happens(ic.CIS.Available(v2, l2), 34$
L11 : $Happens(rc.UIS.CarRequest(c1, l2), 49)$
L12 : $Happens(ic.CIS.FindAvailable(l2, veh$, 50)
L13 : $Happens(ir.CIS.FindAvailable(l2, veh$ 51)
L14 : $Initiates(ir.CIS.FindAvailable(l$, v 51
L15 : $Happens(re.UIS.CarHire(c1, l2$ v2) 52
L16 : $Happens(rc.UIS.RetKey($ 2),
...

Fig. 4. The CRS Event Log

5 Verifying Properties

5.1 The EC Checking

Given the properties specification shown in Figure 2 and the event log of Figure 4, the property P2 is found to be inconsistent with the expected behavior of CBS at t=54. The inconsistency arises because the literals L13 and L14 in Figure 2 and the literal $HoldsAt(equalTo(availability(v2), "not\ avail"), 50)$, which is derived from the literals L9 and L16 and the property P3 entail the negation of P2. In this example, the inconsistency is caused by the failure of the CSS service to send an $R.CSS.Depart(v2, l2)$ event to CBS following the event $Happens(Q.UIS.RelKey(v2, c2, l2), 28)$. Thus, according to P4, CBS invoked the operation *Available* to mark the vehicle v2 as available (see the literal L10 in Figure 4). Subsequently, when the operation $Q.CIS.FindAvailable(l2, v)$ was invoked in CIS (see literal L12), CIS reported v2 as an available vehicle. Note, however, that this inconsistency could only be spotted after the event signified by the literal L16 and by virtue of P4 (according to P4, a car whose key is released should not be considered as available until the return of its key).

One other case is that at t=54, the event L15 which was generated due to P5 can be detected as unjustified behavior. This is because this event can only have been generated by P5. Note that, although in this case CBS has functioned according to P5, one of the conditions of this property is violated by the literal $Initiates(R.CIS.FindAvailable(l2, v), equalTo(v, v2), 51)$. This literal can be

deduced from P2, the literal L13, and the literal $HoldsAt(equalTo(availability(v2), not\ avail), 50)$. The latter literal is deduced from L9 and L16 and assumption P3.

The property P2 is violated by the expected behaviour of CBS. According to this property, the operation *FindAvailable*, which is provided by the CIS service of CRS and searches for available cars at specific car parks should not return the identifier of a car to CBS unless this car is available. The violation of P2 in this case occurs since from P3 we can derive that $v2$ could not be available from T=30 when its key was released (see literal L9 in Figure 4) until T=53 (that is one time unit before its key was returned back - see literal L16 in Figure 4). Nevertheless, the execution of the operation *FindAvailable* of the CIS service at T=51 reported $v2$ as an available vehicle (see literal L14 in Figure 4).

5.2 Implementation Issues

In order to ensure an efficient satisfiability encoding for the EC, Mueller [19] presented an alternative classical logic axiomatization of the Event Calculus called *Discrete Event Calculus*, by restricting the time point sort to integers. Then, Mueller shows how Discrete Event Calculus problems can be encoded in first-order logic, and solved using a first-order logic automated theorem proving system, and developed a tool called the **Discrete Event Calculus Reasoner** (DEC Reasoner) [5].

Our approach has been implemented in Java and incorporates the following components: a requirements (behavioural properties and assumptions) editor, an event collector, a BPEL2EC tool and a deviation viewer. Behavioural properties are extracted according to the patterns that we describe in [28] and represented in an XML-based language that we have defined to represent EC formulas.The properties extractor also identifies events, effects and state variables in the composition process that provide the primitive constructs for specifying further assumptions about the behaviour of the composition. These assumptions are specified by service providers using the assumption editor as shown in Figure 5.

The assumption editor offers to service providers the different types of events and fluent initiation predicates that have been identified in the composition process and supports the specification of assumptions as logical combinations of these event and fluent initiation predicates. Service providers may also use the editor to define additional fluents to represent services, service states, and relevant initiation and holding predicates. When an assumption is specified, the assumption editor can check its syntactic correctness. Figure 5 presents the steps in specifying assumptions using the assumption editor. The BPEL2EC tool is built as a parser that can automatically transform a given WSBPEL process into EC formulas according to the transformation scheme detailed in [28]. It takes as input the specification of the Web service composition as a set of coordinated web services in WSBPEL and produces as output the behavioral specification of this composition in Event Calculus. This specification can be amended by

[5] http://decreasoner.sourceforge.net

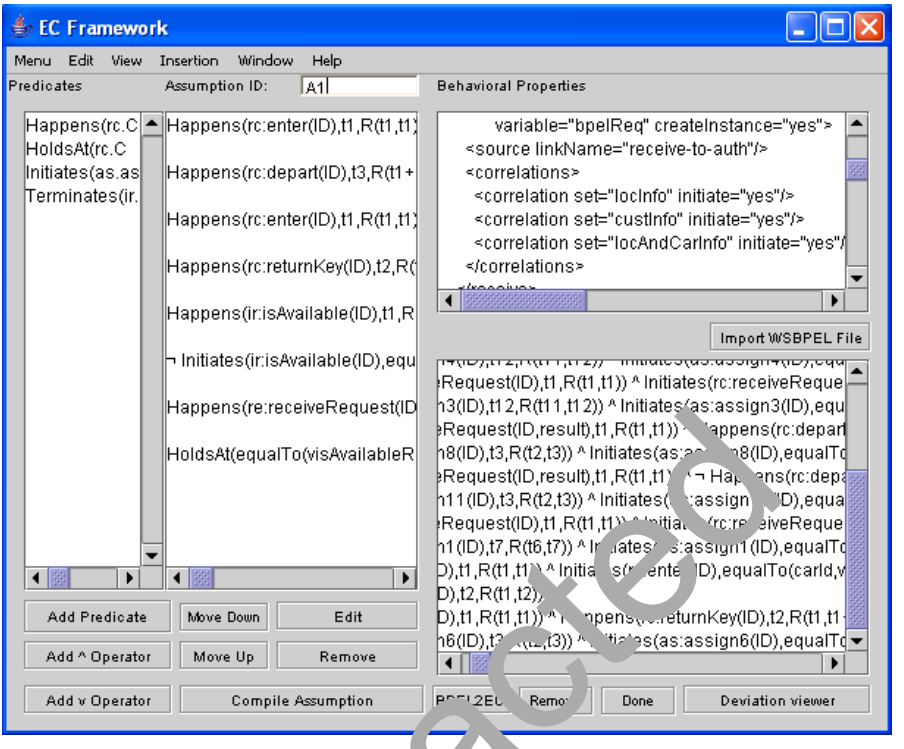


Fig. 5. The principal snapshot of the Monitoring Framework

the service providers to consider additional assumptions about the operations if appropriate.

While executing the composition process, the process execution engine generates events which are sent as during streams to the event collector of our framework. Irrelevant events are determined by the formulas that have been extracted or specified for monitoring by the service provider. Then, the EC checker (DEC for instance) processes the events which are recorded in the log by the event collector in the order of their occurrence, identifies other expected events that should have happened but have not been recorded (these events are derived from the assumptions by deduction), and checks if the recorded and expected events are compliant with the behavioural properties and assumptions of the composition process. In cases where the recorded and expected events are not consistent with these requirements, the EC checker records the deviation in a log deviations. The framework incorporates also a deviation viewer that is used to browse the detected violations of the formulas. A snapshot of this viewer is shown in Figure 6.

To evaluate our monitoring framework we performed a series of experiments in which we used an implementation of the CRS example as a case study. In this case study, we extracted 7 behavioural properties from the WSBPEL specification

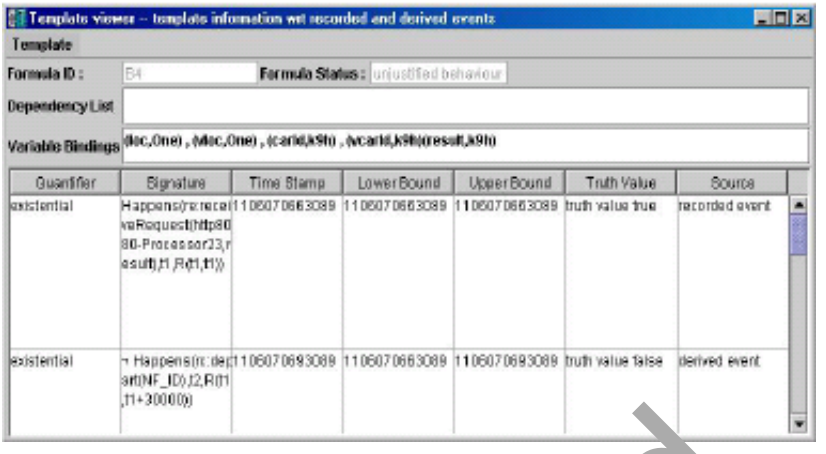


Fig. 6. Deviation Viewer

of the composition process of CRS and specified 4 assumptions. The WSBPEL process of our case study and the behavioural properties and assumptions specified for it can be found at `http://www.loria.fr/~rouached/crs.zip`.

6 Related Work

Several attempts have been made to capture the behavior of BPEL [1] in some formal way. Some advocate the use of finite state machines [9], others process algebras [8], and yet others abstract state machines [7] or Petri nets [23,16,32]. Another branch of work concerning the area of "adapting Golog for composition of semantic web services" is carried out by Sheila McIlraith and others [17]. They have shown that Golog might be a suitable candidate to solve the planning problems occurring when services are to be combined dynamically at run-time. Additionally they propose to "take a bottom-up approach to integrating Semantic Web technology into Web services".

The need for monitoring web services has been raised by other researchers. For example, several research groups have been experimenting with adding monitor facilities via SOAP monitors in Axis `http://ws.apache.org/axis/`. [15] introduces an assertion language for expressing business rules and a framework to plan and monitor the execution of these rules. [4] uses a monitoring approach based on BPEL. Monitors are defined as additional services and linked to the original service composition. In [11,6], Dustdar et al. discuss the concept of web services mining and envision various levels (web service operations, interactions, and workflows) and approaches. Our approach fits in their framework and shows that web services mining is indeed possible, especially when using the existing process mining techniques. Related to the work in this paper is the work on conformance checking using Petri nets and event logs. In [35] abstract BPEL is

mapped onto Petri nets and the resulting Petri net is compared with the event logs based on SOAP messages. ProM's conformance checker [29] is used to do this comparison and the approach has been tested using Oracle BPEL. To give a complete overview of process mining, we refer to a special issue of Computers in Industry on process mining [38] and a survey paper [37]. Process mining can be seen in the broader context of Business (Process) Intelligence (BPI) and Business Activity Monitoring (BAM). In [12,30] a BPI toolset on top of HP's Process Manager is described. The BPI toolset includes a so-called "BPI Process Mining Engine". In [20] Zur Muehlen describes the PISA tool which can be used to extract performance metrics from workflow logs. In [24] a tool named the Web Service Navigator is presented to visualize the execution of web services based on SOAP messages. The authors use message sequence diagrams and graph-based representations of the system topology.

Formal verification of Web Services is addressed in several papers. The SPIN model-checker is used for verification [21] by translating Web Services Flow Language (WSFL) descriptions into Promela. [13] uses a process algebra to derive a structural operational semantics of BPEL as a formal basis for verifying properties of the specification. In [10], BPEL processes are translated to Finite State Process (FSP) models and compiled into a Labelled Transition System (LTS) in inferring the correctness of the Web service compositions which are specified using message sequence charts. In [22], Web services are verified using a Petri Net model generated from a DAML-S description of a service.

One common pattern of the above approaches is that they adapt static verification techniques and therefore violations of requirements may not be detectable. This is because Web services that constitute a composition process may not be specified at a level of completeness that would allow the application of static verification, and some of these services may change dynamically at run-time causing unpredictable interactions with other services.

The Event Calculus has been theoretically studied. Denecker et al. [5] use the Event Calculus for specifying process protocols using domain propositions to denote the meanings of actions. In [31] the Event Calculus has been used in planning. Reasoning in the Event Calculus is an abductive reasoning process through resolution theorem prover. [39] develops an approach for formally representing and reasoning about business interactions in the Event Calculus. The approach was applied and evaluated in the context of protocols, which represent the interactions allowed among communicating agents. Our previous work [27] is close enough to the current work. It presents an event-based framework associated with a semantic definition of the commitments expressed in the Event Calculus, to model and monitor multi-party contracts. This framework permits to coordinate and regulate Web services in business collaborations. Our paper [26] advocates an event-based approach for Web services coordination. We focused on reasoning about events to capture the semantics of complex Web service combinations. Then we present a formal language to specify composite events for managing complex interactions amongst services, and detecting inconsistencies that may arise at run-time.

7 Conclusion and Future Directions

The paper presents a novel approach that uses event logs to enable the verification of behavioral properties in web service composition. The properties to be monitored are specified in event calculus. The functional requirements are initially extracted from the specification of the composition process that is expressed in WSBPEL. This ensures that they can be expressed in terms of events occurring during the interaction between the composition process and the constituent services that can be detected from the log of the execution. Then, we have introduced the idea of Web services mining that makes use of the findings in the fields of process mining and apply these techniques to the context of Web services and service-oriented architectures. The main focus has not been put on discovery but on verification. This means that given an event log and a formal property, we check whether the observed behavior matches the (un)expected/(un)desirable behavior.

Future work will aim at the implementation of the approach in the context of ProM framework. Specifically, we will try to implement a plugin that permits to extract and adapt web service logs (using log4j for instance) to ProM's log format, and to express behaviour properties and assumptions about web service composition in a manner that enable the use (or possibly an extension) of ProM's verification plugins. Moreover, we are also aiming at languages that are more declarative and based on temporal logic. An example is the DecSerFlow: a Declarative Service Flow Language based on LTL [36].

References

1. T. Andrews, F. Curbera, H. Dholakia, Y. Goland, J. Klein, F. Leymann, K. Liu, D. Roller, D. Smith, S. Thatte, I. Trickovic, and S. Weerawarana. Business Process Execution Language for Web Services, Version 1.1. Standards proposal by BEA Systems, International Business Machines Corporation, and Microsoft Corporation, 2003.
2. A. Arkin, S. Askary, B. Bloch, and F.Curbera. Web services business process execution language version 2.0. Technical report, OASIS, December 2004.
3. M. Baglioni, U. Ferrara, A. Romei, S. Ruggieri, and F. Turini. Use soap-based intermediaries to build chains of web service functionality, 2002.
4. L. Baresi, C. Ghezzi, and S. Guinea. Smart Monitors for Composed Services. In *ICSOC '04: Proceedings of the 2nd International Conference on Service Oriented Computing*, pages 193–202, New York, NY, USA, 2004. ACM Press.
5. M. Denecker, L. Missiaen, and M. Bruynooghe. Temporal reasoning with abductive event calculus. In *Proceedings of the 10th European Conference and Symposium on Logic Programming (ECAI)*, pages 384–388, 1992.
6. S. Dustdar, R. Gombotz, and K. Baina. Web Services Interaction Mining. Technical Report TUV-1841-2004-16, Information Systems Institute, Vienna University of Technology, Wien, Austria, 2004.
7. D. Fahland and W. Reisig. ASM-based semantics for BPEL: The negative control flow. In D. Beauquier and E. Börger and A. Slissenko, editor, *Proc. 12th International Workshop on Abstract State Machines*, pages 131–151, Paris, France, March 2005.

8. A. Ferrara. Web services: a process algebra approach. In *ICSOC '04: Proceedings of the 2nd international conference on Service oriented computing*, pages 242–251, New York, NY, USA, 2004. ACM Press.
9. J. Fisteus, L. Fernández, and C. Kloos. Formal verification of BPEL4WS business collaborations. In K. Bauknecht, M. Bichler, and B. Proll, editors, *Proceedings of the 5th International Conference on Electronic Commerce and Web Technologies (EC-Web '04)*, volume 3182 of *Lecture Notes in Computer Science*, pages 79–94, Zaragoza, Spain, Aug. 2004. Springer-Verlag, Berlin.
10. H. Foster, S. Uchitel, J. Magee, and J. Kramer. Compatibility verification for web service choreography. In *ICWS '04: Proceedings of the IEEE International Conference on Web Services (ICWS'04)*, page 738, Washington, DC, USA, 2004. IEEE Computer Society.
11. R. Gombotz and S. Dustdar. On Web Services Mining. In M. Castellanos and T. Weijters, editors, *First International Workshop on Business Process Intelligence (BPI'05)*, pages 58–70, Nancy, France, September 2005.
12. D. Grigori, F. Casati, M. Castellanos, U. Dayal, M. Sayal, and M. Shan. Business Process Intelligence. *Computers in Industry*, 53(3):321–343, 2004.
13. M. Koshina and F. van Breugel. Verification of business processes for web services. Technical report, New York University, SFUCMPT-TR-2003-03.
14. R. Kowalski and M. J. Sergot. A logic-based calculus of events. *New generation Computing 4(1)*, pages 67–95, 1986.
15. A. Lazovik, M. Aiello, and M. Papazoglou. Associating Assertions with Business Processes and Monitoring their Execution. In *ICSOC '04: Proceedings of the 2nd International Conference on Service Oriented Computing*, pages 94–104, New York, NY, USA, 2004. ACM Press.
16. A. Martens. Analyzing Web Service Based Business Processes. In M. Cerioli, editor, *Proceedings of the 8th International Conference on Fundamental Approaches to Software Engineering (FASE 2005)*, volume 3442 of *Lecture Notes in Computer Science*, pages 19–33. Springer-Verlag, Berlin, 2005.
17. S. McIlraith and T. Son. Adapting golog for composition of semantic web services. In *Proc of the 8th International Conference on Principles of Knowledge Representation and Reasoning*, 2002.
18. E. T. Mueller. Event calculus reasoning through satisfiability. *J. Log. and Comput.*, 14(5):703–730, 2004.
19. E. T. Mueller. Event calculus reasoning through satisfiability. *J. Log. and Comput.*, 14(5):703–730, 2004.
20. M. Mühlen and M. Rosemann. Workflow-based Process Monitoring and Controlling - Technical and Organizational Issues. In R. Sprague, editor, *Proceedings of the 33rd Hawaii International Conference on System Science (HICSS-33)*, pages 1–10. IEEE Computer Society Press, Los Alamitos, California, 2000.
21. S. Nakajima. Verification of web service flows with model-checking techniques. In *CW*, pages 378–385, 2002.
22. S. Narayanan and S. A. McIlraith. Simulation, verification and automated composition of web services. In *WWW '02: Proceedings of the 11th international conference on World Wide Web*, pages 77–88, New York, NY, USA, 2002. ACM Press.
23. C. Ouyang, W. Aalst, S. Breutel, M. Dumas, , and H. Verbeek. Formal Semantics and Analysis of Control Flow in WS-BPEL. BPM Center Report BPM-05-15, BPMcenter.org, 2005.
24. W. Pauw, M. Lei, E. Pring, L. Villard, M. Arnold, and J. Morar. Web Services Navigator: Visualizing the Execution of Web Services. *IBM Systems Journal*, 44(4):821–845, 2005.

25. J. Punin, M. Krishnamoorthy, and M. Zaki. Web usage mining: Languages and algorithms. In *Studies in Classification, Data Analysis, and Knowledge Organization.* Springer-Verlag, 2001.
26. M. Rouached and C.Godart. An event based model for web services coordination. In *2nd International Conference on Web Information Systems and Technologies (WEBIST 2006)*, pages 384–388. Setubal, Portugal, 11-13 April 2006.
27. M. Rouached, O. Perrin, and C. Godart. A contract-based approach for monitoring collaborative web services using commitments in the event calculus. In *Sixth International Conference on Web Information Engineering System (WISE05)*, pages 426–434, 2005.
28. M. Rouached, O. Perrin, and C. Godart. Towards formal verification of web service composition. In *Forth International Conference on Business Process Management (BPM06)*, 2006.
29. A. Rozinat and W. M. P. van der Aalst. Conformance testing: Measuring the fit and appropriateness of event logs and process models. In *Business Process Management Workshops*, pages 163–176, 2005.
30. M. Sayal, F. Casati, U. Dayal, and M. Shan. Business Process Cockpit. In *Proceedings of 28th International Conference on Very Large Data Bases (VLDB'02)*, pages 880–883. Morgan Kaufmann, 2002.
31. M. Shanahan and M. Witkowski. Event calculus planning through satisfiability. *J. Log. and Comput.*, 14(5):731–745, 2004.
32. C. Stahl. Transformation von BPEL4WS in Petrinetze (In German). Master's thesis, Humboldt University, Berlin, Germany, 2004.
33. W. M. P. van der Aalst, H. T. de Beer, and B. F. van Dongen. Process mining and verification of properties: An approach based on temporal logic. In *OTM Conferences (1)*, pages 130–147, 2005.
34. W. M. P. van der Aalst, H. T. de Beer, and B. F. van Dongen. Process mining and verification of properties: An approach based on temporal logic. In *OTM Conferences (1)*, pages 130–147, 2005.
35. W. M. P. van der Aalst, M. Dumas, C. Ouyang, A. Rozinat, and H. Verbeek. Choreography Conformance Checking: An Approach based on BPEL and Petri Nets (extended version). BPM Center Report BPM-05-25, BPMcenter.org, 2005.
36. W. M. P. van der Aalst and M. Pesic. Specifying, Discovering, and Monitoring Service Flows: Making Web Services Process-Aware. BPM Center Report BPM-06-09, BPMcenter.org, 2006.
37. W. M. P. van der Aalst, B. van Dongen, J. Herbst, L. Maruster, G. Schimm, and A. Weijters. Workflow Mining: A Survey of Issues and Approaches. *Data and Knowledge Engineering*, 47(2):237–267, 2003.
38. W. M. P. van der Aalst and A. Weijters, editors. *Process Mining*, Special Issue of Computers in Industry, Volume 53, Number 3. Elsevier Science Publishers, Amsterdam, 2004.
39. P. Yolum and M. P. Singh. Reasoning about commitments in the event calculus: An approach for specifying and executing protocols. *Annals of Mathematics and Artificial Intelligence*, 42(1-3):227–253, 2004.

Establishing a Trust Relationship in Cooperative Information Systems

Julian Jang, Surya Nepal, and John Zic*

ICT Centre PO Box 76 Epping NSW 1710 Australia
{Julian.Jang, Surya.Nepal, John.Zic}@csiro.au

Abstract. One method for establishing a trust relationship between two servers in a co-operative information system is to use a mutual attestation protocol based on hardware that implements the Trusted Computing Group's TPM specification. It has been our experience in developing an eHealth demonstration system that the efficiency of such a protocol was relatively low. This inefficiency was a result of the high number of TPM function calls in response to the large number of protocol messages that must be sent by the end server systems to establish mutual trust between them prior to sending each application message (in our case, a medical record). In order to address this inefficiency, we developed a session-based mutual attestation protocol, where multiple application messages are sent over an interval of time where an established trust relationship holds. Moreover, the protocol partially addresses the security flaw due to the time interval between the time-of-attestation and time-of-use. This paper presents this new protocol, once again utilizing TPM microcontroller hardware, and compares its performance with that of our previous (per record) mutual attestation protocol.

1 Introduction

Cooperative information systems, such as those found in government, medical and research organizations, facilitate the exchange and sharing of information between participating organizations. However, the ease with which new technologies allow easy sharing is counterbalanced by policy and legislative requirements on ensuring privacy and control of that information is maintained at all times by the custodians for the respective owners. It is also widely recognized that authentication, security, authorization and access control are major issues in cooperative information systems.

Fundamentally, these issues are all related to the establishment of mutual trust between the major components, including the end users, through the respective elements of the cooperative information systems. Although trust establishment and related issues are applicable to most cooperative information systems such as homeland security and finance, we will draw on our experiences in developing a secure consent-based medical client and utilize a cooperative healthcare system as an example through out this paper. In particular, we concentrate on the development of a hardware-supported mutual attestation protocol for healthcare providers' hosts.

* Authors are listed in an alphabetical order.

R. Meersman, Z. Tari et al. (Eds.): OTM 2006, LNCS 4275, pp. 426 – 443, 2006.

The issue of establishing and maintaining a trust relationship between remote hosts has been recognized and addressed by bodies such as the Trusted Computing Group (TCG) [1]. The TCG has defined a set of best-practice specifications for networking, storage, PC clients and hardware. In particular, the TCG defined a Trusted Platform Module (TPM), which is a cryptographic microcontroller system, and allowed the TPM to be a root of trust for authenticating the hardware and software configuration of the host computer. The TCG specifies that there are three main features that comprise a "trusted platform": *protected capabilities*, *attestation*, and *integrity measurement and reporting*. This paper is concerned primarily with attestation, even though our demonstration system incorporated the remaining feature set required to meet a trusted platform specification. Attestation allows verification, according to a specification, of the hardware and software configuration of a remote host. Trust is increased that the local host is in contact with the correct, expected remote host if the local host can both authenticate and successfully complete the attestation of the remote host.

The TCG remote attestation protocol is designed to establish one-way trust relationship, with the sending host establishing trust with a receiving host. As reported in another paper [5], we extended the TCG remote attestation protocol to allow mutual attestation of hosts in our cooperative healthcare prototype system. We refer to this protocol as *message-based mutual attestation protocol*, as it was designed to be run for each message exchanged between the end hosts. We make the following two observations, based on our experiences in using the protocol in our system. First, it is inefficient as the number of protocol (overhead) messages per application message is high (of the order of eight pairs of protocol messages per application message). As a result, there will be a large number of TPM function calls that are quite expensive. This is particularly noticeable in the cases such as the transfer of patient records from one hospital to another. Second, the difference in the time at which the attestation is completed and the time at which it is used to send the application message makes it vulnerable to attack.

To overcome these shortcomings in this application scenario, we propose a *session-based mutual attestation protocol*. As the name suggests, this protocol establishes a trust relationship between interacting hosts based on an application session rather than for each application message exchanged. Once this trust relationship is bound to the end hosts and the particular application session, any number of application messages can be sent and received (provided that they can be allocated to the session) without any further exchange and verification of the security context. Session-based mutual attestation locks the hardware and software environment for the duration of the session using the TPM "seal" operation [1]. Violations can be readily detected on the attested code/environment using the unseal operation.

It is important to note that this session-based mutual attestation protocol is not meant for general application in arbitrary domains. One of the fundamental design assumptions for our session-based mutual attestation protocol is that the software and hardware configurations of the host systems do not change much over time, and that these changes are over a relatively long time. Examples of these include server-to-server applications, where a session-based mutual attestation is highly suitable.

It should be noted that adopting a session oriented approach implies that there needs to be a session termination protocol in place. We have taken a pragmatic approach by adopting the termination protocol specification of WS-BusinessActivity [2].

The rest of the paper is organized as follows. We first present a motivating scenario in the context of cooperative healthcare information systems in Section 2. In Section 3, we review and present a performance and security analysis of the message-based mutual attestation protocol. Section 4 presents a description of our session-based mutual attestation protocol, and a performance comparison and analysis with respect to the message-based attestation protocol, both implemented on our demonstration system. Section 5 describes related work on different approaches of establishing mutual trust. The last section draws our analysis on the session-based protocol and presents some concluding remarks.

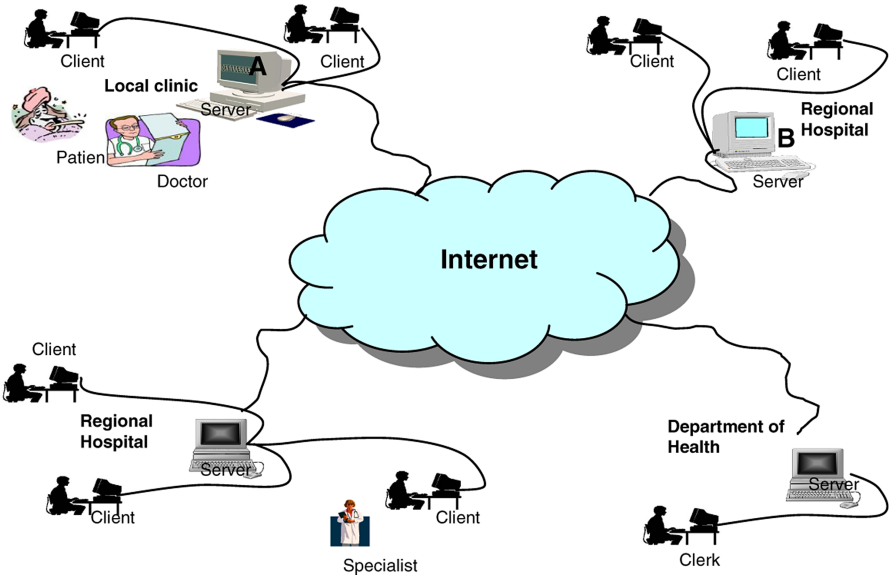


Fig. 1. Distributed Cooperative Healthcare Environment

2 Context and Motivation

The effective coordination of individual's health care relies on the sharing of personal health information among autonomous, independent healthcare providers such as local clinics, test laboratories and hospitals as shown in Figure 1 (we refer these providers as facilities). However, the ease of sharing of personal medical information must be counterbalanced with the need to ensure the privacy and security requirements of the patient and facility. The lack of such mechanisms may have serious consequences for patients; for example, a leaked patient record to an insurance company may block him from buying insurance products.

In order to address the issue of control of private medical information, the CSIRO ICT Centre developed an eConsent model [7] within the context of the Australian Government Department of Health and Aging (DoHA) Electronic Consent Project [6]. The term *eConsent* was coined to refer to a mechanism through which patients can

express their consent policies on their electronic records being accessed and shared between healthcare facilities. The eConsent model was developed and demonstrated to stakeholders from the Australian healthcare sector along with other three research and development projects commissioned by DoHA. The issue of mutual trust has been addressed in the eConsent model fundamentally relying on someone at the receiver facilities for appropriate expression of patient consent in the receiver facility's system.

We later enhanced the trust between two cooperating healthcare providers in eConsent model by using a message-based mutual attestation protocol that utilizes the TPM devices and associated protocols between two server applications [5]. We presented the architecture of the new demonstrator built using TPM devices and explained the mutual attestation protocol between two healthcare facility server applications. We also reported our integrity measurement architecture that uses existing hardware and software infrastructure.

When we applied the mutual attestation protocol to our e-consent application, we observed that in a number of cases the protocol performs poorly, incurring a noticeable delay in the transfer of medical records from one facility's server to another. We illustrate these cases using the scenario shown in Figure 1. These cases occur when sending a large number of records from one facility to another facility. This commonly occurs, for example, when a local clinic sends patients to nearby regional hospitals to perform tests. Large transfers occur at the conclusion of patients' tests, where the regional hospital will send the patients' records back to a local clinic at the end of the day. Again, a large number of records are usually transferred when a patient moves from one regional hospital to another. The message-based mutual attestation protocol was designed to exchange a message between two cooperating systems and runs for each message sent/received by them. The number of runs necessary for this protocol can be reduced if a trust relationship can be established for an entire application session such as those given above.

We also observed that there is a possible security breach between the time when one facility reports the measurement of its hardware and software configuration to another facility to the time when the receiving facility verifies the measurement. The rational is that the measured hardware and software of the sending facility could have been compromised while the receiving facility is verifying the measurement and subsequently any application data sent based on the result of the measurement.

In order to address issues of performance and security breach, we enhanced our message-based mutual attestation protocol, and developed and implemented a session-based mutual attestation protocol, which is the major focus of this paper.

3 Message-Based Mutual Attestation Protocol

In this section, we review the message-based mutual attestation protocol, focusing on trusted measurement and trusted reporting. We refer readers to [5] for the full description of the protocol and its implementation in our e-consent application. We also analyze and report our findings on this protocol's performance.

3.1 Review of Message-Based Mutual Attestation Protocol

Remote attestation allows a platform to verify the hardware and software running on remote host to decide whether or not it can trust the hardware and software configuration of remote host while protecting its privacy [1]. The TCG has proposed that remote attestation be based on two primary mechanisms.

- *Trusted measurement* where the platform characteristics such as boot loader, operating system kernel, and application executables are measured. These measurements, as well as their cryptographic hashes, are stored in the TPM hardware memory. In particular, the measured values are stored in protected registers called Platform Configuration Registers (PCRs). The platform measurement process starts at boot time of the host, then proceeds onto measuring the operating system loading and finally to the loading of applications.
- *Trusted reporting* of the trusted measurements is done by the TPM microcontroller and returns the value of a PCR digitally signed with the TPM's public attestation identity key (AIK). The AIK is created by the owner of the TPM hardware after the platform's credentials have been certified and the AIK is signed by a certifying authority such as TPM manufacturer, system manufacturer or dedicated certifying authorities.

Together these provide the necessary assurance mechanisms that a host is functioning correctly and conforms to a specific configuration. A remote host can verify the digital signature on the trusted report using the public keys of the TPM certifying authorities and then compare the PCR values against its set of trusted configurations. These two mechanisms were used in the development of the "Mutual Attestation Protocol" that allows the establishment of bidirectional trust relationship between two healthcare providers as shown in Figure 2.

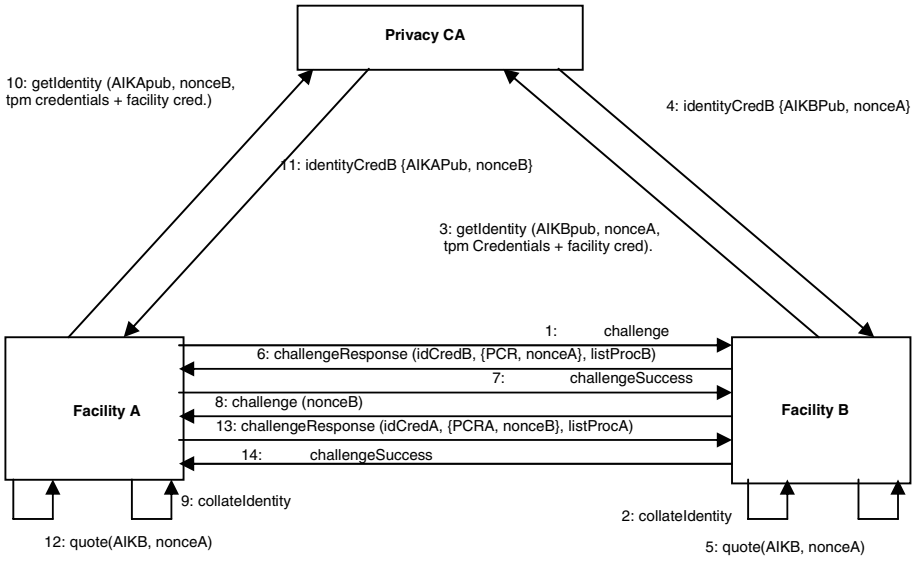


Fig. 2. Message-based mutual attestation protocol

In our implementation, we use a dedicated loader (MedicLoader) as a root of trust due to the lack of appropriate operating system support. The loader intercepts the loaded and unloaded applications, and performs measurements on the executables before their execution. A fundamental assumption is that the loader can be trusted to securely measure itself and the processes it loads before executing them. This means all components that are under this root of trust must also be trusted.

We implemented a privacy certifying authority (CA) that can verify the platform credentials as well by signing valid AIK. Also, each healthcare facility must provide a list, composed of tuples, consisting of executable names and the expected cryptographic hashes (calculated by the trusted measurement) for these executables in both loaded and unloaded cases. The trusted reporting is done when each facility sends off a CA identity credential, signed AIK by privacy CA, signed PCR values, and the list of executables along with calculated loaded and unloaded hash values. The detailed description of the mutual attestation protocol between two healthcare facilities in our demonstration system is summarized in Figure 3.

1. Facility A sends a challenge to facility B with an identifying nonce (nonceA).
2. Facility B obtains its TPM credentials by calling a specific TPM function, which also generates an Attestation Identity Key (AIK). The TPM credentials, Facility Credential and the public part of the AIK are signed by the private part of the TPM Endorsement Key and then encrypted using the public part of the Privacy CA key.
3. The encrypted blob is then sent to the Privacy CA as an identity credential request.
4. The Privacy CA decrypts this blob with its private key. It verifies the signatures of the credential request and then creates and sends an identity credential back to Facility B. This credential is a digital certificate containing the public part of the AIK together with nonceA signed by the CA private key.
5. Facility B then performs an integrity measurement using the TPM “Quote” function using the AIK, nonceA and one or more PCR values as input parameters. The AIK can only be used to sign the quoted PCR registers.
6. The message containing the output of Quote function, the identity credential received from the CA and the list of recorded loaded and unloaded processes is then sent to Facility A from Facility B. On reception of this message, Facility A checks the identity credential, verifying that the Privacy CA has signed it. This also reveals that B has a TPM and that the AIK is an identity key in the TPM of B. With this information, A can then verify the signature of the PCR values and nonceA. If the signature is successfully verified, then A knows the PCR values of B. Facility A then does the following three verifications:
    a. All processes in the received list are in the database of acceptable processes with corresponding hash code.
    b. The expected PCR value is computed from the list of processes and compared with the PCR values of B.
    c. All processes that are marked as MUST in an acceptable process database are loaded.
7. If all checks are successful, Facility A sends a challengeSuccess message to facility B.
8. Facility B receives the challengeSuccess message, and initiates the same procedure (steps 8-14 in Figure 2). At the end of these steps, B knows the PCR values of A and both A and B are authenticated.

Fig. 3. Description of the message-based mutual attestation protocol

3.2 Performance Analysis

We implemented the message-based mutual attestation protocol into our distributed healthcare demonstration system, and observed that the server to server medical record transfer was slow, specifically when one needs to transfer a large number of

records at a time (for example, due to a patient transferring from one regional hospital to another). This observation led to an examination of the message-based mutual attestation protocol. We have conducted performance tests on our test bed machines (Dell Optiplex GX620 PCs, 3.39GHz Intel Pentium IV dual processors, 1GByte RAM, Windows XP Professional SP2 with the .NET framework v2.0).

We found that a single run of the mutual attestation protocol between two servers took an average 111 seconds (from the issuing of a request for a medical record to the return of the result). A run consisted of initialization of the system, connection establishment between two servers, connection to a SQL database, and the TPM function calls required for the mutual attestation protocol. We found that, on average, 39% time was spent on making TPM function calls via the TCG software stack (TSS), whereas 61% time was spent on non-TPM related function calls such as connection to remote hosts or SQL server, as shown in Figure 4. The cost of non-TPM related activities depend on specific applications, as well as connectivity of the sender, receiver and privacy CA. This cost can be optimized using different attestation architectures such as WS-Attestation [15], which is beyond the scope of this study. Our focus here is on minimizing the cost of TPM calls in a remote attestation, as we observed that TPM calls consume a significant portion of the time require for a mutual attestation.

We further analyzed the time spent on TPM calls and identified four critical function calls in the mutual attestation protocol. This is done to ensure that there is no hidden overhead to any particular function calls to TPM chip other than the expense incurred to making TPM calls in general, as well as possibility of optimizing TPM calls. These are: CollateIdentity (step 2 in Figure 3), IdentityCredential (step 3 in Figure 3), Quote (step 4 in Figure 3), and VerifyIdentityCredential (step 6 in Figure 3).

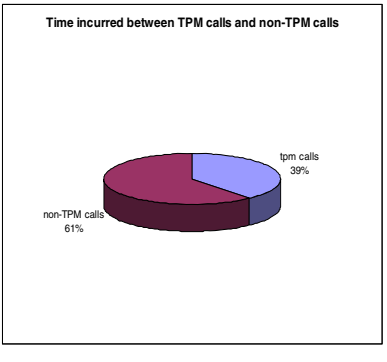


Fig. 4. Time incurred between TPM calls and non-TPM calls

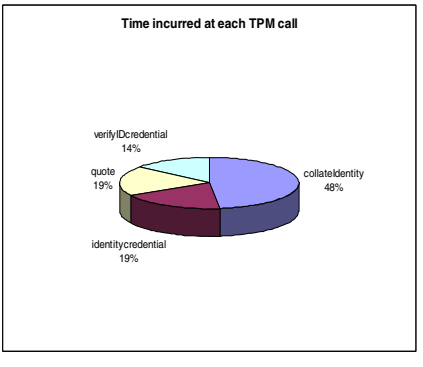


Fig. 5. Time incurred at each TPM call

Figure 5 shows the time spent on these four different function calls. Out of time spent on TPM calls, CollateIdentity took 48% time (20 seconds), IdentityCredential took 19% (8 seconds), Quote took 19% (8 seconds) and VerifyIdentityCredential took 14% (6 seconds). We conducted further analysis of implementation code with the aim of minimizing the number of TPM calls and the complexity of code at each stage. This led us

to the conclusion that TPM calls are stable as each TPM call at different stages took about the same time without having particular overhead on any specific TPM calls because of different implementation code. For example, both "IdentityCredential" and "Quote" made about 12 TPM calls each taking 8 seconds, whereas "VerifyIdentity Credential" made 9 TPM calls with 6 seconds response time. "CollateIdentity" made 20 TPM calls, as well as complex calculation of bytes, to create and use credentials taking more time than calls used in other stages. These observations led us to believe that there is no overhead placed in any particular TPM calls in our implementation. However, if these calls are repeated for every message exchanged between two servers as done in message-based protocol, the TPM calls become a bottleneck for efficient transfer of data between the servers. From these experimental results, we observed that reducing a large number protocol messages that subsequently triggers a large number of TPM calls might result in an efficient attestation protocol. This has led to the development of a session-based attestation protocol.

It must be noted that it is not our intention to do an exhaustive study on performance of a message-based mutual attestation protocol for a particular TPM implementation on a specific TPM chip. We understand the fact that various vendors and commercial IT companies are still working on improving the functionalities and corresponding technologies related to TPM chips. The performance figures in this section are indication purpose only to demonstrate that TPM calls are relatively expensive at currently provided chips and it could result a serious overhead if these calls are made extensively on today's applications such as message-based mutual attestation protocol [5].

3.3 Security Analysis

Attestation in the TCG specification [1] is used to determine whether there have been any unexpected changes to a computer's hardware and software environments. The hardware check utilise the cryptographic features of the TPM microcontroller, including the use of the Attestation Identity Key (AIK). The software environment and configuration is checked for validity, starting from boot time to application load time, with a set of identifying PCR values. During attestation, the PCR values and the AIK are used to validate that a remote machine (or indeed, the local machine, if required by some applications) may trust the platform.

This TCG-type of attestation has been criticised because it performs the integrity validation only at load-time. Successful load-time attestation does not ensure that attestation is always maintained, with possible compromises not being detected post load-time attestation. This feature, referred to as time-of-attestation to time-of-use gap, has been reported in the literature [3]. Figure 6 shows this gap as (1).

From our implementation of the TCG-style attestation protocol, we further identified another gap, representing the time between the PCR values and AIK are being measured by the sending host, which we call the *time-of-measurement,* and the time at which these values are being validated by the receiving host, which we call the *time-of-verification*. This interval is illustrated in Fig. 6 as (2). During this interval, it is possible that by the time PCR values and the AIK are being validated by the receiving host, the sending host's TPM and platform characteristics may have been compromised.

In service-oriented co-operative applications where messages are exchanged asynchronously, it is not difficult to imagine the gap between the time-of-measurement and time-of-verification becoming an easy target for attackers to compromise the attested platform. Our performance analysis results indicated that the interval in between the time-of-measurement and time-of-verification is not small or insignificant, and hence offers an increase in the probability of a successful attack against the protocol over a system that has a smaller time-of-measurement to time-of-verification gap.

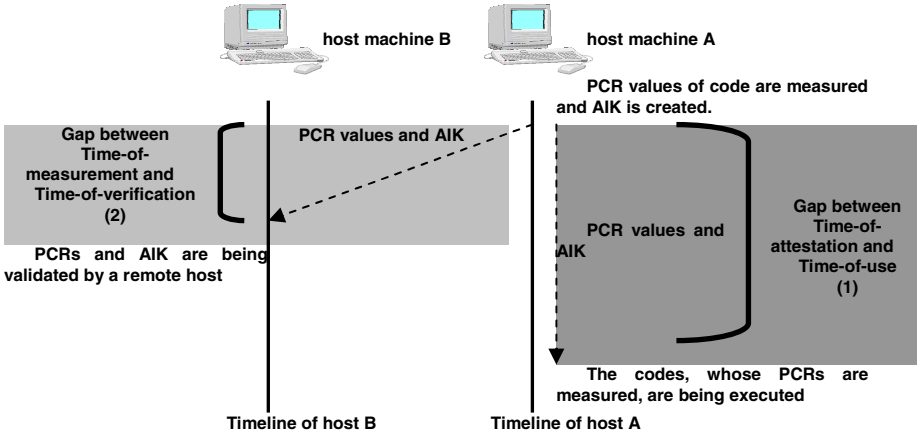


Fig. 6. The gaps between the time-of-attestation and time-of-use, and the time-of-measurement and time-of-verification

4 Session-Based Mutual Attestation Protocol

TPM provides a "Seal" function that encrypts a key, and ties specific platform measurements such that the key can only be decrypted when those platform measurements have the same values that they had when the key was created. The TPM may be used o seal and unseal other data that is generated outside of the TPM. Our session-based protocol uses these sealing and unsealing functions to preserve the environment for a session.

Our session-based mutual attestation protocol is based on the Web Services standard WS-BusinessActivity, and relies on sealing an application's session context using the platform environment's measurements. This sealed object is used as a shared secret between the two interacting parties, and can only be decrypted to the application session context if there has been no change in the platform environment. Further, should any change be detected, our session-based protocol terminates and all related session information is erased.

The protocol itself consists of two phases: the first for the initialization and establishment of the trust relationships, and the second for the maintenance and termination of the protocol.

4.1 First Phase: Establishing a Trusted Session

Our protocol establishes a trusted session by applying some modifications to existing message-based attestation protocol described in [5]. During the first phase of our protocol (Figure 7), the sender facility sends an attestation challenge that contains a random nonce, which acts as a unique communication token from the sender facility to the receiver facility (as before, in the message-based mutual attestation protocol). The receiver facility then signs one or more PCR values representing its platform state using an attestation identity key (AIK). At the same time, it also seals the token (nonce) with its own PCR values using the AIK. The signed PCR values and public part of AIK are sent to the sender facility as a response. The receiver also sends along an attestation challenge that contains another random nonce to the sender, which acts as a unique communication token from the receiver facility to the sender facility. These two unique nonce values form an application session context. The sender then verifies the received signed values with the expected values to determine whether the challenge is succeed or fail.

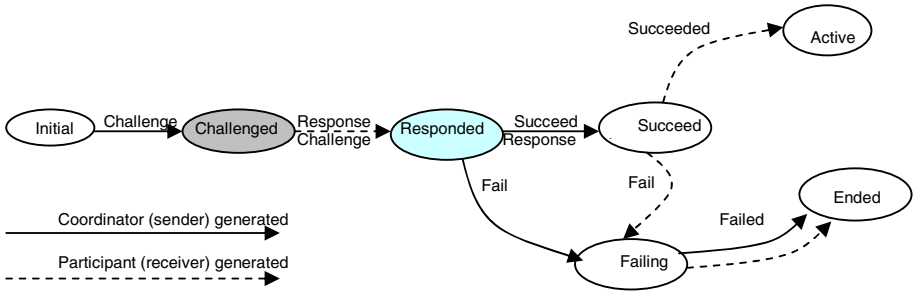


Fig. 7. State diagram of the first phase of the session-based mutual attestation. Coloured states show the difference between the message-based mutual attestation and the session-based mutual attestation protocols, where a sealing of application session context occurs.

If the challenge is successful, the sender facility then signs one or more PCR values representing its platform state using an attestation identity key (AIK). It also seals nonce received from the receiver with its own PCR values using the AIK. After sealing, the sender sends the PCR values and the public part of AIK to the receiver facility, as a response along with the successful result for its response to the earlier challenge. The receiver facility verifies the received signed values with the expected values to determine whether the challenge is successful or fail. The successful result of the first phase of the session-based mutual attestation ends at the active state, where a trusted session is created and two interacting healthcare providers can exchange electronic patient records, whereas unsuccessful attestation ends at the ended state.

In summary, the detailed description of the session-based protocol is similar to that of message-based protocol shown in Figure 3 except that it has a step in between steps 5 and 6 for sealing a context using existing platform configuration. The application context could be a shared secret or any shared data (nonce).

4.2 Second Phase: Session Maintenance and Termination

The second phase of the protocol maintains the trusted session and guarantees the session termination in an agreed outcome. As it uses the Web Services standard WS-BusinessActivity, we briefly introduce the WS-BusinessActivity specification and then describe the use of it in the session-based protocol.

WS-BusinessActivity defines a set of message types, and an ordering relationship between the messages, that may be used by application programmers as a building block for termination in their application protocols. Message types and their priority-based ordering relationships define a possible set of allowable message exchanges when one party initiates the termination. Ordering relationships are defined in such a way that if two messages pass each other in transit when the protocol is in a certain state, one message gets priority over the other. This property guarantees that all possible race conditions at termination are handled, which is one of the problems for termination between two parties using asynchronous communication.

WS-BusinessActivity defines two coordination protocols: Participant Completion and Coordinator Completion. In these protocols, one party is referred to as the coordinator while the other party is referred to as the participant. Coordinator completion is the appropriate one if the coordinator initiates termination; otherwise, participant completion protocol type should be used. The abstract state diagram for the business agreement with coordination completion protocol is shown in Figure 8. We omit the discussion on the participation completion protocol here due to the limitation of space, but the discussion in this paper applies to it as well.

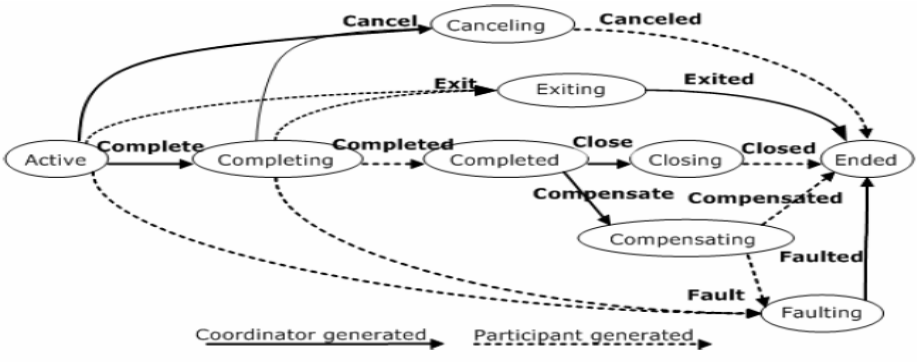


Fig. 8. WS-BA BusinessAgreementWithCoordinatorCompletion Coordination Type

Cooperating facilities involved in sharing medical records can play two different roles: coordinator and participant. One facility must act as a coordinator and the other as a participant. The coordinator creates the context for the interaction and passes it to the participant, along with an application message, as an invitation to participate in an activity. The facility that initiates the interaction by sending *challenge* message first becomes a natural candidate for the coordinator. In our protocol, the facility A becomes a coordinator and the facility B becomes a participant.

In our protocol, we always use coordinator completion protocol shown in Figure 8. However, one can easily use participation completion protocol as well. The coordinator completion defines nine possible paths. It is not necessary to use all paths and one can use a subset of these paths. We now give the semantics of those paths for session-based protocol and explain the second phase.

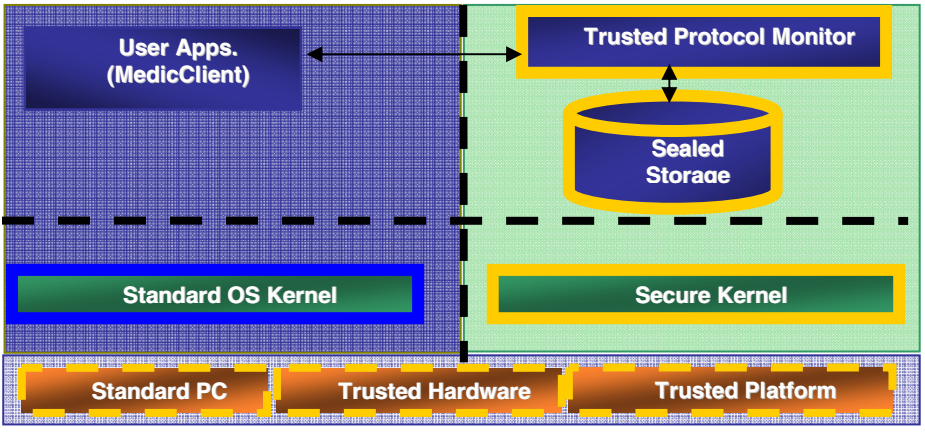


Fig. 9. Implementation Architecture

The second-phase starts with the Active state obtained at the end of the first-phase. The cooperating facilities exchange application messages (such as medical records) in this state. The application messages are encrypted and decrypted using AIKs. The sending party encrypts the messages using AIK public key of the receiving party obtained in the first phase of the protocol. When the receiving application receives the encrypted messages, the application requests trusted protocol monitoring code (see Figure 9) for decryption. Note that the application itself can not decrypt the messages as the AIK private key is securely stored in TPM and only accessible by trusted code. The trusted protocol monitor first checks the environment by running unsealing operation with input parameters such as the nonce created by the sending party in the first phase, PCR values, and the AIK. If the unsealing operation is successful, the application message is decrypted and passed to the application running in the user space. Otherwise, the protocol exchanges fault/faulted messages and terminates. Note that though the WS-BusinessActivity coordination completion supports fault message for participant, we modify it to support fault message for both coordinator and participant, which is not shown in Figure 8. If all messages are successfully decrypted, the protocol terminates using the successful path of WS-BA protocol such as complete/completed followed by close/closed. It implies that the platform configuration and software environments have not changed during the session.

It is natural for facilities to start termination in the middle of a session due to many reasons such as there is a damaging report in the newspapers about how the interacting facility handles client reports. The sending facility uses exiting path whereas the receiving facility uses the canceling path to terminate the session in such situations.

4.3 Security Analysis

It is important to state the fundamental assumptions made by the session-based protocol before performing its security analysis. We assume that the code that runs the protocol is implemented as a trusted code as shown in Fig. 9. That is, the code is always run in an isolated memory space and can never be interfered by other concurrently running programs. The application data encrypted by AIK public key cannot be decrypted at the user space as the AIK private key is only known to the trusted component of the architecture. This is a reasonable assumption in trusted computing as such protocol can be run as a trusted code in "nexus" of NGSCB. This assumption guarantees that the sealing and unsealing functions can be used to establish the trusted session.

As explained in earlier section, the message-based mutual attestation protocol can suffer from a security attack. It is possible that a malicious attacker can hack the sending party's host system and alter the code that was already attested as depicted in Figure 10. The receiving party cannot notice this malicious activity being done at the sending party's host because the receiving party is still validating the sending party's platform configuration based on the PCR values and AIK being received before the malicious attack at the sender's host. The receiving party will validate the sending party's host machine and sends a success message along with sensitive data only meant to be shared with the legitimate sender host. The malicious attacker at the sender's host pretends nothing has happened and receives all sensitive data the receiving party is sending.

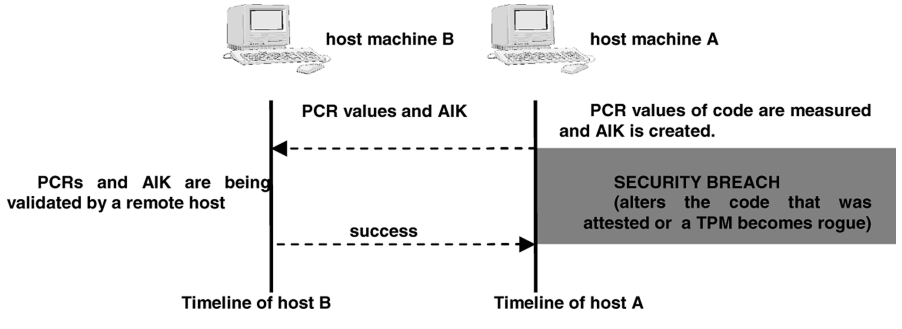


Fig. 10. Possible security attack in TCG remote attestation

This situation is prevented in session-based mutual attestation protocol. In a session-based mutual attestation, a random nonce (or shared secret) used for the communication between the sending party and the receiving party is sealed with PCR values using AIK as soon as PCR values are measured and reported to the receiver. Whenever a message is arrived from a receiving party, unsealing operation is performed to test if the platform environment at the time of sealed operation is still preserved. For example, if there has been any malicious attack that has changed the PCR values since attested platform is measured, unsealing operation performed before each arriving message being decrypted will fail. When the unsealing operation fails, a fault message is sent off to the receiving party to indicate that the sender host's

platform environment has been altered. Moreover, our assumption of trusted code prevents already arrived message being decrypted by rouge host as AIK private key is secured by trusted component in the architecture. It is also important to note that the context tokens (nonce) are unique and they are attached with each message sent. Unmatched security tokens also generate fault message like "unsealing" operations.

4.4 Performance Analysis

This section compares the performance of our session-based protocol with the message-based protocol in the context of its use in a co-operative healthcare information system. An example of the patient record used for the performance analysis is shown in Figure 11.

Patient name	: char(30)
Patient DOB	: date
Date checked	: date
Doctor	: char(30)
Description	: char (200)

Fig. 11. A structure of the sample patient record

First, we used the mutual attestation protocol to send or receive a single encrypted patient health record between facility A and facility B. It took an average 50 seconds for total TPM calls made during the process. This includes 43 seconds spent on TPM calls made for the mutual attestation protocol, 3 seconds on TPM calls for the message encryption and 4 seconds on TPM calls for the message decryption.

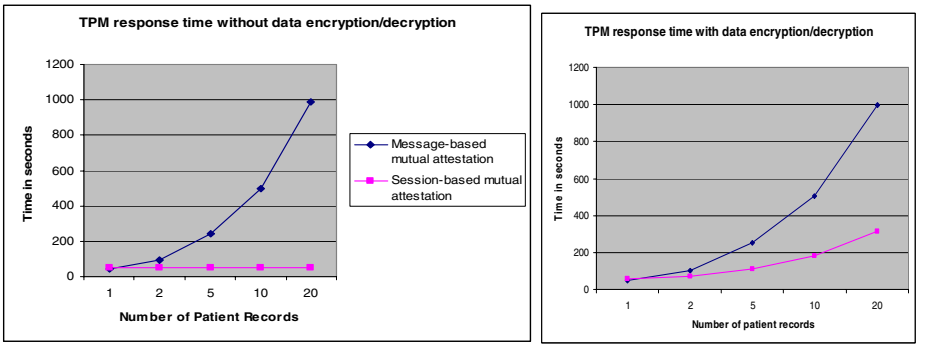


Fig. 12. Time incurred for TPM calls between message-based mutual attestation and session-based mutual attestation

When we applied the session-based protocol for sending a single encrypted patient health record, it took 129 seconds spending approximately 59 seconds on TPM calls. There was an overhead of 9 seconds due to additional TPM calls for sealing/unsealing operations. This small overhead was greatly compensated as the number of patients' health records transferred between facilities is increased, as shown in Figure 12. The X-axis represents the number of patients' records being sent and Y-axis represents the

time spent on TPM calls in seconds. We repeated the same test for other five different set of patients' health records: 2, 5, 10, and 20. The purpose of these tests was to study the impact of the number of records on the performance, when different protocols were used for establishing mutual trust between cooperating facilities. It can be seen that the session-based mutual attestation starts outperforming the message-based mutual attestation after exchanging two patients' records. For five patient records, the session-based mutual attestation only takes about half the time than that for the message-based mutual attestation protocol.

The left hand figure in Figure 12 shows the response time of TPM calls due to protocol messages without data encryption and decryption. The right hand figure Figure 12 in shows the TPM response time with data encryption and decryption. As can be seen from the figures, the TPM response time increases linearly with the number of messages due to the encryption and decryption costs. It is important to note that due to the expensive TPM calls in the current implementation, the performance figure of the session-based mutual attestation is much higher than one expects from such protocol.

5 Related Work

This section presents reviews on different methods of establishing a trust between two interacting parties including remote attestation. The most commonly used trust models in eBay like e-commerce application is reputation-based trust models [17]. In these models, each host is assigned a value of trustworthiness by other hosts based on their past experiences. Such trust models are vulnerable to attacks where attackers can create a large number of false hosts and increase/decrease the trustworthiness of the host. Another set of trust models are based on public-key infrastructure. In these models, when a host is challenged by a remote host, it can verify the identity and trustworthiness of the remote host by verifying the signature on remote host's public key [23]. Despite the efforts of public key infrastructure trust models, the host still lacks the way of verifying the hardware and software environment of the remote hosts.

One way of establishing trust is to use software stack to measure and verify the integrity of co-operating systems. Kennell and Jamieson [8] proposed a technique that computes the checksum of the memory space that are used by the software subject to measure. The host then sends the outcome of the checksum to the third party authority that can verify the checksum result. Monrose et al. [10] illustrated a similar system that verifies the correct execution of code between remote Java Virtual Machines to detect the possibility of rough JVM. SoftWare-based ATTestation (SWATT) [9] proposed a technique using a challenge-response protocol. A challenging party sends a challenge to the embedded device. The embedded device computes its memory content as a checksum and returns it to the challenging party. The challenging party can locally compute the correct answer to its challenge, and can thus verify the answer returned by the embedded device. One of the most critical disadvantages of using software-based solution is that the hardware that hosts the software stack can be stolen or the hardware can be hacked to monitor the behavior of the software stack.

In recent times, hardware-based techniques have gained popularity due to the fact that hardware is relatively harder to hack than software. The most relevant to our work is TCG remote attestation [1] that allows distributed remote host to verify each

other by sending the snapshot of current state of platform configuration. Several problems have been identified related to remote attestation due to the difficulty to measure a platform configuration accurately in today's complex system and inapplicability of having a trusted central authority in this untrusted world, where totally independent systems communicate with each other in an autonomous manner [11, 12]. These problems are driving further research activities in remote attestation.

BIND [3] used a technique that attests only the critical code immediately before it executes and used a sand-boxing mechanism similar to Secure Kernel found in AMD's Secure Execution Mode (SEM) [13] to protect the execution of the attested code. It can detect the changes of attested code at runtime caused by buffer overflows or string malformation. Alternative to such approaches is proposed by Terra [14], which is a flexible architecture for trusted computing. The key primitive that Terra builds on is a trusted virtual machine monitor (TVMM) that allows Terra to partition the platform into multiple isolated VMs. The TVMM provides a narrow interface for attestation. It first generates a certificate that forms the basis of attestation that contains the hash of a VM, TVMN's public key and any other application data used for authenticating the VM. The remote party retrieves this certificate to check the validity of a VM.

The most relevant work to our session-based attestation is the proposal of WS-Attestation by Yoshiham et al. in [15], where they presented an attestation architecture that can be used in Web Services environment. In WS-Attestation, a secure communication channel is established between two interacting Web Services before the attestation. The challenging Web Service application then attests its platform environment using three security tokens; Measurement token that contains binary PCR values, Platform Measurement Description (PMD) token that contains software environment such as lists of components and their SHA-1 hash values, and Attestation Credential token that contains a certificate of valid AIK. These three security tokens are attached to each message (along with a HMAC authentication code) and send to another Web Service using the standard key exchange protocol WS-Trust [16] over the secure channel. The receiving Web Service uses Attestation Credential to validate the authenticity of the sending Web Service whilst uses Measurement and PMD tokens to check the platform and software configuration and its freshness of the sending Web Service. First, this protocol is an extension of remote-attestation for Web Services world and does not support mutual attestation. Second, though this protocol addresses the problem of the gap to certain extent, it has the same drawback as message-oriented protocol in terms of performance. Rather than attaching a large set of validation messages along with each application message as in message-oriented attestation, our approach exploits the TPM functionalities and uses "sealing" and "unsealing" operations to preserve and validate the state of the platform for each message.

6 Conclusions

In this paper, we first analysed the message-based mutual attestation protocol and measured its performance in the context of cooperative healthcare information systems. Our observations and analysis have led us to the conclusion that the

implemented protocol had two major shortcomings: (a) poor performance and (b) a security flaw due to the time interval between the time-of-attestation and time-of-use. To compensate, we proposed a session-based mutual attestation protocol.

One of the advantages of the session-based protocol is that cooperating parties can exchange any number of messages during an application session, without re-attesting already attested platform. This reduces the number of protocol related message exchanges and TPM calls, and improves the performance significantly, particularly when a large number of messages are exchanged. This is achieved by sealing the session context (or shared secret) using attested platform environment, and unsealing the context with the current environment along with the execution of the attested code in an atomic operation. If the environment is changed, the atomic operation fails before executing the attested code, thereby bridging the gap between the time-of-attestation and time-of-use.

We have implemented the session-based mutual attestation protocol in a cooperative distributed healthcare system, and conducted a performance analysis comparing it to the message-based mutual attestation protocol. As expected, by separating out control messages from the information exchanged during a session, performance did improve, and was directly proportional to the number of medical records and other related messages exchanged in a session.

One of the drawbacks of the session-based attestation is that it enforces participating parties not to run any software (including maintenance activities during the session) as such activities could change the attested platform configuration. This type of protocol would not be suitable for deployment in general, desktop PCs or workstations, as the environments are highly variable. However, this protocol is more suited to systems such as server applications where the platform configurations remain stable for a longer duration.

Between two aspects of remote attestation, the session-based protocol detects and prevents the changes in software environment, but the detection and prevention of rouge TPM during a session has not been addressed. That is, the trusted platform that issued AIK has been registered as rogue while interacting participants in a session are using it. We plan to address this problem in our future work. In future work, we plan to do detailed security analysis of the session-based protocol and study the relationships of this protocol with other Web Services standards such as WS-Security, WS-Policy and WS-Trust in the context of service oriented co-operative information systems.

Acknowledgements

This work is completed as part of CeNTIE project that is supported by the Australian Government through the Advanced Networks Program of the Department of Communications, Information Technology and the Arts.

References

1. TCG specification v1.1 https://www.trustedcomputinggroup.org/specs/TPM/
2. WS-BusinessActivity ftp://www6.software.ibm.com/software/developer/library/WS-BusinessActivity.pdf

3. E. Shi, A. Perrig, and Van D. L. BIND: a fine-grained attestation service for secure distributed systems. IEEE Symposium on Security and Privacy, pp. 154- 168, 2005
4. S. Nepal, J. Zic, F. Jaccard and G. Krachenbuehl. A Tag-based Data model for privacy-preserving medical applications. In Proceedings of EDBT IIHA Workshop, Munich, Germany, 2006, pp. 77-88.
5. S. Nepal, J. Zic, G. Krachenbuehl and F. Jaccard. Secure Sharing of Electronic Patient Records, 1st European Conference on eHealth, 2006, Fribourg, Switzerland October 12 – 13, 2006 (to appear).
6. Australian Government Department of Health and Aging Project. Consumer consent in electronic health data exchange – e-consent.
7. O'Keefe, C.M., Greenfield, P., and Goodchild, A. A Decentralised Approach to Electronic Consent and Health Information Access Control. Journal of Research and Practice in Information Technology, Vol. 37(2):161-178, May 2005.
8. R. Kennell and L.H. Jamieson. Establishing the genuinity of remote computer systems. In Proceedings of the 11th USENIX Security Symposium. USENIX, August 2003.
9. A. Seshadri, A. Perrig, L. van Doorn, and P. Khosla. SWAtt: SoftWare-based Attestation for embedded devices. In Proceedings of IEEE Symposium on Security and Privacy, May 2004.
10. F. Monrose, P. Wyckoff, and A. D. Rubin. Distributed execution with remote audit. In ISOC Network and Distributed System Security Symposium, pages 103–113, 1999.
11. V. Haldar and M. Franz. Symmetric Behavior-Based Trust: A New Paradigm for Internet Computing. New Security Paradigms Workshop, Sept 2004
12. J. Reid, M. Juan, G. Nieto, E. Dawson and E. Okamoto. Privacy and Trusted Computing, 14th International Workshop on Database and Expert Systems Applications (DEXA'03) pp. 383, 2003.
13. AMD platform for trustworthy computing. WinHEC 2003, http://www.microsoft.com/whdc/winhec/papers03.mspx, Sept. 2003.
14. T. Garfinkel, B. Pfaff, J. Chow, M. Rosenblum, and D. Boneh. Terra: A virtual machine-based platform for trusted computing. In Proceedings of Symposium on Operating System Principles (SOSP), Oct. 2003.
15. S. Yoshihama, T. Ebringer, M. Nakamura, and S. Munetoh.. WS-Attestation: Efficient and Fine-Grained Remote Attestation on Web Services, International Conference on Web Services. pp 743-750. July 2005.
16. Web Services Trust Language (WS-Trust) http://specs.xmlsoap.org/ws/2005/02/trust/WS-Trust.pdf
17. L. Xiong and L. Liu. A reputation-based trust model for peer-to-peer ecommerce communities, Proceedings of 4th ACM Conference on Electronic Commerce, 2003, pp. 228-229.
18. Jonathan K. Millen, Rebecca N. Wright: Reasoning about Trust and Insurance in a Public Key Infrastructure. 13th IEEE Computer Security Foundations Workshop (CSFW), 2000: 16-22.

A Unifying Framework for Behavior-Based Trust Models

Christian von der Weth and Klemens Böhm

Institute for Program Structures and Data Organization
Universität Karlsruhe (TH), 76128 Karlsruhe, Germany
{weth, boehm}@ipd.uni-karlsruhe.de

Abstract. Trust models have been touted to facilitate cooperation among unknown entities. Existing behavior-based trust models typically include a fixed evaluation scheme to derive the trustworthiness of an entity from knowledge about its behavior in previous interactions. This paper in turn proposes a framework for behavior-based trust models for open environments with the following distinctive characteristic. Based on a relational representation of behavior-specific knowledge, we propose a trust-policy algebra allowing for the specification of a wide range of trust-evaluation schemes. A key observation is that the evaluation of the standing of an entity in the network of peers requires centrality indices, and we propose a first-class operator of our algebra for computation of centrality measures. This paper concludes with some preliminary performance experiments that confirm the viability of our approach.

1 Introduction

With the advent of a broad range of new services on the web such as auction sites (eBay, Yahoo! Auctions, Amazon), expert sites (ExpertCentral.com, AskMe), distributed computing (Seti@Home, Folding@Home) and file-sharing (Kazaa, Gnutella, Freenet), virtual social networks have proliferated as well. In such environments, an entity that is part of the network may enter interactions with another entity for many reasons [19]. An interaction always bears the risk that the service is not performed as expected or desired. The other party can impair the outcome of an interaction by behaving uncooperatively or maliciously or simply because of bugs. Mechanisms to minimize the exposure to risky interactions are indispensable.

Promising solution in this respect are trust models. Trust is a fundamental concept of human behavior that lets entities distinguish between good and bad partners. A lot of work has been done to formalize the notion of trust (see [17] for an overview). Because of the many facets of human notion of trust, there are many different approaches to this formalization as well. So-called *behavior-based trust models* are a common way to derive the trustworthiness of an entity in open environments. The behavior-based trust models we are aware of follow the same principle: They define a representation of the knowledge about the previous

R. Meersman, Z. Tari et al. (Eds.): OTM 2006, LNCS 4275, pp. 444–461, 2006.

behavior of an entity and propose an evaluation scheme how to derive the trustworthiness of the entity from this knowledge. The number of such trust models is large, e.g., [1,2,24,13,20,10,22]. However, a fixed evaluation scheme contradicts the subjective nature of trust. Humans in social networks have different policies to derive the degree of trust in another party.

In social networks in the physical world, humans do not bother to make their trust policies explicit, i.e., how to derive the degree of trust in another party. This article in turn considers networks where the entities are not humans, but software artefacts controlled by humans. We refer to networks where user-controlled entities interact with each other as *virtual social networks*. One important aspect of the control over entities is that the control instance must be able to specify the trust policy of the entities explicitly. To facilitate this, there exist so-called *trust policy languages*, serving as an interface between a human and his entities in the virtual social network. Existing policy languages such as [21] define rules or clauses to derive the trustworthiness of an entity. However, various behavior-based trust polices require complex operations such as aggregation or centrality computation. To the best of our knowledge, existing policy languages do not support this. Further, if logic-based approaches did, the resulting policies could be very complex (we argue). On the other hand, existing behavior-based trust models tend to propose only a fixed evaluation scheme to derive the trustworthiness of an entity. One can only decide whether the derived trustworthiness of another entity is sufficient or not. Only relatively little work has gone into defining policy languages for behavior-based trust models.

In this paper we propose a unifying framework for behavior-based trust models for the specification of a wide range of trust-evaluation schemes. The framework consists of a formal representation of the knowledge about the behavior of an entity in previous interactions and a mechanism to allow users to make his trust policies explicit. This approach overcomes the limitation of a fixed evaluation scheme and emphasizes the subjective nature of trust. More specifically, we make the following contributions:

Specification of the characteristics of behavior-based trust models. We provide an overview of existing behavior-based trust models, in the following way: We outline all aspects of knowledge about the behavior of an entity that we deem essential to model the human intuition of trust in a realistic way. Such aspects are context-dependency and time-dependent decay of the knowledge, to give examples. By pointing out which trust model considers which aspect, we are in the position to illustrate similarities and differences between the models. We will show that some aspects of trust are not considered by any model from the literature we are aware of. For instance, an entity may wish to express its certainty in a rating of another entity it has generated or its estimation of the complexity of a task carried out by another entity.

Formal definition of underlying concepts. Based on the characteristics of behavior-based trust models we define a representation of the knowledge about an entity within our framework. There are different types of knowledge that describe the behavior of a partner. A partner is an entity the current entity has

interacted or is currently interacting with. In a nutshell, the types of knowledge are *feedback* about an interaction with the partner, *recommendations* about the partner from others, the *reputation* of the partner and the degree of *trust* in that partner. We formalize the representation of these concepts. The representation is the result of the analysis of existing behavior-based trust models and of our own attempt to formalize trust. There are some design decisions which had to be taken, and we will discuss these as well.

Definition of a query algebra for trust. Our representation of the underlying knowledge maps to the relational model in a straightforward way. We propose a mechanism based on relational algebra that lets a user make his trust policies explicit. The basic operators of the relational algebra are not sufficient to formulate sophisticated trust policies, as we will demonstrate. Thus, we extend the algebra with a few new operators. In particular, we define an operator for the computation of recursive centrality measures as a first-class operator of the algebra. Centrality measures are graph-based concepts that define the standing of an entity within the network. The basic operators and the extensions form our query algebra for trust. This algebra lets us formulate all kinds of behavior-based trust policies we have identified from the literature. We illustrate the expressiveness of our query algebra by formulating various example policies.

Performance experiments. The computation of recursive centrality measures is time-consuming and resource-intensive. To quantify the impact of our new operator in this respect, we have carried out several experiments. We have implemented different centrality measures in Oracle 10g. In this paper we present some initial results that illustrate the performance of different centrality measures in populations of various sizes (up to 100.000 entities). The different measures have significantly different performance characteristics.

Summing up, after having reviewed existing trust models from literature, we proposes a unifying framework for behavior-based trust models. While this is interesting in itself, it is also the first step of a more long-term research effort. We intend to compare the various trust policies regarding both effectiveness, i.e., rate of interactions whose outcome is as desired, and efficiency, i.e., performance of the implementation. We will be particulary interested in the effectiveness in situations where members of a population with different trust policies keep interacting.

The remainder of this paper is organized as follows: In Section 2 we motivate our approach to a unifying framework for behavior-based trust models. Section 3 discusses that representation of knowledge about an entity's previous behavior in detail. Section 4 introduces our query algebra. We present some first experimental results in Section 5. Section 6 concludes.

2 Our Approach Towards a Unifying Framework

There has been wide-spread interest in the notion of trust from various scientific disciplines. Consequently, many different definitions have come into being. Among existing trust models for open environments, the following definition

prevails: The trust of an Entity A in Entity B is A's subjective degree of belief that B can and will perform a specific task in a certain situation. This definition includes distrust as negative trust.

There are many ways to derive the degree of trust of an entity in a partner within a social network. Ford [6] gives a good survey. Since some approaches to establish trust rely on the social background, not all of them are suitable in virtual social networks. With *cognition-based trust*, cognitive cues such as the first impression determine the degree of belief in the future behavior of another party. Another example is *affect-based trust*. Here, trust is based on emotional bonds between the entities. According to Ford's classification, we see three common ways to derive trust within virtual social networks: *Deterrence-based trust* stimulates entities to behave well because they fear punishment in case of unsatisfactory performance. Newsgroups are a popular example. If a user is lacking good manners, e.g., insults or bashing, he may be banned from the system. Due to the fear of punishment, a user can trust others that their postings have a useful and proper content. *Institution-based trust* reflects the degree of trust felt due to guarantees or other structures. A popular example are public-key infrastructures to issue, distribute and verify certificates or passwords (e.g., *X.509* or *Kerberos*). They allow verifying authenticity or identity of an entity. *Knowledge-based trust* is the result of aggregating relevant knowledge on another entity in order to predict its behavior. This knowledge can include observations of previous behavior of that individual, subsequently referred to as *behavior-specific knowledge*, and personal information, e.g., its real name, location, or occupation.

In large-scale open environments, only some of these derivations are feasible. Deterrence-based trust requires a central instance, a so-called *Trusted Third Party*, which can issue and conduct punishments. Regarding institutional-based trust, even though decentralized infrastructures exist, e.g., *Web of Trust* [14], verifying authenticity or identity of an entity is not sufficient to predict its behavior. Consequently, trust in open environments should be based on knowledge as much as possible, cf. [12]. This includes the limitation to behavior-specific knowledge about an entity. Personal knowledge about an entity in turn is problematic in open environments, for at least two reasons. First, since anonymity and privacy are inherent characteristics of such environments, personal knowledge is rarely available. Second, personal knowledge about a partner has little significance for predicting the behavior of the partner in future interactions. The large number of existing trust models for such environments reflects the relevance of behavior-specific knowledge. We call this kind of trust models *behavior-based trust models*. In what follows, when referring to knowledge about an entity, we mean behavior-specific knowledge.

With these considerations in mind we now propose a unifying framework for behavior-based trust models. We say how the knowledge about an entity's previous behavior is represented, and how an entity can derive the trustworthiness of a partner by means of a query algebra for trust. With that approach we allow the specification of a wide range of trust-evaluation schemes and are not limited to a fixed one like in existing behavior-based trust models.

3 Representation of Behavior-Specific Knowledge

This section formalizes our representation of the knowledge about an entity. Subsection 3.1 gives an overview of the notions that form the knowledge about the past behavior of an entity. Subsection 3.2 outlines the different characteristics of those concepts, which we refer to as *aspects of behavior-based knowledge*. Finally, Subsection 3.3 defines the representation of those notions.

3.1 Overview of Behavior-Specific Knowledge

In general, one can divide behavior-specific knowledge into two classes: While *first-hand knowledge* is based on one's own experiences, *second-hand knowledge* is based on the experiences of others. Furthermore, we distinguish between the following four types of behavior-specific knowledge:

- **Feedback.** A feedback item is Entity A's rating of an interaction performed by Partner B.
- **Recommendation.** A recommendation is the opinion of an Entity A about the previous behavior of a Partner B. In existing models, a recommendation typically is an aggregation of the direct experiences of A with B.
- **Reputation.** Reputation is the general opinion of the whole population of the virtual social network towards a single entity B. Reputation is a global value. With most existing models, it results from the aggregation of direct experiences or recommendations of all entities about B. The aggregation function is specified globally.
- **Trust**. The trust of an Entity A in a Partner B is A's degree of belief that B will behave as expected in future interactions. It is computed based on the aggregation of direct experiences or recommendations about B or on the reputation of B. The aggregation function – which is the trust policy in this case – is left to the trusting entity A.

If Entity A evaluates an interaction with Partner B or provides a recommendation about B to others, A makes that new information about B available to other entities. We say that *A issues knowledge about B*.

Trust and recommendation are similar notions. Both represent the opinion of an entity about the behavior of another one. In fact, some trust models, e.g. [22], do not distinguish between a recommendation of A regarding B and B's trustworthiness in the eye of A. Other models in contrast, e.g., [13], make this distinction. For instance, a recommendation of A regarding B is derived only from A's first-hand knowledge about B, and the trust of A in B is derived from all kinds of knowledge about B. It is important that our framework will allow for both alternatives: The policies of a user specify how recommendations and trustworthiness are derived.

Based on these concepts, a user formulates trust policies to derive the trustworthiness of a partner. For instance, a trust policy could be "I trust a partner if the average of the values of all feedback items issued about it by the 10 most reputable entities exceed a certain threshold". Existing behavior-based trust

models are limited to one trust policy for all entities. In our framework in turn, we let each user make his trust policies explicit for the entities under his control. A user is free to choose which knowledge about a partner he wants to consider and how to combine that knowledge to derive the trustworthiness of the partner. For instance, a user can specify whether the trustworthiness of a partner depends on first-hand knowledge only or on both first-hand and second-hand knowledge. Furthermore, a user can explicitly limit the considered knowledge about a partner within his policies. In particular, this lets him rule out *spoof feedback*, or whatever he deems/could be spoof feedback. For instance, a user A can reject feedback from entities that rated interactions with A negative, though A showed a good behavior. Regarding the reputation of an entity, one can specify its computation using our framework as well. A global policy accomplishes this.

3.2 Aspects of Behavior-Specific Knowledge

In the following, we outline all aspects of trust that we deem essential to model the human intuition of trust as realistically as possible. Different trust models from the literature consider different aspects, as we will illustrate. Furthermore, there are aspects of trust currently not considered by any model we are aware of.

Reference to entities. A piece of behavior-specific knowledge always refers to a certain entity. For feedback, recommendation and trust we can specify the issuer of each piece of such knowledge in addition. M is the set of all entities that currently participate in the virtual social network, and we refer to an element of M as μ.

Context-dependency. The degree of trust in another entity is context-specific. It depends on the circumstances and conditions which determine or specify the meaning of a piece of knowledge. The *context of the interaction* is the set of all facts or circumstances that surround an interaction. For instance, we tend to trust a car mechanic to check our car, but not to check our health. Therefore, trusting a partner to perform a certain task does not mean that one trusts him for other tasks. Most existing trust models feature context-dependency. This indicates the importance of the aspect. We denote the set of all contexts that are possible in a given scenario as Φ and an instance of this set as ϕ.

In spite of context dependency, humans tend to use the knowledge about another party in a certain context to assess its trustworthiness in other contexts. Consider again the previous example. If the car mechanic has always provided good service in fixing cars, one tends to believe that he will also fix motor bikes well. This is particularly useful if one has only little knowledge about Entity μ in Context ϕ_1, but more knowledge about μ in the related Context ϕ_2. Obviously, the more related ϕ_1 and ϕ_2 are, the more significant is the knowledge transfer between both contexts. There are few trust models, e.g. [20], which feature this kind of knowledge transfer. We for our part model the relatedness of contexts using a hierarchy of the elements of Φ. The query algebra which we are about to define will then allow to specify the knowledge transfer between contexts.

Facets of a context. Even though the context may be the same, knowledge about an entity may correspond to different facets of the context. Facets are

distinct features of a context μ, they represent the different perspectives. Think of our car mechanic. In the context of fixing cars, facets may be the speed of his performance, his behavior, the price, etc. Thus, we can deem a partner trustworthy or not in the same context, but according to different facets. Some existing trust models consider different facets of a context, e.g., [24]. We denote the set of all facets of a context ϕ as Ψ_ϕ, an instance of that set as ψ_ϕ. We use the term *situation* to denote a combination of context and facet a piece of behavior-specific knowledge may refer to.

Similarly to the knowledge transfer between different contexts, we define a hierarchical structure over the set of all facets to model the relatedness between them. This allows to support the knowledge transfer between different facets within our query algebra.

Time-dependency. The trustworthiness of an entity is dynamic – it typically changes with every new piece of information about the entity. Furthermore, more recent information about the behavior of an entity influences our decision to trust a partner or not more than older one. To allow for consideration of this aspect, we make the age of each piece of knowledge explicit with a timestamp, denoted by τ.

Most existing models take the dynamic nature of trust into account: the trustworthiness of an entity at time t_1 can differ from its trustworthiness at time t_2. But only few models consider the decay of behavior-specific knowledge, e.g., [5,20]. In contrast to our framework, the rate of decay is part of the fixed evaluation scheme, and the user cannot modify it.

Certainty. When a user assesses the previous behavior of a partner, he does not necessarily feel confident about his own assessment. To assess the behavior of the partner with 'good' or 'bad', the assessor must have an idea of what good or bad behavior might be to him. Depending on the complexity of the task, this is not always obvious. Consider the following example: A car mechanic has fixed the broken front light of our car. To assess the outcome of the repair, in this case the functioning of the front light, is simple. However, assessing the task according to facet 'costs' is more difficult, since we must have an idea of current market prices. The degree of certainty is a subjective factor. Therefore, our representation of (most kinds of) behavior-specific knowledge will include this aspect as well. This does not extend to the reputation of an entity, since it is a global value. We denote the degree of certainty with σ; its range is the interval [0, 1] (0: absolutely uncertain, 1: absolutely certain). No trust model we are currently aware of takes uncertainty into account.

Estimated Effort. In addition to the actual outcome of an interaction, the estimated costs of the partner that has performed the task is an important factor that describes his behavior. If a partner has shown a good behavior in a task that has required much effort, it is likely that he will show a good behavior in a similar task that requires less effort. The other direction is not true. Performing a complex task satisfactorily tends to be a more useful indicator of trustworthiness than a good performance on a simple task. The estimated effort refers to the complexity of an interaction. Thus, we can only describe feedback with this aspect, since a feedback item is the rating of an interaction. We denote the

estimated effort with ϵ; its range is the interval [0..1] (0: very simple task, 1: very complex task). As far as we know, no existing trust model considers this aspect of behavior-specific knowledge.

Valuation. In our model, the assessment of the previous behavior of a partner is continuous. This allows for a finer granularity and for the ranking of different assessments. We denote the valuation of each piece of behavior-specific knowledge by ν from the interval [-1, 1]. Trust models like [20,10] feature a continuous valuation. Other trust models use a discrete one. For instance, [1] distinguishes four degrees of trust, namely 'very trustworthy', 'trustworthy', 'untrustworthy' and 'very untrustworthy'. Few models use a binary valuation, e.g, [2]. Finally, several trust models combine discrete and continuous valuations depending on the type of behavior-specific knowledge, e.g., [11,26].

A trust policy, as we will define it, does not need to refer to all of those aspects explicitly. It is left to the discretion of the policy designer how to take the aspects into account, if at all.

3.3 Defining the Representation of Notions Behind Behavior-Specific Knowledge

We now are in the position to specify the representation of the different types of knowledge about the behavior of an entity in previous interactions. We propose a tuple-structured representation.

Definition 3.1 (Feedback)**.** A feedback tuple is an 8-tuple

$$feedback = (\mu_{rater}, \mu_{ratee}, \phi, \psi_{\phi}, \tau, \sigma, \epsilon, \nu)$$ ■

The relation that contains all feedback tuples is `Feedback (rater, ratee, context, facet, timestamp, certainty, effort, value)`. It is clear that `Feedback` is a base relation, since it is not derived from other data.

In this article, we do not address physical design issues, i.e., how to store the data, or materialization strategies in case of derived data. We currently pursue a centralized approach, i.e., use a RDBMS storing all feedback available. However, and more realistically, one can also envision more sophisticated distributed architectures, e.g., structured P2P systems [9]. For instance, each peer could store the feedback it has generated and the one other peer have made available to it.

Definition 3.2 (Recommendation)**.** A recommendation tuple is a 7-tuple

$$recommendation = (\mu_{recommender}, \mu_{recommendee}, \phi, \psi_{\phi}, \tau, \sigma, \nu)$$ ■

The relation that contains all recommendation tuples is `Recommendation (recommender, recommendee, context, facet, timestamp, certainty, value)`. Clearly, when an algebra expression generates such tuples, the name of the relation may be omitted. As indicated before, `Recommendation` is a derived relation. We give examples of policies how to derive such knowledge in Section 4.

Definition 3.3 (Reputation)**.** A reputation tuple is a 5-tuple

$$reputation = (\mu, \phi, \psi_\phi, \tau, \nu)$$ ■

The relation that contains all reputation tuples is `Reputation (entity, context, facet, timestamp, value)`. Like recommendations, reputation is derived from other knowledge. `Reputation` is a derived relation as well.

Definition 3.4 (Trust)**.** A trust tuple is a 7-tuple

$$trust = (\mu_{truster}, \mu_{trustee}, \phi, \psi_\phi, \tau, \sigma, \nu)$$ ■

The relation that contains all trust tuples is `Trust (truster, trustee, context, facet, timestamp, certainty, value)`. `Trust` is a derived relation, too.

Next to behavior-specific knowledge, our framework also consists of *scenario-specific knowledge*. This includes the set of entities M, the set of contexts Φ and the set of facets Ψ_ϕ. To formulate trust policies, a user also must have access to scenario-specific knowledge. Thus, we specify the following relations: `Entity(id)` contains the unique identifiers of all entities, and `Situation(context, facet)` contains all possible combinations of contexts and facets.

This paper confines itself to behavior-specific and scenario-specific knowledge, in order to compute trust, but this does not need to be the case in general. Any knowledge about an entity could be taken into account, e.g., its online time, its available resources, its workload, etc. If there is a relational representation of this knowledge, the formulation of respective trust policies – we illustrate in Section 4 - is straightforward.

Previous work, e.g., [7], has already discussed the relational representation of hierarchies including operators for navigation. We borrow from this work to facilitate specification of contexts and facets in trust policies. We do not repeat the discussion here, i.e., do not describe the mapping of relations to hierarchies and the definition of those operators. The rationale is that this would not provide any further insight regarding the representation of the notions of behavior-specific knowledge and the basic design of our framework.

In addition to the differentiation between behavior-specific and scenario-specific knowledge, we can classify behavior-specific knowledge along two further dimensions (we pick up this classification in a later section): (1) *Number of entities referenced*. The relations contain either knowledge about a single entity (*entity-specific*) or about a pair of entities (*pair-specific*). Feedback for instance is pair-specific, since there are two roles, the rater and the ratee. (2) *Source of the knowledge*. Some knowledge represents direct experiences regarding the behavior of a partner (*basic*). This basic knowledge is used to derive the other knowledge, which we refer to as *derived* knowledge. Note that the only kind of basic behavior-based knowledge is feedback. We can now classify all relations containing behavior-specific knowledge as shown in Table 1. According to the previous classification, reputation, recommendation and trust are derived knowledge. Since behavior-specific knowledge is subjective, each user has policies of his own to derive trust or recommendations. Reputation requires the existence of

Table 1. Classification of the types of behavior-specific knowledge

	entity-specific	pair-specific
basic		`Feedback`
derived	`Reputation`	`Recommendation`, `Trust`

(at least one) global policy. Our goal is the provision of mechanism for specifying such policies. We do so by defining a query algebra for trust in the next section.

4 Defining a Query Algebra for Trust

The previous section has introduced our relational representation of behavior-specific knowledge about an entity. In this section we define a mechanism that allows users to make their trust policies explicit, based on the relational algebra. We think that existing trust-policy languages do not support the complex operations necessary. To address this issue, we for our part favor an algebraic approach over a logic-based one. Our algebra includes the operators from the relational algebra. However, our query algebra for trust needs additional operators, to be described next. All operators preserve the closure property of the algebra.

A trust policy is a user specification under which conditions he is willing to trust a partner. For instance, behavior-based trust policies are "I trust a partner if the average feedback about it is positive" or "I trust a partner if it is one of the k entities with the highest reputation". The result of a trust policy is a set of entities deemed trustworthy. In our framework, we will use a relational representation of the notions behind behavior-specific knowledge. A trust policy in our framework is an algebra expression over the relational representation of behavior-specific and scenario-specific knowledge. To ease presentation, a trust policy in the examples that follow generates a relation of trust, recommendation or reputation items whose structure conforms to the one from Subsection 3.3. One still would have to filter these tuples (e.g., is the reputation value above a given threshold?) and project out everything but the entity IDs to arrive at the structure desired. To simplify the examples, we will however omit this last step.

Throughout the rest of this section we use the following notation: Let $\mathcal{A}$ be a set of attribute names A_i and $\mathcal{D}$ a set of domains D_i where each D_i is a set of atomic values. To associate an attribute A_i with a domain D_i we write $D_i : A_i$. A relation schema $\mathcal{R}$ consists of a set of distinct attribute names $\{A_1,...,A_n\}$. A relation is a finite set $R \subseteq D_1 : A_1 \times ... \times D_n : A_n$. We write tuple $t \in R$ as $(A_1 : a_1, ..., A_n : a_n)$ where value a_i is from the domain D_i of attribute A_i. Schema(R) denotes the set of attributes $\{A_1, ..., A_n\}$ of R. To refer to the value of an attribute A_i of a tuple t, we write $t.A_i$.

4.1 Conventional Extensions to the Relational Algebra

Our query algebra for trust includes some operators that have already been proposed in the literature as extensions to the relational algebra.

Grouping and Aggregation. Any realistic trust policy requires aggregation, for instance, the average feedback value about an entity. Aggregation typically comes together with grouping. To ease understanding of our example policies to follow, we repeat the definition of the Group operator.

Definition 4.5 (Group operator)**.**

$$\mathbf{GROUP}[\mathrm{A}, \Gamma(\mathrm{A_i}), \emptyset](\mathrm{R}) := \\ \{(\mathrm{A_1}:\mathrm{a_1}, ..., \mathrm{A_n}:\mathrm{a_n}, \mathrm{A}:\mathrm{a}) | (\mathrm{A_1}:\mathrm{a_1}, ..., \mathrm{A_n}:\mathrm{a_n}) \in \mathrm{R} \wedge \mathrm{a} = \Gamma(\mathrm{A_i})\}$$

$$\mathbf{GROUP}[\mathrm{A}, \Gamma(\mathrm{A_i}), \{\mathrm{A_1}, ..., \mathrm{A_k}\}](\mathrm{R}) := \\ \bigcup_{\mathrm{t} \in \mathrm{R}} \mathbf{GROUP}[\mathrm{A}, \Gamma(\mathrm{A_i}), \emptyset](\mathbf{SELECTION}[\bigwedge_{\mathrm{i}=1}^{\mathrm{k}} (\mathrm{a_i} = \mathrm{t}.\mathrm{A_i})](\mathrm{R}))$$

where $\mathrm{Schema}(\mathrm{R}) = \{\mathrm{A_1}, ..., \mathrm{A_n}\}$, $\mathrm{A_i} \notin \{\mathrm{A_1}, ..., \mathrm{A_k}\}$, $\mathrm{A_i} \in \mathrm{Schema}(\mathrm{R})$, $\Gamma(\mathrm{A_i})$ applies the aggregation function Γ to the values of the attribute $\mathrm{A_i}$ and furthermore $\{\mathrm{A_1}, ..., \mathrm{A_k}\} \subseteq \mathrm{Schema}(\mathrm{R})$. ■

Deployment of External Functions. Many trust policies explicitly require external functions. For instance, to express that recent feedback about a partner has higher impact on his trustworthiness than older one, we require a function that describe the decay of a feedback item according to its age. We accomplish this using the Map operator (cf. [3]). It applies a user-defined expression to the attributes of a relation, one tuple at a time. A new attribute stores the result. Again, the Map operator (or similar operators with different names) is not new; we just repeat it here to ease presentation.

Definition 4.6 (Map operator)**.**

$$\mathbf{MAP}[\mathrm{A}, \mathrm{expression}(\mathrm{a_1}, ..., \mathrm{a_n})](\mathrm{R}) := \\ \{(\mathrm{A_1}:\mathrm{a_1}, ..., \mathrm{A_n}:\mathrm{a_n}, \mathrm{A}:\mathrm{a}) | (\mathrm{A_1}:\mathrm{a_1}, ..., \mathrm{A_n}:\mathrm{a_n}) \in \mathrm{R} \wedge \mathrm{a} = \mathrm{expression}(\mathrm{a_1}, ..., \mathrm{a_n})\}$$

where $\mathrm{Schema}(\mathrm{R}) = \{\mathrm{A_1}, ..., \mathrm{A_n}\}$ and $\mathrm{A} \notin \mathrm{R}$. ■

Top operator. Within trust policies, the k tuples with the highest values of an attribute frequently are of special interest. For instance, a policy might refer to the entities with the highest reputation values. Several definitions of the Top operator exist, e.g., [4]. Again, to ease presentation, we repeat the definition of the Top operator here.

Definition 4.7 (Top operator)**.**

$$\mathbf{TOP}[0, \mathrm{A_i}](\mathrm{R}) := \emptyset, \quad \mathbf{TOP}[\mathrm{k}, \mathrm{A_i}](\emptyset) := \emptyset$$

$$\mathbf{TOP}[\mathrm{k}, \mathrm{A_i}](\mathrm{R}) := \\ (\mathrm{M} = \{\mathrm{t_1}, ..., \mathrm{t_m} \in \mathrm{R} | (\nexists \mathrm{x} \in \mathrm{R}, \mathrm{x} \neq \mathrm{t} : \mathrm{x}.\mathrm{A_i} > \mathrm{t}.\mathrm{A_i}) \wedge (\mathrm{m} \leq \mathrm{k})\}) \\ \cup \mathbf{TOP}[\mathrm{k} - |\mathrm{M}|, \mathrm{A_i}](\mathrm{R} - \mathrm{M})$$

where $\mathrm{A_i} \in \mathrm{Schema}(\mathrm{R})$ and k is a natural number. $\geq$ refers to a given partial order over $\mathrm{A_i}$. ■

To give an example, we consider the following policy: "I (μ_{self}) trust a Partner $\mu_{partner}$ in Context ϕ and Facet ψ_ϕ if the average of all available feedback values about $\mu_{partner}$ is positive. The certainty of the result depends on the average certainty values of all considered feedback. Only feedback from the 10 entities with the highest reputation is considered". Example 1. shows the corresponding algebra expression. The Top operator applied to Relation `Reputation` retrieves the 10 most reputable entities in Context ϕ and Facet ψ_ϕ. The inner Group operator computes the average feedback value. Its results are the values of the new attribute `avg_value`. Furthermore, a trust tuple requires a certainty value σ and a timestamp τ. In this example, the second group operator computes σ as the average of the certainty values of all feedback tuples. Its results are the values of the new attribute `avg_certainty`. The Map operator accomplishes the computation of τ.

Example 1. The 10 most reputable entities in Context ϕ and Facet ψ_ϕ

PROJECTION[truster, trustee, context, facet, time, avg_certainty, avg_value](
SELECTION[truster=μ_{self}](
RENAME[rater→truster, ratee→trustee](
MAP[time, getCurrentTime()](
GROUP[avg_certainty, AVG(certainty), {ratee}](
GROUP[avg_value, AVG(value), {ratee}](
JOIN[rater=entity](
TOP[10, value](
SELECTION[context = $\phi \wedge$ facet = ψ_ϕ](Reputation)),
SELECTION[ratee = $\mu_{\text{partner}} \wedge$ context = $\phi \wedge$ facet = ψ_ϕ](Feedback))
))))))

4.2 Centrality

A centrality index is a graph-based measure to quantify the importance of a vertex among all vertices of a graph according to the structure of the graph. Centrality indices are a heavily studied concept in social network analysis. A lot of measures have been developed, including degree centrality, closeness, betweenness, eigenvector centrality and others. Existing trust models apply centrality indices to evaluate the trustworthiness of entities, e.g., [13,25]. To support these schemes in our framework, a user must be able to specify the computation of centrality indices within his policies. To preserve the closure property of the algebra – allowing the nesting of all operators – we introduce centrality as a first-class operator.

Computing centrality requires that the underlying graph is made explicit in the first place. A language for the specification of trust policies should provide a high degree of flexibility in this respect. A trust policy must be able to specify for which entities centrality is computed, and what exactly the edges are, e.g., feedback, recommendations, etc. In other words, the trust policy specifies the underlying graph. Recall that a trust policy is an algebra expression in the context of this

paper. Hence, policies referring to centrality index values of entities must contain a subexpression, subsequently referred to as $\mathbf{R_{vertices}}$, which specifies the vertices, and another one, subsequently referred to as $\mathbf{R_{edges}}$, specifying the edges. Put differently, an operator for centrality computation has parameters $\mathbf{R_{vertices}}$ and $\mathbf{R_{edges}}$. In addition, the operator will have the following parameters:

- $\mathbf{A_v}$: attribute of $\mathrm{R_{vertices}}$ that specifies the vertices
- $\mathbf{A_s}$: attribute of $\mathrm{R_{edges}}$ that specifies the start vertices of edges
- $\mathbf{A_t}$: attribute of $\mathrm{R_{edges}}$ that specifies the target vertices of edges
- $\mathbf{A_w}$: attribute of $\mathrm{R_{edges}}$ that specifies the weight of edges

If our algebra contained further graph-based operators, these would feature these parameters as well. To ease presentation of further definitons, we introduce following abbrevations to represent previous parameters for a graph G: $\mathcal{R}_\mathcal{G} = (\mathrm{R_{vertices}}, \mathrm{R_{edges}})$ and $\mathcal{A}_\mathcal{G} = (\mathrm{A_v}, \mathrm{A_s}, \mathrm{A_t}, \mathrm{A_w})$.

Our goal is the definition of one Centrality operator for the computation of various centrality measures. Note that we do not want to have several such operators. The situation is similar to the deployment of external functions. There is one Map operator that is part of the algebra, and the function is a parameter of it. However, we cannot use the Map operator to compute centrality indices, since the Map operator applies the function to one tuple at a time, independent of the other tuples. Centrality measures in turn are more complex.

First, we specify the characteristics of a centrality measure, in order to serve as a parameter of the Centrality operator. For illustrative purposes, we then give the definition of one centrality measure. We choose *PageRank* [18], since it is the basis for most existing centrality-based trust models. Finally, we define our Centrality operator as first-class construct of the query algebra.

Definition 4.8 (Centrality Measure)**.** A centrality measure is a parameterized function $\mathbf{CentralityMeasure}[\mathrm{A_v}, \mathrm{A_s}, \mathrm{A_t}, \mathrm{A_w}](\mathrm{R_{vertices}}, \mathrm{R_{edges}}, \mathrm{v_i})$ with the following arguments:

- $\mathbf{R_{vertices}}$, $\mathbf{R_{edges}}$ are database relations
- $\mathbf{v_i}$ is a tuple from Relation $\mathrm{R_{vertices}}$
- $\mathbf{A_v}$ is an attribute of Relation $\mathrm{R_{vertices}}$
- $\mathbf{A_s}, \mathbf{A_t}, \mathbf{A_w}$ are attributes of Relation $\mathrm{R_{edges}}$
- $\mathbf{R_{edges}}$ must not contain two different tuples with identical values of Attributes $\mathbf{A_s}$ and $\mathbf{A_t}$ (we limit ourselves to centrality measures for graphs without multiple edges).

If one of the previous characteristics is not fulfilled, the result of the centrality measure is $\perp$ (undefined). ■

Example: PageRank. The PageRank of a vertex in a directed, weighted graph G(V,E) is defined as follows:

$$\mathrm{PageRank}(\mathrm{v_i}) = (1 - \mathrm{d}) + \mathrm{d} \cdot \sum_{\mathrm{v_j} \in \mathrm{In}(\mathrm{v_i})} \frac{\mathrm{w_{ji}} \cdot \mathrm{PageRank}(\mathrm{v_j})}{\sum_{\mathrm{v_k} \in \mathrm{Out}(\mathrm{v_j})} \mathrm{w_{jk}}} \tag{1}$$

where w_{ji} denotes the weight of the edge from vertex v_j to vertex v_i. $In(v_i)$ denotes the set of vertices that have an edge to v_i and $Out(v_i)$ is the set of vertices that have an edge from v_i. We call $In(v_i)$ the *In-set* and $Out(v_i)$ the *Out-set* of v_i. – The following definitions are auxiliary in nature. We define the concepts of the In-set and the Out-set of a vertex and the weight of an edge according to our relational representation of the graph.

Definition 4.9 (In-set, Out-set, Weight)**.**

$$\mathbf{IN}[\mathcal{A}_\mathcal{G}](\mathcal{R}_\mathcal{G}, v_i) := \{(A_v : t.A_v) |\ (t \in R_{vertices}) \wedge\ (\exists e \in R_{edges} : (e.A_s = t.A_v \wedge e.A_t = v_i.A_v))\}$$

$$\mathbf{OUT}[\mathcal{A}_\mathcal{G}](\mathcal{R}_\mathcal{G}, v_i) := \{(A_v : t.A_v) |\ (t \in R_{vertices}) \wedge\ (\exists e \in R_{edges} : (e.A_s = v_i.A_v \wedge e.A_t = t.A_v))\}$$

$$\mathbf{WEIGHT}[\mathcal{A}_\mathcal{G}](\mathcal{R}_\mathcal{G}, v_s, v_t) := \sum_{e \in \{t \in R_{edges} | t.A_s = v_s.A_v \wedge t.A_t = v_t.A_v\}} t.A_w$$

■

These auxiliary definitions let us now define the PageRank measure. The difference to Equation 1 is that the following definition is a centrality measure, as defined earlier. It is based on the relational representation of graph structures. It can be an argument of the Centrality operator of our algebra, to be defined subsequently.

Definition 4.10 (PageRank measure)**.**

$$\mathbf{PageRank}[\mathcal{A}_\mathcal{G}](\mathcal{R}_\mathcal{G}, v_i) := (1-d) + d \cdot \sum_{v_j \in \mathbf{IN}[\mathcal{A}_\mathcal{G}](\mathcal{R}_\mathcal{G}, v_i)} \left(\frac{\mathbf{WEIGHT}[\mathcal{A}_\mathcal{G}](\mathcal{R}_\mathcal{G}, v_j, v_i) \cdot \mathbf{PageRank}[\mathcal{A}_\mathcal{G}](\mathcal{R}_\mathcal{G}, v_j)}{\sum_{v_k \in \mathbf{OUT}[\mathcal{A}_\mathcal{G}](\mathcal{R}_\mathcal{G}, v_j)} \mathbf{WEIGHT}[\mathcal{A}_\mathcal{G}](\mathcal{R}_\mathcal{G}, v_j, v_k)} \right)$$

where $Schema(R_{vertices}) = \{A_1, ..., A_n\}$. ■

Having introduced the notion of a centrality measure, we are now in the position to define the Centrality operator. Note that only this operator has to preserve the closure property of the algebra, since only this operator is a first-class construct.

Definition 4.11 (Centrality Operator)**.**

$$\mathbf{CENTRALITY}[A, \mathcal{A}_\mathcal{G}, CentralityMeasure](\mathcal{R}_\mathcal{G}) := \{(A_1 : a_1, ..., A_n : a_n, A : a) |\ v = (A_1 : a_1, ..., A_n : a_n) \in R_{vertices} \wedge\ a = CentralityMeasure[\mathcal{A}_\mathcal{G}](\mathcal{R}_\mathcal{G}, v)\}$$

where $Schema(R_{vertices}) = \{A_1, ..., A_n\}$. ■

To illustrate, we modify the trust policy of Example 1. Instead of taking feedback from the entities with the highest reputation into account, we specify explicitly how the entities are ranked by PageRank, using the Centrality operator. To keep the example simple, we assume that each entity has issued at most one other feedback item about another entity.

Example 2. PageRank of all entities in Context ϕ and Facet ψ_ϕ

PROJECTION[truster, trustee, context, facet, time, avg_certainty, avg_value](
SELECTION[truster=μ_{self}](
RENAME[rater→truster, ratee→trustee](
MAP[time, getCurrentTime()](
GROUP[avg_certainty, AVG(certainty), {ratee}](
GROUP[avg_value, AVG(value), {ratee}](
JOIN[rater=id](
TOP[10, pagerank](
CENTRALITY[pagerank,id,rater,ratee,value,**PageRank**](
Entity,
SELECTION[context = $\phi \wedge$ facet = ψ_ϕ](Feedback)))
SELECTION[ratee = $\mu_{\text{partner}} \wedge$ context = $\phi \wedge$ facet = ψ_ϕ](Feedback))
))))))

The examples so far provide an insight into the formulation of trust policies with our framework. We think that our algebra lets us express most realistic behavior-based trust policies, including the ones of existing behavior-based trust models.

5 Preliminary Experimental Results

In the following, we describe preliminary experiments regarding the performance characteristics of our Centrality operator. We have implemented various centrality measures in PL/SQL, including PageRank, Authorities (A) [15], Proximity Prestige (PP) [16], Integration (I) [23] and Positional Weakness Function (PWF) [8]. As eigenvector centrality measures, PageRank and Positional Weakness Function are implemented based on the *power iteration* algorithm. We carried out various experiments to explore their performance in networks consisting of up to 100.000 entities. Our testbed was a dual 2.2 GHz Opteron 64bit platform, 2GB main memory, SCSI Ultra 320 hard drive running Red Hat Enterprise Linux 3 and Oracle Database 10g Release 2.

Figure 1 graphs the performance of our implementations of the measures in networks of up to 2.000 entities. Performance varies significantly from measure to measure. In general, eigenvector centrality measures based on the power iteration implementation show a better performance. This is because the power-iteration implementation causes significantly less write operations on the database than the implementations of the other measures. Having observed this, we ran another

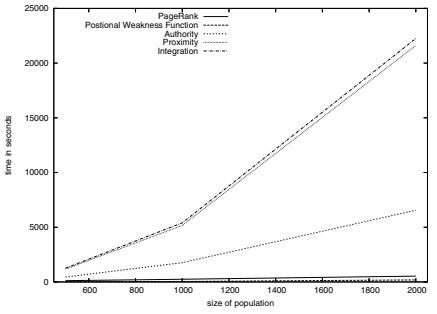


Fig. 1. Performace of all centrality measures

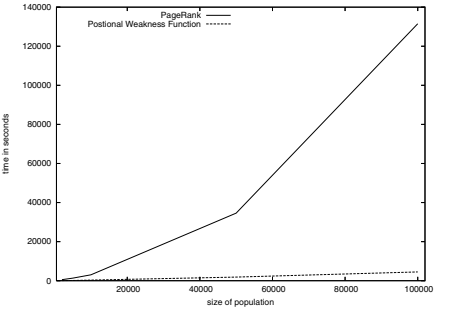


Fig. 2. Performance of PR and PWF in large networks

test with the eigenvector-based measures PageRank and Positional Weakness Function in networks of up to 100.000 entities (see Figure 2). Again, there is a huge difference between the performances of both measures. At this stage, we cannot completely explain this effect. Other experiments (omitted here for lack of space) indicate that performance hinges on the implementation of matrix operations or the error threshold for the power iteration. A systematic investigation which shall lead to a comprehensive cost model is part of our ongoing work.

Besides efficiency, effectiveness of the measures is important as well. In another experiment we compared the rankings resulting from the various measures in a network of 1.000 entities. Table 2 illustrates the differences between the measures in percent (0%: equal ranking, 100%: maximum difference between two rankings). The value reflects the mean distance between the positions of an entity in two rankings. Except for Integration and Proximity Prestige, all measures yield different rankings. As a result, the choice of the centrality measure influences the result of trust policies. We will have to analyze which measure is most appropriate in a given situation.

Table 2. Difference between the rankings of various centrality measures

	PWF	HITS	PP	I
PR	6.2%	8.2%	5.3%	5.3%
PWF	-	5.4%	9.5%	9.5%
A	-	-	9.7%	9.7%
PP	-	-	-	0.0%

6 Conclusions and Future Work

In this paper we presented our approach towards a unifying framework for behavior-based trust models. While existing models propose a fixed evaluation

scheme to derive the trustworthiness of an entity, we have proceeded as follows: First, we have described the different kinds of knowledge about the behavior of an entity in previous interactions and have specified its representation. Given this, we have proposed a query algebra to formulate trust policies, based on the relational algebra. The basic operators of the algebra do not suffice to formulate all policies that are conceivable. In particular, this includes centrality to compute the standing of an entity in the network. We have addressed this issue by defining a first-class operator for centrality computation.

Letting a user formulate trust policies gives way to new questions. For instance, how efficient is the evaluation of the various trust policies, i.e., performance of the implementation? In particular, computation of recursive centrality measures is time-consuming and resource-intensive. How about effectiveness of the different trust polices, i.e., the rate of interactions whose outcome is as desired? We are particularly interested in effectiveness of the different policies in situations where members of a community with different policies interact repeatedly over a period of time. We intend to answer these and related questions as future work.

References

1. A. Abdul-Rahman and S. Hailes. Supporting trust in virtual communities. In *HICSS '00: Proceedings of the 33rd Hawaii International Conference on System Sciences-Volume 6*, Washington, DC, USA, 2000. IEEE Computer Society.
2. K. Aberer and Z. Despotovic. Managing trust in a peer-2-peer information system. In H. Paques, L. Liu, and D. Grossman, editors, *Proceedings of the Tenth International Conference on Information and Knowledge Management (CIKM01)*, pages 310–317. ACM Press, 2001.
3. K. Aberer and G. Fischer. Semantic query optimization for methods in object-oriented database systems. In *ICDE*, pages 70–79, 1995.
4. E. Bertino, D. Montesi, and A. Trombetta. Fuzzy and presentation algebras for web and multimedia data. *ideas*, 00:134, 2000.
5. J. Carter, E. Bitting, and A. Ghorbani. Reputation formalization within information sharing multiagent architectures, 2002.
6. D. Ford. Trust and knowledge management: The seeds of success. Technical Report WP 01-08, Queen's School of Business, Queen's University at Kingston, Canadas, november 2001.
7. T. Grust and J. Teubner. Relational algebra: Mother tongue—xquery: Fluent. In *Proceedings of the first Twente Data Management Workshop on XML Databases and Information Retrieval, Enschede, The Netherlands*, 2004.
8. P. Herings, G. v. d. Laan, and D. Talman. Measuring the power of nodes in digraphs. Technical report, 2001.
9. R. Huebsch, B. Chun, J. M. Hellerstein, B. Thau, P. Maniatis, T. Roscoe, S. Shenker, I. Stoica, and A. R. Yumerefendi. The architecture of pier: an internet-scale quey processor. In *Proceedings of the 2005 CIDR Conference*, 2005.
10. T. D. Huynh, N. R. Jennings, and N. R. Shadbolt. Developing an integrated trust and reputation model for open multi-agent systems. In *AAMAS-04 Workshop on Trust in Agent Societies*, 2004.

11. R. Ismail and A. Josang. The beta reputation system. In *Proceedings of the 15th Bled Conference on Electronic Commerce*, 2002.
12. A. Josang. The right type of trust for distributed systems. In *Proceedings of the 1996 New Security Paradigms Workshop (ACM)*, 1996.
13. S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina. The EigenTrust algorithm for reputation management in P2P networks, 2003.
14. R. Khare and A.Rifkin. Weaving a web of trust. *World Wide Web Journal*, 2:77–112, 1997.
15. J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, 1999.
16. N. Lin. *Foundations of Social Research*. New York: McGraw-Hill, june 1976.
17. S. Marsh. *Formalising Trust as a Computational Concept*. PhD thesis, Department of Mathematics and Computer Science, University of Stirling, UK, 1994.
18. L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford University, 1998.
19. P. Resnick, R. Zeckhauser, E. Friedman, and K. Kuwabara. Reputation systems: Facilitating trust in internet interactions. In *Communications of the ACM*, pages 45–48, december 2000.
20. J. Sabater and C. Sierra. REGRET: reputation in gregarious societies. In J. P. Müller, E. Andre, S. Sen, and C. Frasson, editors, *Proceedings of the Fifth International Conference on Autonomous Agents*, Montreal, Canada, 2001. ACM Press.
21. K. E. Seamons, M. Winslett, T. Yu, B. Smith, E. Child, J. Jacobson, H. Mills, and L. Yu. Requirements for policy languages for trust negotiation. In *POLICY*, pages 68–79, 2002.
22. R. Sherwood, S. Lee, and B. Bhattacharjee. Cooperative peer groups in nice. *Computer Networks*, 50(4):523–544, 2006.
23. T. V. und R.K. Foreman. Integration and radiality: Measuring the extent of an individual's connectedness and reachability in a network. *Social Networks*, 20:89–105, 1998.
24. Y. Wang and J. Vassileva. Trust and reputation model in peer-to-peer networks. In *Peer-to-Peer Computing*, pages 150–, 2003.
25. A. Yamamoto, D. Asahara, T. Itao, S. Tanaka, and T. Suda. Distributed pagerank: A distributed reputation model for open peer-to-peer networks. In *SAINT-W '04 (SAINT '04 Workshops)*, Washington, DC, USA, 2004. IEEE Computer Society.
26. B. Yu and M. P. Singh. A social mechanism of reputation management in electronic communities. In *Cooperative Information Agents*, pages 154–165, 2000.

A WS-Based Infrastructure for Integrating Intrusion Detection Systems in Large-Scale Environments

José Eduardo M.S. Brandão[1,2], Joni da Silva Fraga[1], Paulo Manoel Mafra[1], and Rafael R. Obelheiro[1]

[1] Universidade Federal de Santa Catarina (UFSC), LCMI,
Caixa Postal 476 – CEP 88040-900 – Florianópolis – SC – Brasil,
{jemsb, fraga, mafra, rro}@das.ufsc.br
http://www.das.ufsc.br
[2] Instituto de Pesquisa Econômica Aplicada (IPEA)
SBS Q.1 – Brasília – DF – Brasil

Abstract. The growing need for information sharing among partnering organizations or members of virtual organizations poses a great security challenge. One of the key aspects of this challenge is deploying intrusion detection systems (IDS) that can operate in heterogeneous, large-scale environments. This is particularly difficult because the different networks involved generally use IDSs that have not been designed to work in a cooperative fashion. This paper presents a model for integrating intrusion detection systems in such environments. The main idea is to build compositions of IDSs that work as unified systems, using a service-oriented architecture (SOA) based on the Web Services technology. The necessary interoperability among the elements of the compositions is achieved through the use of standardized specifications, mainly those developed by IETF, W3C and OASIS. Dynamic compositions are supported through service orchestration. We also describe a prototype implementation of the proposed infrastructure and analyze some results obtained through experimentation with this prototype.

1 Introduction

The popularization of the Internet has brought about a new generation of cooperative, large-scale distributed systems. In this context, one of the trends is the emergence of virtual organizations, which are formed when groups of organizations pool resources and share information in order to achieve common goals. The need for information and resource sharing require a certain level of integration of the networks pertaining to the different organizations. This integration raises a number of security concerns, since most of these networks were architected to operate in isolation, usually behind firewalls and other traffic-filtering devices, and thus require adjustments to cooperate with other networks. This is especially true when one considers that virtual organizations are often established dynamically, which means that such adjustments have to be made on-the-fly.

R. Meersman, Z. Tari et al. (Eds.): OTM 2006, LNCS 4275, pp. 462–479, 2006.

One of the key challenges involves security monitoring and management, more specifically in the area of intrusion detection systems (IDSs). Just as the networks they are deployed on, these systems are seldom designed to cooperate with other, potentially different, IDSs, whether on the same or on separate networks. In a virtual organization, this is particularly troublesome, since a security manager may need to collect information from — or even to act on — other networks in order to analyze suspicious traffic and thwart ongoing attacks. These needs are hard to satisfy with current IDS installations, and a better alternative becomes necessary.

We propose such an alternative in this paper. Our approach, called *IDS composition*, is an infrastructure for combining intrusion detection elements (which can be complete IDS systems or components thereof) distributed across different networks so that they operate in a cooperative fashion in order to provide a unified service. The infrastructure relies on standardized specifications to provide the necessary interoperability across composition elements and is built according to a service-oriented architecture (SOA) based on the Web Services technology [1]. IDS compositions can be dynamically established and reconfigured, enabling them to adjust to changing conditions and providing great flexibility to security administrators; the dynamic aspects of IDS compositions are implemented using service orchestration [2].

Paper contributions. The main contributions of this paper are the following. First, we present an infrastructure for IDS composition based on Web Services that is capable of supporting commercial off-the-shelf (COTS) IDS elements, a crucial feature for providing interoperability and deployability. Second, we introduce the use of service orchestration for building dynamic IDS compositions, a powerful approach that provides security managers with great flexibility in security monitoring and response.

Paper organization. In the next section we review related work. Section 3 introduces our basic composition infrastructure. Section 4 discusses the creation and management of IDS compositions using service orchestration. The evaluation of the proposed infrastructure is presented in Section 5, which describes a prototype implementation and the results of experiments conducted with this prototype. Finally, Section 6 presents some concluding remarks.

2 Related Work

Web services composition applied to intrusion detection systems is a largely uncharted territory, as both Web Services and large-scale IDS are still young research areas. Therefore, we chose to discuss related work on each of those areas separately, examining the few intersection points at the end of Section 2.2.

2.1 Web Services Composition

Web Services provide a standard means of interoperating between different software applications, running on a variety of platforms and/or frameworks [1]. Characteristics such as platform transparency and loose coupling make the Web Services technology a good choice for IDS integration.

Many solutions for creation and management of Web Services compositions have been proposed in the literature, exploring themes such as orchestration, choreography, workflow, and standards; Peltz [2] summarizes the most relevant proposals in this area. Wang et al. [3] provide an overview on Web Services composition, analyzing the possible problems and solutions related to this topic. The problems and requirements for the management of Web Services compositions were discussed by Esfandiari and Tosic [4]. They affirm that the management of service compositions must support a complete lifecycle, including service discovery, composition requirements management, possible service rearrangements and the contracting of services. These requirements are considered in our work.

Orchestration and *choreography* are terms usually applied to describe creation aspects of Web Services compositions [2]. Both terms are used to represent business-oriented processes. A business process is a set of partially ordered steps, whose objective is to reach a goal such as the construction of a large-scale IDS.

An orchestration describes a business process, involving Web Services interactions with both internal and external Web Services of an administrative domain. It includes the business logic (the desired composition behavior) and the order of execution defined by control flows that cross organizations and applications. An orchestration always represents the process flow, controlled from the perspective of one of the parts involved; in an IDS composition, this would be a security administrator's vision of how the IDS elements are organized. A choreography, another term used to describe compositions, concerns the observable interactions of services with their users [5]. A choreography description is a multi-party contract that describes from a global viewpoint the external observable behavior across multiple clients (which are generally Web Services but not exclusively so), where "external observable behavior" is defined as the presence or absence of messages that are exchanged between a Web Service and its clients. Choreography and orchestration are complementary ways for Web Services arrangement.

In this paper, we consider the use of the Web Services orchestration to describe the management steps used to create IDS compositions. We use the Business Process Execution Language for Web Services (BPEL4WS) [6] [7] to write these compositions.

A mechanism for composition management in a global scale was proposed by Vambenepe et al. [8]. Their work consists of using Web Services to configure services in a distributed environment and to monitor the activation of applications. A workflow in BPEL describes the composition of tasks that will be executed.

2.2 Intrusion Detection in Large-Scale Environments

The literature on intrusion detection in large-scale systems is scarce. Conventional IDSs in general are not a good fit for large-scale environments, since usually their design cannot accommodate the exchange of security information across organizations: such information are restricted to the scope of the organization or network where they are collected.

Recent proposals of distributed IDSs [9] [10] [11] generally do not use standard formats and protocols for the communication between intrusion detection

elements. However, the use of common formats is important to provide interoperability between IDSs. Bass's proposal [12] is an example of this. In his proposal, security notifications are generated in a native format and then translated into a single format; however, this format does not follow any existing standard.

Recent standardization efforts related to the exchange of security information are being developed mainly by the IETF through its IDWG and INCH working groups. IDWG is finishing up the Intrusion Detection Message Exchange Format (IDMEF) [13] and Intrusion Detection Exchange Protocol (IDXP) [14] specifications. These efforts aim at the exchange of information among complete IDSs and IDS elements. The INCH group is working on the exchange of information and statistics about security incidents among incident response teams (CSIRTs). The requirements [15] and the data model (IODEF—Incident Object Description Exchange Format) [16] for implementation are still in specification phase. All these specifications are based on XML [17].

The IDWG also defines a general model for intrusion detection [18], as illustrated in Fig. 1. This taxonomy is adopted in our infrastructure. A *sensor* is an element that collects data from one or more data sources. The sensor is setup to forward events to the analyzer. An *analyzer* inspects data collected by a sensor looking for signs of unauthorized or undesired activity or for events that might be of interest to the security administrator. A sensor and an analyzer can be part of the same component. A *manager* manages the various elements of an IDS. Management functions typically include (but are not limited to) sensor configuration, analyzer configuration, event notification management, data consolidation, and reporting. IDS elements can be parts of a monolithic IDS or can be distributed on more than one location. In our model, sensors and analyzers can use IDMEF to send messages to their peer elements in a composition, while managers can use IDMEF or IODEF messages to communicate with other managers.

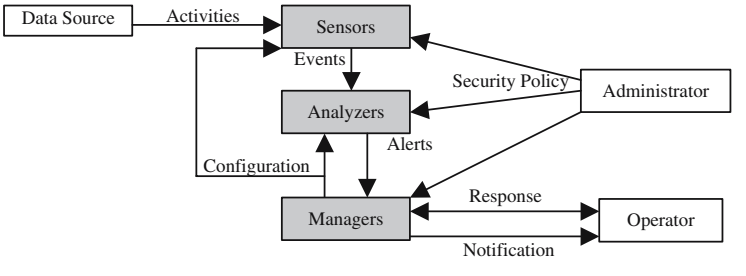


Fig. 1. Basic elements of the IETF intrusion detection model

There are few experiences in the literature or even products that use these recent standards. Snort[1], which is one of the most popular IDSs available, can send alerts in the IDMEF format through a plug-in. However the IDMEF version presently used is not fully compatible with the current specification. DOMINO

[1] http://www.snort.org/

[19] and the STAT family of systems [20] extended the IDMEF format to accommodate their necessities. The use of standards in these cases is limited and the modifications (extensions) are not fully compatible with the original specifications. On the other hand, our IDS composition infrastructure is fully based on the original standard.

A Web Services-based IDS that uses the IDWG model is described in [21]. Unfortunately, the model uses only one method of detection and it centralizes the analysis. Although it uses the IDMEF, originally, it is not possible to integrate it with other IDSs. Our framework may be used to make this integration.

An IDS that is closely related to our proposal is Prelude[2], which is a hybrid IDS which adds and correlates alerts generated by many sensor types from diverse suppliers, distributed across a computer network. Messages are sent over SSL connections, using an IDMEF adaptation. Prelude uses a hierarchical analysis model, which reports events to one or more managers. In the prototype described in Section 5, Snort and Prelude are used as analyzers and sensors.

IDS composition using Web Services was previously introduced in [22], which presented a preliminary version of our composition infrastructure. For the current paper the infrastructure has been expanded, with a new format compatibilization layer that enables the use of commercial off-the-shelf (COTS) IDS elements. Also, this paper introduces and discusses in detail several aspects of the use of service orchestration for building dynamic compositions, a key functionality that was missing from our previous effort.

3 Infrastructure for IDS Composition

In this section we present the service infrastructure for IDS compositions. This infrastructure allows the integration of stand-alone IDSs and of IDS components as elements that form a single, unified IDS composition. We generally assume that these intrusion detection elements are specialized, being responsible for monitoring certain types of activity within a single (sub)network; this closely mirrors the reality of the Intranets found in medium- and large-sized organizations that are connected to the Internet. The infrastructure supports IDS elements supplied by different vendors and which are incompatible with each other (e.g. an analyzer from vendor A that cannot process events generated by a sensor from vendor B).

IDS compositions can cross organizational boundaries, allowing e.g. the sharing of security alerts. Such traffic is usually subject to policies that regulate information flow across organizations. To better accommodate these policies, both the IDS composition elements and the services in our infrastructure are implemented as Web Services.

A composition can be permanent or temporary, and is usually defined with a specific purpose, such as collecting data from a set of sensors, searching various event databases or sharing information about ongoing attacks. Dynamic

[2] http://www.prelude-ids.org/

compositions allow for the adaptation to new situations in distributed large-scale networks.

Since we want to integrate potentially incompatible intrusion detection elements, interoperability becomes an key requirement of the infrastructure. To satisfy this requirement, our work adopts specifications developed by the IETF[3], OASIS[4] and W3C[5] standards organizations.

The security of an IDS composition is essential to its effectiveness. Therefore, we employ a number of mechanisms to guarantee the security of composition elements and also of the information exchanged and processed by them.

3.1 Support Services for IDS Composition

Fig. 2 depicts our infrastructure for IDS composition. At the core of the infrastructure are a set of services based on the Web Services technology. These services support the creation and management of compositions comprised of IDS elements and other applications that support Web Services, either as a native characteristic or through the use a format compatibilization layer. Communication across these WS-based components is implemented using the Simple Object Access Protocol (SOAP) [23] and its underlying transport protocols.

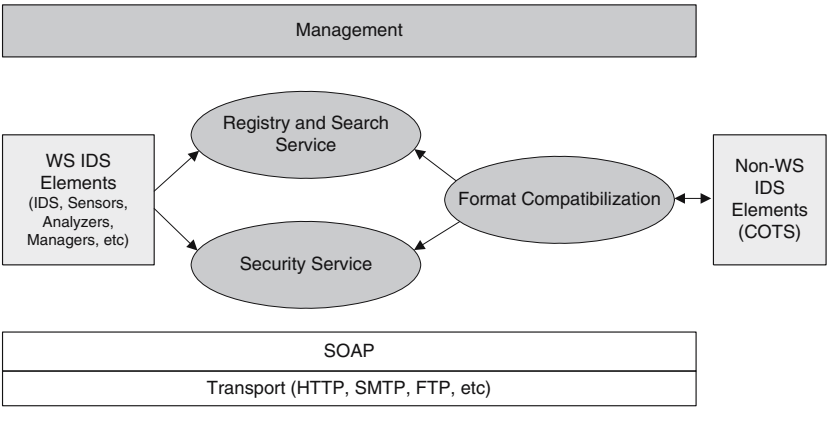


Fig. 2. Infrastructure for IDSs compositions

Information about intrusion detection elements, necessary for these elements to interact with each other, is provided by the Registry and Search Service (RSS), which is based on the Universal Description, Discovery and Integration (UDDI) specification [24]. The description of an element stored in the RSS contains the service identification and its localization. It also supplies information related to

[3] http://www.ietf.org/.
[4] http://www.oasis-open.org/.
[5] http://www.w3.org/.

the policies governing access to the service. Service interfaces are described using the Web Services Description Language (WSDL) [25], a machine-processable format. The location of these interface descriptions is also kept in the RSS. All IDS composition elements must provide their WSDL interface descriptions.

The interactions between IDS elements follow the client/server model, and consist of request/reply exchanges. An element can act as a client (in requester mode), as a server (in provider mode), or as both. The services provided by these elements will depend on their role: according to its specific functionality, a provider may send several replies to a single request, and these replies may not be sent immediately after the corresponding request. For example, when a sensor, acting as a service provider, identifies a suspicious activity, a security alert message is sent to the client service. This monitoring can have a limited lifetime after which the service is no longer available, e.g. after the first occurrence of some activity. But it also can run indefinitely, generating several alerts. In the proposed infrastructure, the IDSs compositions elements communicate through events and alert notifications.

Providing conventional IDS parts with a service interface requires the introduction of a "format compatibilization" layer (Fig. 2). The main role of this layer is to deal with the formats and protocols used in message exchanges in service compositions. Its functions are used, for example, to mediate the communication between conventional IDS parts (without WS support) and the Web Services that make them available to compositions.

Configuration of service elements and message security (security context establishment, XML message encryption and signing, etc.) also use this layer. The compatibilization layer translates the messages generated by IDS elements into a standard format, such as IDMEF [13] or Syslog [26]. The translated messages are enveloped in SOAP, encrypted and signed as specified by the WS-Security standard [27] (using XML Encryption [28] and XML Signature [29]), and sent over HTTP, as shown in Fig. 3.

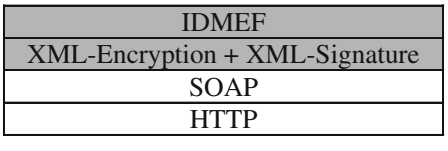


Fig. 3. Encapsulation of intrusion detection messages

In the next subsections, we provide further detail on the services provided by the infrastructure.

3.2 Registry and Search Service

This service is a fundamental component in our proposal. The Registry and Search Service is based on the UDDI specification [24], which adopts a registry approach [1] with emphasis on the creation of administrative domains for

information storage. The specification also defines mechanisms for associating different UDDI servers, providing the necessary scalability to the RSS.

The elements that can be used in IDS compositions must be first registered and described in the UDDI and then made available as Web Services. After that, service elements can then be located to interact with other services in the composition. Fig. 4 depicts the lifecycle of a composition element as a simplified UML class diagram.

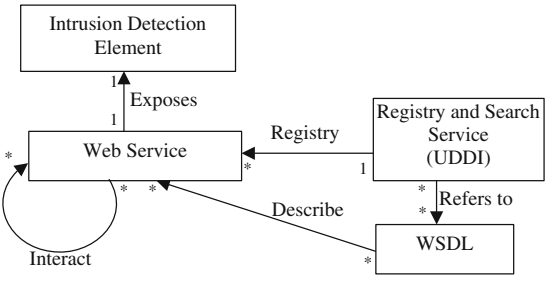


Fig. 4. Lifecycle of a IDS composition element

In the UDDI, *tModel* structures represent reusable concepts, such as Web service types, protocols used by Web services, or category systems. Service interfaces, described by WSDL documents, are obtained from the URLs available in *tModels*. Each service instance is registered in the UDDI using a *businessService* structure, whose *bindingTemplate* substructures contain technical information necessary to use the service and references to *tModels*. The IDS element access is made from the URL indicated by the *accessPoint* tag in a *bindingTemplate* substructure.

Web Services applications are usually oblivious to the location of services and their implementation details. However, these information can be highly important for dynamic IDS compositions, as when one needs to find a sensor at a specific network location. The IDMEF specification defines an *Analyzer* class that identifies and provides detailed information on IDS components. This same class, which is based on XML, is also adopted by other specifications [15] [16] that define standards for exchanging information about security incidents. Therefore, it is naturally used to identify possible IDS composition elements. A *bindingTemplate* structure is used to store details of an IDS element, such as the original element address and its Analyzer class. One or more *tModelInstanceInfo* will indicate the protocols and native formats supported by the element that can be used in a composition.

To facilitate the location of specific types of IDS elements for a composition, we developed a classification scheme for these elements according to their roles. In the UDDI registry, the *categoryBag* structure indicates the element category. Sensor elements follow the taxonomy proposed in [30]. For analyzers and monolithic IDSs, we use recent IDS taxonomies [31] [32] [33] [34] which are combined

according to the architecture of the elements and the detection method they use. These categories are represented by specific *tModels*, stored in the UDDI. Fig. 5 shows an example of an IDS registry in the UDDI. This IDS can be used both as a sensor and as an analyzer; in both cases, its characteristics are identified in *categoryBag* tag.

The registry and search messages are encapsulated in SOAP, using the WS-Security recommended mechanisms.

```
<businessService businessKey="104522bf-f411-0452-a82b-8c7512184005"
  serviceKey="10452c21-8ae1-0452-e610-300113d34ddc">
  <name xml:lang="en">Snort WS1</name>
  <description xml:lang="en">Snort IDS 1</description>
  <bindingTemplates> ... </bindingTemplates>
  <categoryBag>
    <keyedReference keyName="Meta" keyValue="uddi:ids:sensor:informationsource:log:meta"/>
    <keyedReference keyName="Simple rule-based"keyValue="uddi:ids:analyser:
     detectionmethod:signature:programed:simplerulebased" />
    <keyedReference keyName="Passive"
     keyValue="uddi:ids:analyser:behaviorondetection:passive" />
    <keyedReference keyName="Network"
     keyValue="uddi:ids:analyser:informationsource:network" />
    <keyedReference keyName="Real-time"
     keyValue="uddi:ids:analyser:timeofdetection:real-time" />
    <keyedReference keyName="Continuously"
     keyValue="uddi:ids:analyser:data-processing:continuously" />
    <keyedReference keyName="Central location"
     keyValue="uddi:ids:analyser:processinglocus:central" />
    <keyedReference keyName="Central location"
     keyValue="uddi:ids:analyser:collectinglocus:central" />
  </categoryBag>
</businessService>
```

Fig. 5. Example of element registry with category classification at UDDI

3.3 Security Service

The Security Service deals with authentication and access control of the composition elements. In its current version, it is also based on the UDDI. The security mechanisms provided in the UDDI specification are used to authenticate operators and administrators in their interactions with the RSS, and also to provide confidentiality and integrity to the information about IDS elements that is stored in the RSS.

The public keys and the security contexts to be used in the communications between the composition elements follow the WS-Security specification and are obtained from these elements registry in the UDDI through the Security Service. In an element registry, a *bindingTemplate* structure stores a public key or indicates its location. The public and private keys are based on the X.509 model [35].

4 Service Orchestration

In order to take part in a composition, each IDS element must have a Web Service interface (either native or through format compatibilization), and be associated

to a process flow. A process flow defines how an element interacts with other elements and with the services infrastructure as part of the composition. Each composition has a single process flow, which defines all the behaviors available in this composition. The behavior of any given element will depend on the role (sensor, analyzer, or manager) it plays in the composition.

Process flows are described using the Business Process Execution Language for Web Services (BPEL4WS) [6, 7], also called BPEL, which provides an XML-based grammar to describe both the business logic of an element (in a specific role) and the control logic needed to coordinate the composition. This grammar can be interpreted and executed by an orchestration engine controlled by one of the parts involved. The BPEL4WS process execution blocks describe the activities executed inside the composition. There are basic and structured activities. Basic activities are instructions for interactions between Web Services, while structured activities describe the process execution flow.

An IDS composition is created from its representation registered in the UDDI. This registry is contained in a *tModel* structure. Documents indicated in *tModel overview_Doc* tags identify and describe the composition, as shown in Fig. 6. These documents include a WSDL document that describes the interfaces of the composition and an XML document that describes how the services interact and how the elements of the composition are organized. From these two documents it is possible to implement the composition.

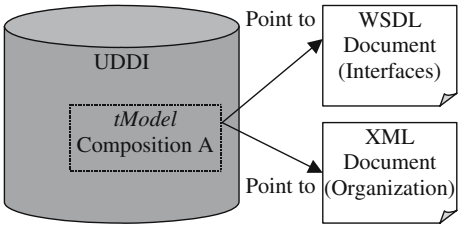


Fig. 6. Composition registry

The organization of the elements of a composition described in the XML document expresses an *orchestration*. We have chosen orchestration over choreography because it provides the flexibility needed to deal with dynamic compositions. An orchestration allows a security manager to define a generic process flow that can be used to form a composition and which remains unchanged when IDS elements are added or removed, while a choreography lacks the flexibility to accommodate such evolution (if elements are added or removed the choreography has to be changed).

Fig. 7 shows a simplified orchestration for an IDS composition (adapted from an example in [2]). The internal steps are interpreted by an orchestration engine that will execute them sequentially or in parallel, as defined by the administrator. The support services are invoked in the order defined by the process flow. In the

same way, the process flow determines the order in which the IDS elements are added to the composition. The tools used for the creation and description of IDS compositions are presented in Section 5.1.

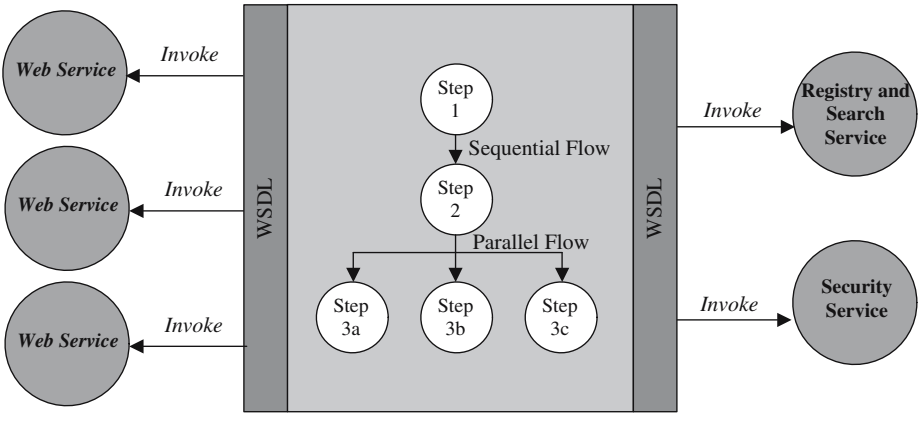


Fig. 7. Using orchestration in an IDS composition

Every IDS composition has one or more Manager Services. These elements allow an administrator (or other responsible person) to interact with the composition. This interaction includes alert visualization and element reconfiguration. To support reconfiguration and activation by Manager Services, all IDS elements must provide interfaces based on the Web Services Distributed Management (WSDM) specification [36], thus allowing services to be configured and monitored using WS-specific management standards. These interfaces are grouped according to their capabilities. The capabilities defined by WSDM include identification, description, manageability characteristics, correlatable properties, metrics, configuration, state, operational status, and advertisement. Each Web Service defines in its WSDL document what capabilities are available. In this work, only the advertisement (event notification) and configuration capabilities are defined for IDS composition elements.

5 Experiments

In order to validate our proposal, we implemented a prototype of the services infrastructure. For this prototype implementation we used BEA Weblogic Workshop 8.1[6], which provides an integrated programming environment, a Web Service server and a UDDI. We also tested the JUDDI[7] and WSDP[8] UDDI Servers,

[6] http://www.bea.com
[7] http://ws.apache.org/juddi/
[8] http://java.sun.com/webservices/downloads/webservicespack.html

in addition to the TomCat WS server TomCat[9] and other tools to support WSDM (Muse[10]) and SOAP (Axis[11]).

The prototype was tested with two popular open-source IDSs, Snort and Prelude, which provided the sensors and analyzers in our composition. We developed one format compatibilization layer for each IDS; this layer is responsible for converting IDS messages to and from the IDMEF standard format[12] and for providing a Web Services interface to the IDS.

5.1 Implemented Composition

Fig. 8 depicts the experiment that was built. This figure shows the composition used for testing, together with the sensors and analyzers obtained from Prelude and Snort. In a sub-net "A", two Snort-based sensor services are configured to send event notifications to a Prelude-based analyzer service. This analyzer service correlates notifications received from the sensors and generates new alerts according to criteria defined by the manager service. In a sub-net "B", a sensor service (Snort) sends its event notifications directly to the manager service.

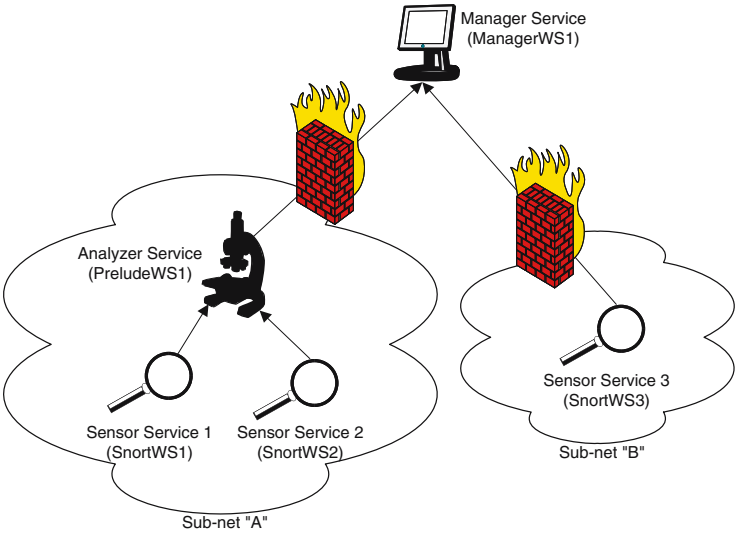


Fig. 8. IDS composition used for testing

The manager service used in our experiments was implemented in Java using resources provided by BEA Weblogic Workshop for creating Web Services. The manager interface follows the format of a web portal. Through this portal it is

[9] http://jakarta.apache.org/tomcat/
[10] http://ws.apache.org/muse
[11] http://ws.apache.org/axis
[12] Prelude uses a binary-encoded IDMEF format. The Snort IDMEF plug-in is not fully compliant with the current IDMEF specification.

possible to manage the composition and its services (elements) in an independent way.

A composition is described using a modeling tool available in Weblogic Workshop. From this description, the composition can be deployed or exported to BPEL with the corresponding WSDL document. From the WSDL and BPEL documents it is possible to recover (rebuild) the IDS composition and its corresponding orchestration. These documents are made available in the composition registry in the UDDI.

5.2 A Generic Procedure for Composition

We developed a generic procedure for the creating and maintaining dynamic IDS compositions. Table 1 presents the steps used to create an IDS composition. Fig. 9 shows the basic operations of an orchestration defined by this procedure, using a graphical interface.

Table 1. Steps for creating an IDS composition

Step	Operation
1	Locate the Manager Service in the UDDI using the parameters specified in the *ClientRequestDiscovery* operation.
2	Execute the *ClientRequestCreate* operation to create a composition registry in the UDDI.
3	Start the Manager Service (invoke).
4	Link the Manager Service to the composition using the *ClientRequestRegistry* operation.
5	Locate the Analyzer Service in the UDDI using the parameters specified in the *ClientRequestDiscovery* operation.
6	Execute the *ClientRequestSubscribe* operation in the Analyzer Service for subscribing to alerts, and send these alerts to the Manager Service.
7	Link the Analyzer Service to the composition using the *ClientRequestRegistry* operation.
8	Repeat steps 5, 6 and 7 for each one of the composition elements (sensors 1, 2 and 3).

In principle, one does not know where services are located, only what functionalities are desired. The *ClientRequestDiscovery* operation (Fig. 9a) is responsible for searching the service in the UDDI, using the Registry and Search Service, according to certain defined parameters. In our tests with the prototype, two parameters have been used: the characteristics of the intrusion detection element, described in the *categoryBag* structure (Sec. 3.2), and the subnetwork address where the element is operating, contained in the Analyzer class.

The *ClientRequestCreate* operation (Fig. 9b) is responsible for the request to create a composition registry. The *Perform* operation verifies the data in the request and inserts the composition registry in the UDDI. The *ClientRequestRegistry* and *ClientResponseRegistry* operations (Fig. 9c) are responsible,

respectively, for the request and the reply of a service publication in the UDDI. These service registries can be inserted in the UDDI or not according to the security policies defined.

ClientRequestSubscribe and *ClientResponseSubscribe* (Fig. 9d) are the operations responsible for service subscription. These operations are based on the *SubscribeRequest* and *SubscribeResponse* operations defined by the standard interfaces of the Web Service Notification (WSN) [37] specification. The *ClientRequestSendAlert* operation (Fig. 9e) is used by services already registered in a composition for sending alerts to the composition. This operation is based on the *Notify* operation of WSN. The orchestration of compositions follows a logical and chronological sequence of the events represented in Fig. 9. From each event it is possible to continue execution or to go to the end of the execution flow (represented by *Condition*). In order to execute the *ClientRequestSendAlert* operation, for instance, it is not necessary to execute the previous events if they have already been executed at least once.

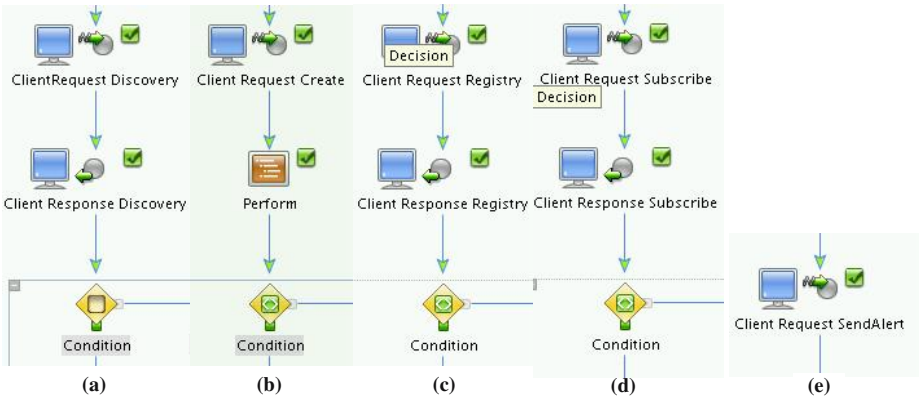


Fig. 9. General composition model

5.3 Tests with the Prototype

The tests with the prototype were conducted on our campus network, using IDS composition elements located in distinct subnetworks. We applied the composition described in Fig. 8 and performed simulated attacks against the test network. The tests were executed in the sequence indicated in Table 2. In this simulation, a user in subnet "B" executes a port scan attack using Nmap[13] against a host in subnet "A". In addition to the tests described, other services could have been activated to isolate the source of the attack and to assess possible damage to the system. An event log database could also have been used to search for evidence that would help to locate and identify the attacker with greater accuracy.

[13] http://www.insecure.org/

Table 2. Description of the tests with the prototype

1. Several possible IDS composition elements were registered in the UDDI.
2. A composition was initially created in subnet "A", with two monolithic Snort-based IDSs (*SnortWS1* and *SnortWS2*) acting as sensors and sending alerts to an analyzer based on Prelude (*PreludeWS1*). The Manager Service (*ManagerWS1*) receives and sends notifications to the security administrator.
3. A user on host 10.2.13.122, located in subnet "B", starts a port scan against a host located in subnet "A".
4. *SnortWS1* (in subnet "A") detects suspicious traffic coming from a host in subnet "B".
5. The composition is modified to include a new sensor located in subnet "B". The *SnortWS3* service is located and activated to collect traffic from the suspicious host and to send events directly to *ManagerWS1*.
6. Alert messages sent by *SnortWS3* confirm the attack.
7. After action is taken to stop the attack, the composition returns to its original configuration.

6 Concluding Remarks

This paper presented the IDS composition approach for intrusion detection in large-scale environments, based on an infrastructure that enables the cooperation of IDS elements across network and organizational boundaries. This infrastructure uses the Web Services technology, and has as its chief advantages the flexibility offered by its support for dynamic IDS compositions and the interoperability provided by its use of standardized specifications for exchanging security alerts and managing compositions and their elements.

Dynamic IDS composition is achieved through service orchestration, and allows compositions to be started and (re)configured as needed, thus supporting the changing requirements that characterize large-scale environments. Interoperability is provided by a format compatibilization layer, which equips IDS elements with a Web Services interface and is responsible for converting IDS messages to and from standardized formats. This integration of diverse standardized specifications is not trivial in practice. The different standards-defining bodies are often in disagreement, and some standards supposed to be complementary are in fact incompatible. Another difficulty is related to incipient standardization efforts.

We believe our approach brings interesting opportunities for cooperation. Partner organizations can cooperate by providing a set of shared intrusion detection elements that can be used for monitoring suspicious activity coming from the various networks involved. This requires adequate specification and monitoring of security policies between the concerned parties. Another possibility is the outsourcing of the security alert analysis task. Analyzer services provided by Computer Security Incident Response Teams (CSIRTs) or other third parties can be activated to receive alerts that are correlated with alerts from other

customers. Benefits for customers include access to global security alerts, accurate statistics about new incidents and parameters for the reconfiguration of their security tools.

In future work, we intend to perform a quantitative assessment of the proposed infrastructure, including its operational and computational costs, and to further refine the security service.

Acknowledgement

The research described in this paper was sponsored by CNPq (Brazil) Project No. 550114/2005-0.

References

1. W3C: Web Services Architecture. W3C Working Group Note 11 (2004)
2. Peltz, C.: Web services orchestration and choreography. IEEE Computer **36**(10) (2003) 46–52
3. Wang, H., Huang, J.Z., Qu, Y., Xie, J.: Web Services: problems and future directions. Web Semantics: Science, Services and Agents on the World Wide Web **1**(3) (2004) 309–320
4. Esfandiari, B., Tosic, V.: Towards a Web Service composition management framework. In: Proceedings of IEEE International Conference on Web Services (ICWS'05), IEEE (2005) 419–426
5. Austin, D., Babir, A., Peters, E., Ross-Talbot, S.: Web services choreography requirements. W3c working draft 11, W3C (2004)
6. Andrews, T., Curbera, F., Goland, Y., Klein, Y., Leymann, F., Roller, D., Weerawarana, S.: Business Process Execution Language for Web Services (2003) Version 1.1 - 5 May 2003.
7. OASIS: Business Process Execution Language for Web Services (2005) Version 2.0 - Committee Draft, 01 September 2005.
8. Vambenepe, W., Thompson, C., Talwar, V., Rafaeli, S., Murray, B., Milojicic, D., Iyer, S., Farkas, K., Arlitt, M.: Dealing with scale and adaptation of global web services management. In: Proceedings of IEEE International Conference on Web Services (ICWS'05), IEEE (2005) 339–346
9. Teo, L., Zheng, Y., Ahn, G.J.: Intrusion Detection Force: An infrastructure for Internet-scale intrusion detection. In: First IEEE International Information Assurance Workshop (IWIA 2003), Germany (2003) 73–88
10. Tolba, M., Abdel-Wahab, M., Taha, I., Al-Shishtawy, A.: GIDA: Toward Enabling Grid Intrusion Detection Systems. 5th IEEE International Symposium on Cluster Computing and the Grid (2005)
11. Leu, F.Y., Lin, J.C., Li, M.C., Yang, C.T., Shih, P.C.: Integrating Grid with intrusion detection. In: Proceedings of AINA'2005. (2005) 304–309
12. Bass, T.: Service-oriented horizontal fusion in distributed coordination-based systems. IEEE MILCOM 2004 (2004)
13. Debar, H., Curry, D., Feinstein, B.: The intrusion detection message exchange format. Internet Draft draft-ietf-idwg-idmef-xml-16, IETF (2006)

14. Feinstein, B., Matthews, G., White, J.: The Intrusion Detection Exchange Protocol (IDXP). Internet Draft draft-ietf-idwg-beep-idxp-07, IETF (2002)
15. Keeni, G., Danyliw, R., Demchenko, Y.: Requirements for the format for incident information exchange (FINE). Internet Draft draft-ietf-inch-requirements-08.txt, IETF (2006)
16. Danyliw, R., Meijer, J., Demchenko, Y.: The Incident Object Description Exchange Format data model and XML implementation. Internet Draft draft-inch-ietf-iodef-08.txt, IETF (2006)
17. Bray, T., Paoli, J., Sperberg-McQueen, C.M.: Extensible Markup Language (XML) 1.0 (third edition)". W3C Recommendation (2004)
18. Wood, M., Erlinger, M.: Intrusion Detection Message Exchange Requirements. Internet Draft draft-ietf-idwg-requirements-10, IETF (2002)
19. Yegneswaran, V., Barford, P., Jha, S.: Global intrusion detection in the DOMINO overlay system. In: NDSS, San Diego, California, USA, The Internet Society (2004)
20. Vigna, G., Valeur, F., Kemmerer, R.A.: Designing and implementing a family of intrusion detection systems. In: Proceedings of the 9th European Software Engineering Conference, Helsinki, Finland (2003) 88–97
21. Park, S., Kim, K., Jang, J., Noh, B.: Supporting interoperability to heterogeneous IDS in secure networking framework. The 9th Asia-Pacific Conference on Communications (APCC 2003) **2**(21-24) (2003) 844–848
22. Brandão, J.E., Mafra, P.M., Fraga, J.S.: A new approach for IDS composition. In: Proceedings of the IEEE International Conference on Communications (ICC 2006), IEEE (2006)
23. W3C: Soap version 1.2. W3C World Wide Web Consortium (2003)
24. OASIS: UDDI Version 3.0.2. OASIS UDDI Spec Technical Committee Draft (2004)
25. W3C: Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language. W3C Working Draft (2005)
26. Lonvick, C.: The BSD Syslog protocol. Request for Comments 3164, Internet Engineering Task Force (2001)
27. OASIS: Web services security: SOAP message security 1.0 (2004) http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf.
28. Imamura, T., Dillaway, B., Simon, E.: XML Encryption syntax and processing. W3c recommendation, W3C (2002)
29. Eastlake, D., Reagle, J., Solo, D.: (Extensible Markup Language) XML-Signature syntax and processing. Request for Comments 3275, Internet Engineering Task Force (2002)
30. Alessandri, D., Cachin, C., Dacier, M., Deak, O., Julisch, K., Randell, B., Riordan, J., Tscharner, A., Wespi, A., Wüest, C.: Towards a taxonomy of intrusion detection systems and attacks. MAFTIA Deliverable D3, EU Project IST-1999-11583 Malicious- and Accidental-Fault Tolerance for Internet Applications (MAFTIA) (2001) Version 1.01.
31. Axelsson, S.: Intrusion Detection Systems: A survey and taxonomy. Technical Report 99-15, Department of Computer Engineering, Chalmers University of Technology, SE-412 96 Göteborg, Sweden (2000)
32. Debar, H., Dacier, M., Wespi, A.: Towards a taxonomy of intrusion detection systems. Computer Networks (Amsterdam, Netherlands: 1999) **31**(8) (1999) 805–822
33. Debar, H., Dacier, M., Wespi, A.: A revised taxonomy for intrusion detection systems. Annales des Telecommunications **55**(7–8) (2000) 361–378

34. McHugh, J.: Intrusion and intrusion detection. International Journal of Information Security **1**(1) (2001) 14–35
35. ITU-T: ITU-T recommendation X.509 (1993)
36. OASIS: Web Services Distributed Management: Management Using Web Services (MUWS 1.0) Part 2 - Web Services Distributed Management: Management of Web Services (WSDM-MOWS) 1.0. OASIS Web Services Distributed Management (WSDM) TC (2004)
37. OASIS: Web services base notification 1.3. OASIS Web Services Notification (WSN) TC (2005)

An Adaptive Probabilistic Replication Method for Unstructured P2P Networks

Dimitrios Tsoumakos and Nick Roussopoulos

Department of Computer Science
University of Maryland
College Park, MD 20742, USA

Abstract. We present *APRE*, a replication method for unstructured Peer-to-Peer overlays. The goal of our method is to achieve real-time replication of even the most sparsely located content relative to demand. *APRE* adaptively *expands* or *contracts* the replica set of an object in order to improve the sharing process and achieve a low load distribution among the providers. To achieve that, it utilizes search knowledge to identify possible replication targets inside query-intensive areas of the overlay. We present detailed simulation results where *APRE* exhibits both efficiency and robustness over the number of requesters and the respective request rates. The scheme proves particularly useful in the event of flash crowds, managing to quickly adapt to sudden surges in load.

1 Introduction

Peer-to-Peer (hence P2P) computing represents the notion of sharing resources available at the edges of the Internet. Its success can still be largely attributed to file-sharing applications which enable users worldwide to exchange locally maintained content. A basic requirement for every P2P system is fault-tolerance. Since the primary objective is resource location and sharing, we require that this basic operation takes place in a reliable manner. In a variety of situations, the distributed and dynamic nature of the target environments stress the system's ability to operate smoothly. For example, the demand for certain content can become overwhelming for the peers serving these objects, forcing them to reject connections. *Flash crowds*, regularly documented surges in the popularity of certain content, are also known to cause severe congestion and degradation of service [1]. Failing or departing nodes further reduce the availability of various content. Consequently, resources become scarce, servers get overloaded and throughput can diminish due to high workloads.

Data replication techniques are commonly utilized in order to remedy these situations. Replicating critical or frequently accessed system resources is a well-known technique utilized in many areas of computer science (distributed systems, databases, file-systems, etc) in order to achieve reliability, fault-tolerance and increased performance. Resources such as content, location of replicas, routing indices, topology information etc, are cached/replicated by multiple nodes, alleviating single points of contact in routing and sharing of data. This has the additional benefit of reducing the average distance to the objects. Replication can be performed in a variety of manners: Mirroring, Content Distribution Networks (CDNs [2, 3]), web caching [4], etc.

R. Meersman, Z. Tari et al. (Eds.): OTM 2006, LNCS 4275, pp. 480–497, 2006.

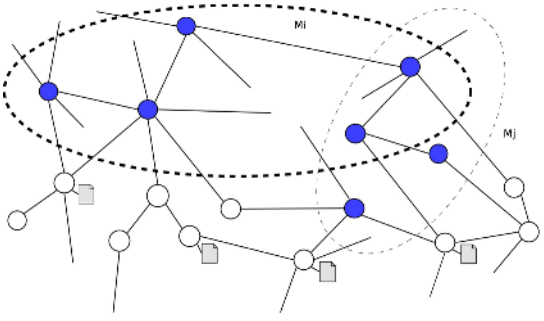


Fig. 1. Part of the overlay network of our model. Dark nodes inside the bold dotted ellipse represent $\mathcal{M}_i$, while those inside the thin dotted ellipse represent $\mathcal{M}_j$. Peers with a file attached also serve objects i or j.

However, these approaches often require full control and provide static replication. Static replication schemes require a priori knowledge of the popularity/workload distribution in order to compute the amount of replicas needed. In large scale unstructured P2P networks, peers usually operate on local knowledge, having variable network connectivity patterns and no control over the induced topology or workload. Data availability and efficient sharing dictate replication in this challenging environment. Structured P2P systems (DHTs) provide with the state necessary to accurately identify the paths that requests take. However, such information is not available in unstructured overlays. File-sharing applications implicitly handle replication through object downloads, while some force their users to maintain the new replicas for the benefit of others. Yet, this does not tackle the issue of real-time replication responsive to workload for unstructured environments.

In this work we present *APRE* (Adaptive Probabilistic REplication), a replication method based on soft-state probabilistic routing indices. Our approach focuses on providing an adaptive solution to the problem of availability together with minimizing the instances of server overloads and serious service degradation. We intend for our system to "expand" and "contract" its resources according to the workload as perceived locally. New replicas are created in areas of high demand in the overlay, thus disposing of the need of advertising them. Moreover, this will be done in a completely decentralized manner, with minimal communication overhead and using absolutely affordable memory space per node.

1.1 Our Framework and Overview of APRE

We assume a *pure* Peer-to-Peer model, with no imposed hierarchy over the set of participating peers. All of them may serve and request various objects. Each peer locally maintains its own collection of objects, as well as a local view of the system. Ignoring physical connectivity and topology from our talk, we generally expect peers to be aware of their one-hop neighbors in the overlay, while maintaining any other protocol-specific information (e.g., search indices, routing tables, etc). The system is expected to exhibit a dynamic behavior, with peers entering and leaving at will and also inserting/removing

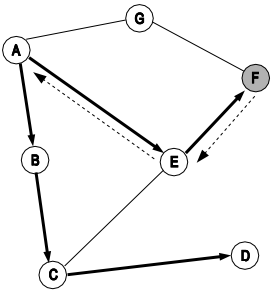
objects from their repositories. The overlay structure will also be affected, since nodes are not guaranteed to connect to the same neighbors each time.

As a motivating example, assume an unstructured P2P system, where peers share and request replicated resources. Objects are assumed to be requested regularly, e.g., results of a live sports meeting, weather maps, security updates, real time aggregated statistics, software, etc. There exist some nodes (similar to the web servers or mirror sites in the Internet) that provide with fresh content, but their connectivity or availability varies, as happens with all other network nodes. Peers that are interested in retrieving the newest version of the content conduct searches for it in order to locate a fresh or closer replica.

Figure 1 gives a graphic representation of our system. For each object i, there exists a set of peers called the *server set* $\mathcal{S}_i = \{s_{i_1}, s_{i_2}, \ldots, s_{i_k}\}$ that serve the specific object. These are the nodes that, at a given time, are online, store object i and are willing to share it. A subset of $\mathcal{S}_i$, the *mirror set* $\mathcal{M}_i \subseteq \mathcal{S}_i$ (the shaded peers) represents the set of peers that, if online, *always* serve i. This does not imply that all peers in $\mathcal{M}_i$ will always be online, their connectivity in the overlay will remain the same, or that they will never refuse connections. But we can assume, without loss of generality, that these nodes will be mostly available. Our assumption is not unrealistic: Imagine that these servers can represent mirror sites/authority nodes that provide with up-to-date content. Nevertheless, they are not guaranteed to be always on-line, nor do they provide similar services. Apart from the mirror set, other peers that already host or have recently retrieved an object can serve requests for it (nodes with files attached to them in Figure 1). A server set comprises of these nodes plus the corresponding mirror set.

Naturally, peers may belong to server or mirror sets for multiple objects. While this is a symmetric environment, it is clear that nodes will exhibit different sharing abilities. A variety of parameters, including storage and CPU capability, popularity of stored objects, system workload, connectivity, etc, contribute to this fact. Some of these factors remain more or less static over time (e.g., processing power or the maximum available bandwidth of a host), while others change dynamically. In this work, we focus on two of these parameters, namely workload and object popularity as they are manifested through the request rate λ. It is obvious that servers of popular (or temporally popular) items receive a larger number of requests, which can possibly affect their sharing ability as well as the system's behavior.

Given this general framework, our goal is to design and implement a replication protocol that will provide efficient sharing of objects (in terms of providing low load operation), scalability and bandwidth-efficiency. *APRE* is a distributed protocol that automatically adjusts the replication ratio of every shared item according to the current demand for it. By utilizing inexpensive routing indices during searches, loaded servers are able to identify "hot" areas inside the unstructured overlay with a customizable push phase. Chosen nodes receive copies thus sharing part of the load. Under-utilized servers become freed and can host other content. The rationale behind *APRE* is the tight coupling between replication and the lookup protocol which controls how searches get disseminated in the overlay. By combining the Adaptive Probabilistic Search (*APS*) state with *APRE*, we are able to identify in real-time "hot" or "cold" paths and avoid the need of advertising constantly created replicas. Furthermore, we show that this method



Indices	Initially	After walkers finish	After the updates
A→B	30	20	20
B→C	30	20	20
C→D	30	20	20
A→E	30	20	40
E→F	30	20	40
A→G	30	30	30

Fig. 2. Node A searches for an object stored at node F using APS (pessimistic) with two walkers. The table shows how index values change. X→Y denotes the index value stored at node X for neighbor Y relative to the requested object.

provides a very robust replication with minimum change in the server set per replication cycle.

2 Probabilistic Resource Location

2.1 Probabilistic Search

We now briefly describe the *APS* [5] search method, which is the basis for our replication scheme. In *APS*, each node keeps a local index consisting of one entry for each object it has requested per neighbor. The value of this entry reflects the relative probability of this node's neighbor to be chosen as the next hop in a future request for the specific object. Searching is based on the deployment of k independent walkers and probabilistic forwarding. Each intermediate node forwards the query to one of its neighbors with probability given by its local index. Index values are updated using feedback from the walkers. If a walker succeeds (fails), the relative probabilities of the nodes on the walker's path are increased (decreased). The update procedure takes the reverse path back to the requester and can take place either after a walker miss (*optimistic* update approach), or after a hit (*pessimistic* update approach). Figure 2 shows an example.

APS exhibits many plausible characteristics as a result of its *learning* feature. Every node on the deployed walkers updates its indices according to search results, so peers eventually share, refine and adjust their search knowledge with time. Walkers are directed towards objects or redirected if a miss or an object deletion occurs. *APS* is also bandwidth-efficient: It induces zero overhead over the network at join/leave/update operations and displays a high degree of robustness in topology changes.

2.2 Utilizing Search Indices

One interesting observation is that the values of the stored indices are refined as more searches take place, enabling the network to build a useful soft-state. After some queries take place, paths with large index values connect the requesters to the content providers and identify query-intensive areas inside the overlay.

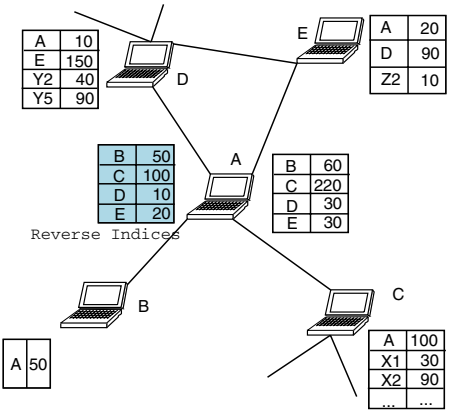


Fig. 3. Graphic explanation of the reverse indices. The filled table represents the reverse index values stored at node A, which coincide with the APS index values that nodes B,C,D,E store regarding A.

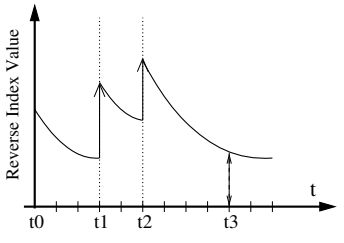


Fig. 4. Example of exponential decay in the reverse index values

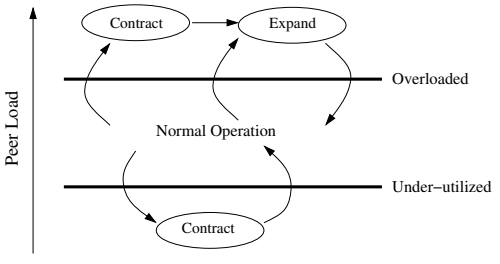


Fig. 5. State transitions in our system

APS keeps an index value for each neighbor. Extending this, each peer P also maintains the index values that its neighbors hold relative to P. If $X \rightarrow P$ denotes the index value stored at node X concerning neighbor P for a particular object, then peer P must know $X \rightarrow P$, for each neighbor X. These values can be made known to P during the search phase: Whenever a search is conducted and X forwards to P, it piggybacks $X \rightarrow P$. We call these new stored values the *reverse indices*, to distinguish them from the indices used by *APS* in searches (see Figure 3).

Reverse indices are used by nodes in order to forward messages along high demand paths in an unstructured overlay. These messages can be forwarded either to the top-k or probabilistically selected neighbors on a hop-by-hop basis. They are discarded either when their TTL value reaches zero or if they are received by a node more than once due to a cycle. Reverse indices get updated during searches, but this is not enough: There may be peers that have searched for an object and built large index values in the past, but are no longer interested in it. If searches are no longer routed through those peers, the respective reverse index values will not be updated and will remain high.

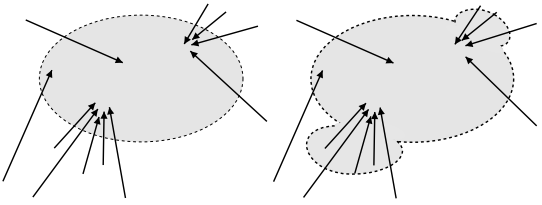


Fig. 6. The shaded oval represents a server set for a specific object. Our system expands by creating replicas inside two areas where demand (depicted by arrows) is high.

Algorithm 1. *Expand*

1: **if** Replica i at node s reaches its limit **then**
2: $P \leftarrow FindPossibleServers(i)$; $\{P \cap \mathcal{S}_i = \emptyset\}$
3: $Activate(i)$ at $Y \subseteq P$ {Replicate at a subset of the nodes in the high-demand area}
4: **end if**

To correct this situation, we add an *aging* factor to the reverse indices, which forces their values to decrease with time. Peers need to keep track of the time that a reverse index was last updated in order to acquire its correct value before using it. When a peer receives a search message, it sets the corresponding reverse index to the piggybacked value and its last modified field to the time of receipt. We describe this process in Figure 4. The value of the index decreases exponentially, while two searches at times t_1, t_2 reset its value. A push message received at time t_3 will use the value as shown in the figure. The last modified value is also reset when a reverse index is used, since a peer computes its current value before using it.

3 Our "Expand-Contract" Technique

Our main goal is to provide a completely decentralized mechanism through which the system will adaptively expand its replica size when demand is increased and will shrink when demand will fall. *APRE* is based on two basic operations: *Expand* and *Contract*.

The high-level behavior of our system can be described using a simple model (Figure 5): In normal mode, nodes can adequately serve requests and also retrieve objects. As load increases due to incoming requests, some reach their self-imposed limits. By invoking the *Expand* process, we aim at bringing the node status back to normal and lower the average load for a specific object through the creation of more replicas. Normal operation through the distribution of load will not be necessarily achieved in a single step. Consider, for example, that a peer initiating *Expand* may receive requests for multiple objects. Expanding with respect to one of them will probably lower its load, but will not necessarily bring its level back to normal. As load decreases, nodes can free up space (and the respective resources) and thus share a bigger portion of the workload.

Conversely, consider that one or more subsets of $\mathcal{S}_i$ have recently received very few requests for object i. This practically means that an amount of their storage space is under-utilized. They could remove i to free up space or replace it with another object of

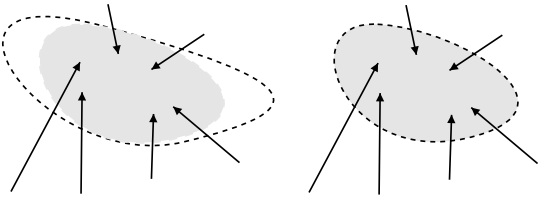


Fig. 7. Due to low demand in certain regions of the server set (depicted as white areas inside the dotted line), our system contracts its replica set

Algorithm 2. *Contract*

```
1: if (Replica i at node s is under-utilized) or (s receives Activate(j)) then
2:    i ← ChooseObject(); {i is among the candidates for eviction}
3:    Deactivate(i);
4:    if (s received an Activate(j)) then
5:       Activate(j);
6:    end if
7: end if
```

higher demand. We have to stress here the point that the system will not force any peer to store or serve an object until this becomes necessary. Peers with available storage can play that role. *Contract* will also be invoked when a peer is called to join $\mathcal{S}_i$ but cannot do so without exceeding its limits (e.g., available storage). Note that peers can still choose to reject a certain action, e.g., refuse to remove an object in order to serve a new one.

Algorithm 1 describes the high-level operation of the *Expand* process. It is invoked by peers receiving more requests than those that they are willing to accept. Overloaded peers have to identify the set P, i.e., candidate nodes for replication inside query intensive areas. A subset Y of these nodes is selected and, upon their agreement, the new replicas are transfered (*Activate*). Figure 6 shows an example of our system expanding in response to increased demand for a specific object. On the left, we see some initial server set (gray oval) and the demand for i (arrows from various parts of the network). Servers in two areas are overloaded with requests, thus forcing extra replicas in those two areas to be activated. $\mathcal{S}_i$ expands, as we see on the right part of the picture, in response to the current demand for object i.

Algorithm 2 describes our *Contract* process. It is invoked by a peer that either receives a low amount of requests for the object(s) it serves or is requested to serve a more popular one but cannot do so without freeing up some space. In any case, peers stop serving the object(s) that fall into these categories (*Deactivate*). Function *ChooseObject* decides at each point which object should be deactivated at nodes that have decided to serve a new object (i.e., received an *Activate*) but have reached their storage capacities. Natural choices are to have the new replica replace the least recently requested or the least popular one. Figure 7 shows that two areas of the server set (the areas inside the dotted line) do not receive any requests for object i. This leads to the contraction of $\mathcal{S}_i$

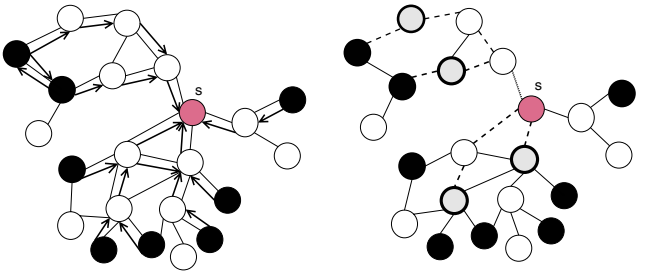


Fig. 8. After searches for an object at s take place, reverse index values are updated and a push phase creates new replicas inside areas of high demand (dotted links)

which is now the gray oval on the right part of the figure. Our goal is to achieve a system behavior that resembles the buffer management techniques in databases: Viewing the P2P network as a large buffer, we want to decide (in a distributed and dynamic manner) the ratios of objects in the buffer according to user-specified queries (i.e., workload).

3.1 Protocol Implementation

In this section we describe the actual implementation of the *APRE* protocol as described by the *Expand* and *Contract* algorithms. We assume that servers measure load and perform replication on a per-object basis, at the same level of granularity with lookup and reverse indices.

The conditions of line 1 in Algorithms 1 and 2 describe when *Expand* or *Contract* are initiated. We believe that each peer can independently choose when to initiate an expansion or when to deactivate a replica. Therefore, there is no need for any message exchange between servers. We assume that each server s defines the maximum number of requests that each object i can accept per time unit $Limit^{up}_{s,i}$. If it receives less than $Limit^{down}_{s,i}$ requests for object i, this replica is deactivated/deleted from the node's cache without any further communication. Obviously, the total maximum capacity for server s is equal to $\sum_i Limit^{up}_{s,i}$, where i refers to every object that s serves.

In order to discover candidate new servers to host replicas of i (i.e. locate subset Y), whenever the local load for object i (measured in requests per time unit) $\lambda^s_i(t)$ exceeds the limit $Limit^{up}_{s,i}$, the respective server s issues a special message which is forwarded to k neighbors with the k highest reverse index values. The push message also contains the amount of overload $D_i(t) = \lambda^s_i(t) - Limit^{up}_{s,i}$. Each node that receives this message, independently decides whether to join S_i according to our implemented replication policy. This phase continues with each intermediate node forwarding this message to k neighbors in a similar fashion until either its TTL value reaches zero or a duplicate reception is detected. Figure 8 shows an example of our scheme at work: Black nodes represent requesters of the item held at node s. *APS* searches are depicted by arrows. In the push phase, paths with high index values are visited (links with dotted lines). The new shaded nodes with bold outline represent possible replicas created.

Each node on the path independently decides whether it will join S_i according to our replication policy. Currently, we have implemented three: *FurthestFirst*, *ClosestFirst*

and *Uniform*. In *FurthestFirst*, the probability of a node joining S_i increases with the message distance, while the opposite occurs in *ClosestFirst*. In *Uniform*, all nodes are given the same probability. After subset Y has been identified, replicas are transmitted and activated.

In order for *APRE* to adapt to various workloads and avoid system oscillation at the same time[1], we introduce a *scaled* replication policy: We regulate the number of replicas activated per push phase according to the amount of overload for object i, $D_i(t)$, as observed by the server initiating the push at time t. To achieve that, we define a set of intervals $\{d_1, d_2, \ldots, d_m\}$ that group the different values of D_i. Each interval $d_k : \{(l_k, u_k), \{p_{k_1}, p_{k_2}, \ldots, p_{k_{TTL}}\}\}$ is defined by an upper and lower value and TTL probability values, one for each hop distance. For the interval limits, we require that $l_1 < u_1 = l_2 < u_2 \ldots < u_m$. When a server receives a push message, it joins $\mathcal{S}_i$ with probability p_{k_δ}, if $l_k < D_i \leq u_k$ and the message has travelled δ hops. Probability values increase as D falls into higher number intervals (i.e., $p_{k_\delta} < p_{(k+1)_\delta}$). Thus, a heavily overloaded server will create more replicas than a less overloaded one and marginally overloaded peers will not alter $\mathcal{S}_i$ significantly. We note here that each server locally estimates $\lambda_i(t)$, the number of requests for object i per time unit.

4 Results

We test the effectiveness of *APRE* using a message-level simulator written in C. Object requests occur at rate λ_r with exponentially distributed inter-arrival times. At each run, we randomly choose a *single* node that plays the role of the initial $\mathcal{M}_i \equiv \mathcal{S}_i$ set and a number of requesters, also uniformly at random. Results are averaged over several tens of runs. We utilize 10,000-node *random* graphs with average node degrees around 4 (similar to gnutella snapshots [6]) created with the *BRITE* [7] topology generator.

To evaluate the replication scheme, we utilize the following metrics: The average load Λ observed as the number of received requests per second in $\mathcal{S}_i$. To measure the disparity of the load distribution, we measure its standard deviation σ_Λ. A high value for σ_Λ indicates that load is very unevenly balanced across $\mathcal{S}_i$. Besides the size of the server set, we also keep track of the number of replica activations/de-activations. Frequent changes in $\mathcal{S}_i$ incur huge overheads in terms of messages and bytes transferred.

APRE Parameters: We assume that $(Limit_s^{up}, Limit_s^{down}) = (18, 3)$ requests/sec, for each server s. When *Expand* is initiated, peers forward to 2 neighbors with the largest reverse index values. We utilize a scheme with 3 distinct intervals for values of D: $[0-5]$, $(5-20]$ and $(20-\infty)$. While we experimented with more fine-grained granularities, the results did not vary considerably. Finally, we assume no item can be replicated at more than 40% of the network nodes[2].

We compare our method against a random replication scheme as well as path-replication as applied by Freenet [8] (and in effect by *lar* [9]). In the first case, we randomly create the same number of replicas as our method. In path replication (hence *path-cache*), each time a server is overloaded we replicate the object along the

[1] Replicas with perceived load a little above or below the limits, frequently entering and leaving $\mathcal{S}_i$.

[2] This external condition simulates the natural limitations in storage that exist in most systems.

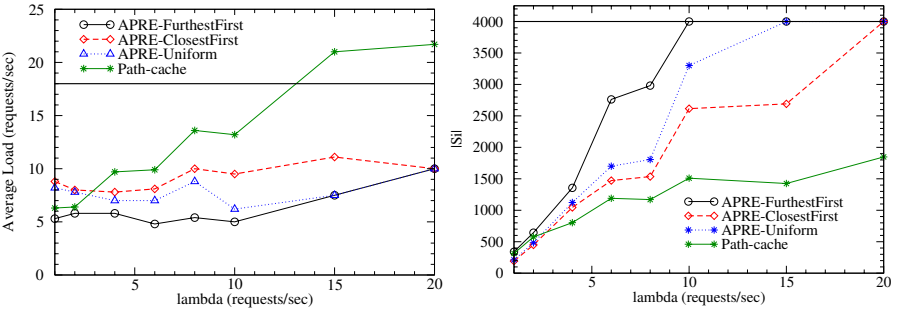


Fig. 9. Variation in Λ and $|\mathcal{S}_i|$ over increasing λ_r values

reverse path to the requester. In every case, the *APS* method is used for lookups, while in *path-cache* replicas are also deactivated using our *Contract* scheme. Obviously, by varying the push method and the replication probabilities, *APRE* can behave either as *path-cache*, random, or in between with a variable rate of replica creation per workload. This allows for full customization according to the system's primary objects, namely low load (more replicas) or space (replicas only where necessary).

4.1 Basic Performance Comparison

For our default setting, we assume 2000 random requesters as we vary their request rate. Figure 9 presents the variation in Λ and $|\mathcal{S}_i|$.

APRE effectively expands $\mathcal{S}_i$ in order to accommodate increased demand and manages to keep all servers into the *Normal Operation* zone, well below $Limit^{up}$ (identified by the bold horizontal line). Our first observation is that *FurthestFirst* achieves lower Λ values by creating more replicas than *ClosestFirst*. Downstream paths during the "push" phase contact an increasing number of nodes as their distance from the initiator increases, thus giving *FurthestFirst* an increased probability of replication. *Uniform* behaves in-between, creating replicas equally at all distances. *Path-cache* exhibits a steeper increase in Λ and fails to keep its value within the acceptable region for large λ_r. Choosing only successful walks to replicate along, quickly "saturates" the frequently used paths with replicas. Increased demand merely forces the algorithm to utilize a few more paths, which is the reason why this method fails to increase the replica set to meet the limits.

It is interesting to note that *APRE* exhibits small σ_Λ values, ranging from 3.3 to 11. It increases to 14.9 only when $\lambda_r = 20/sec$ (see Figure 10). These values are either smaller or at most comparable to Λ, a good indication of load balancing. On the other hand, randomly placing the same number of replicas yields significantly worse load distributions, with σ_Λ values roughly twice as large. This is a clear indication of the need for correct placement inside structureless multi-path overlays. Finally, *path-cache* behaves in-between, with larger deviation values than *APRE* that converge as load increases. This happens since both methods base their replication on paths connecting requesters and servers.

Moreover, we show that *APRE* achieves a much more robust replication. The stability of the server population constitutes an important metric to the evaluation of a replication

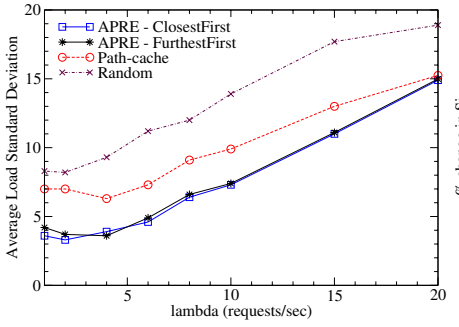


Fig. 10. Variation of σ_Λ vs. variable λ_r

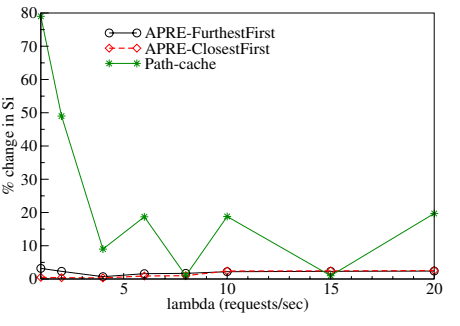


Fig. 11. Percentage of change in $|\mathcal{S}_i|$

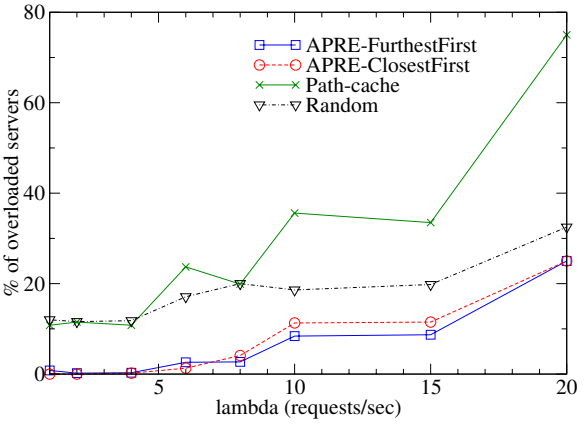


Fig. 12. Ratio of overloaded servers vs. variable λ_r

scheme. This is measured by the average ratio of new replicas entering the server set per replication phase over the size of the server set. This quantity approximates the amount of marginally under-utilized replicas in the overlay: Receiving few requests, they get deactivated. Server overloads force them to get re-activated, producing an oscillating effect. Obviously, this is a highly undesirable situation: network and local resources are burdened by a multiplicative factor, since replicas need both control messages and data transfer for reactivation. Figure 11 shows that *APRE* is particularly robust, altering at most 3% of $\mathcal{S}_i$ per push phase, while *Path-cache* oscillates and performs poorly in most runs. altering a large percentage of the server set. The variability in the amount of oscillation is due to the effect we described before: An increase in the demand is not always followed by an increase in the number of replicas. In these situations, the existing ones receive the extra amount of requests (assisted by the *APS* scheme), thus reducing the marginally idle servers.

Figure 12 displays the average percentage of overloaded servers at any time for all three methods. Our technique clearly outperforms the two competing methods: For

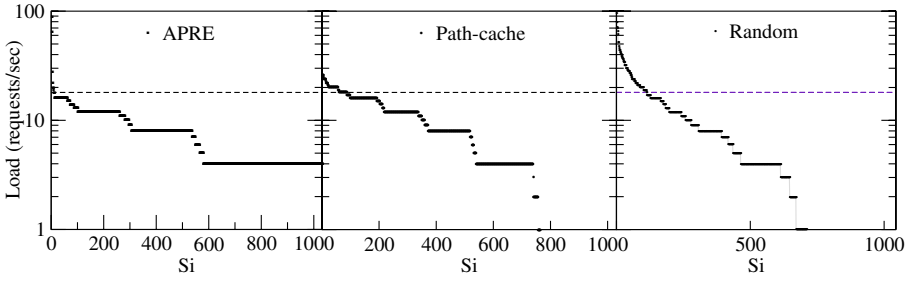


Fig. 13. $\mathcal{S}_i$ load distribution for $\lambda_r = 4/sec$

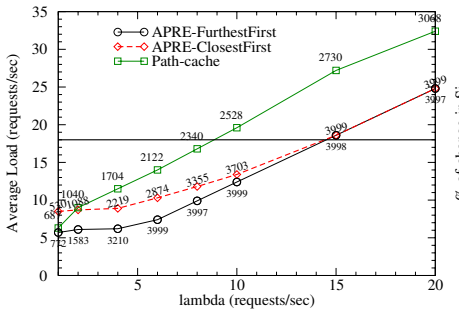


Fig. 14. Variation in the average load vs. variable λ_r (5000 requesters)

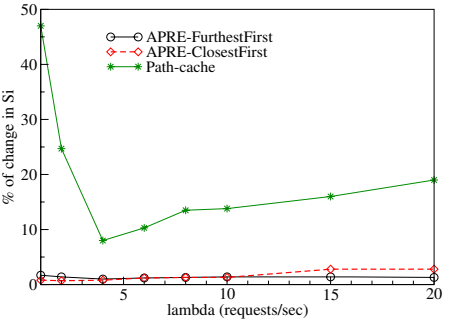


Fig. 15. Percentage of change in $\mathcal{S}_i$ and σ_Λ vs. variable λ_r (5000 requesters)

$\lambda_r < 10/sec$, less than 4% of servers are overloaded, while about 10% and 25% are documented as overloaded for the largest demand. *Random*, having the same number of servers, exhibits twice as many overloaded nodes. Even though the learning feature of *APS* helps in redirecting queries, yet the load cannot be evenly distributed. *Path-cache* shows the worst performance (at least 3 times larger ratio of overloaded peers than *APRE*), reaching 75% at the highest λ_r value. Replicating closer to requesters creates, as we saw, more service points, thus marginally reducing the number of overloaded instances for *FurthestFirst* (*Uniform* exhibits the same curve).

Figure 13 shows the load distribution of every server s at a random point in time ($\lambda_r = 4/sec$). Servers are sorted in decreasing order of load. Our method exhibits a less steep curve and, more importantly, has only 3 servers above $Limit^{up}$, compared to 54 for *path-cache*. Random replication causes even more unbalanced load.

The same experiment is repeated with 5,000 requesters, which constitute 50% of the overlay (see Figures 14 and 15). *APRE* again keeps the system within its limits, except for the two cases where even the largest replica set cannot achieve that (75k and 100k total queries per second). Still, our method shows remarkable stability in the $\mathcal{S}_i$ population for both strategies, while *Path-cache* exhibits even worse performance than before.

Finally, Figure 16 shows how Λ and $|\mathcal{S}_i|$ vary with time, using *ClosestFirst*. For all values of λ_r, *APRE* manages to bring Λ to a steady state in few time steps, a state which

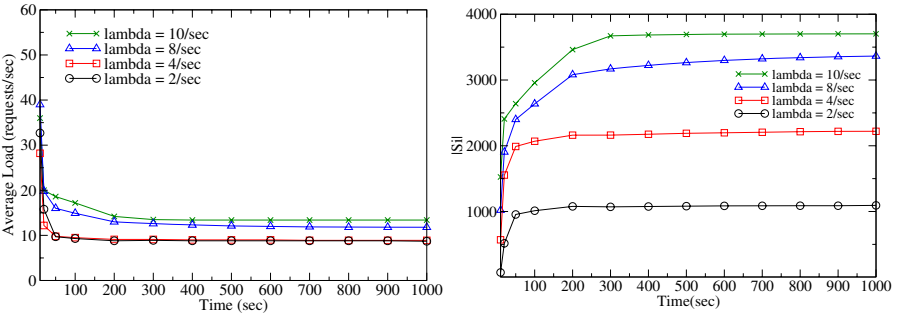


Fig. 16. Λ and $|\mathcal{S}_i|$ over time for 5000 requesters and multiple λ_r values

is hence maintained with almost no deviation. The same is true for the size of $\mathcal{S}_i$, with the exception that for high total demand, it takes longer to reach the steady state. This is due to the fact that there is a limit to the maximum amount of replication per push phase for our method (as there is for *path-cache*) that causes the delay in reaching the constant values.

4.2 Flash Crowds

Thus far we established our basic premise, that replication along high demand paths in the overlay proves an effective and highly robust solution in a variety of metrics and workloads. Although our method does not explicitly offer load-balancing, it achieves a well-proportionate load distribution. We also showed that our method is advantageous to randomly replicating inside the network or merely choosing a single path and fully replicating along it. In the first case, few replicas receive the majority of requests, while in the second case the composition of the replica sets changes very frequently. Our method outperforms both alternatives by keeping fewer peers over the sharing limit and showing less disparity in the distribution of load among servers.

In the next experiment, we examine the behavior of our method when we experience a sudden surge in the workload. This is often referred to as a *flash crowd*, an unexpected rise in requests towards specific item(s), typically due to some newsworthy event that just took place. Flash crowds have been regularly documented in web traffic history (e.g., September 11th) and are known to cause severe congestion at the network layer. Requests may never reach content servers while others do so with significant delays, caused by packet loss and retransmission attempts. Content holders are unable to handle the volume of requests while end-users experience long delays and failures in their queries.

To simulate a similar situation, we start our system with 500 requesters querying at rate $\lambda_r = 2/sec$. At time t=401sec, 10 times as many requesters start querying for this item at rate $\lambda_r = 12/sec$. The parameters return to their initial values at time t=601sec. On average, the total demand during the flash-crowd period increases by a factor of over 70. Note that this is the worst case scenario, when simultaneously both requesters and rates increase. We present the variations in Λ and $|\mathcal{S}_i|$ in the first 2 graphs of Figure 17.

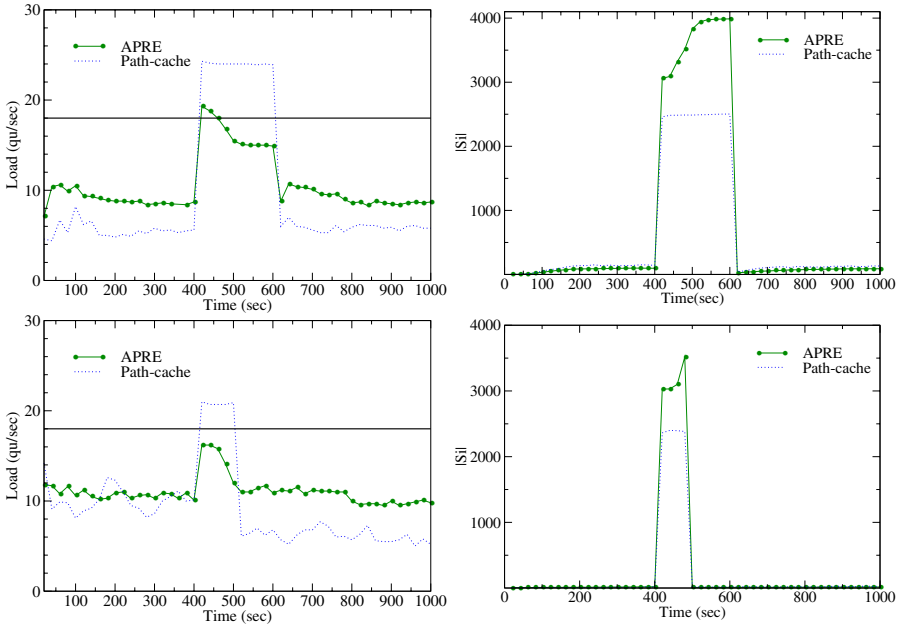


Fig. 17. Effect of flash crowds in Λ and $\mathcal{S}_i$ in two different settings

APRE promptly manages to meet the surge in requests by increasing the replication ratio by a factor of 30. Excluding a very short window due to our mechanism's response, our method succeeds in keeping the load factor below the limit (with $\sigma_\Lambda < 10$) and steady through time. At both moments of load change, replicas are activated and deactivated rapidly to meet the extra requests or reduced traffic. While *path-cache* shows similar response speed, it creates more servers in the low-workload period and less than the minimum number required to keep content providers from overloading during the surge.

The next two figures show how the same two metrics vary in a more challenging flash-crowd setting. Here, we initially set 500 requesters with $\lambda_r = 1/sec$, while for time $t \in (400, 480]$, 5000 requesters query at rate $\lambda_r = 10/sec$. On average, the workload inside the overlay increases by a factor of 120. Our results show that, even for shorter and steeper changes, *APRE* very successfully adapts to the surge in requests. On average, $\mathcal{S}_i$ is expanded by a factor of 175 in order to reduce and balance load (our results document an average $\sigma_\Lambda \simeq 8.6$).

4.3 Other Experiments

We test our method on a set of 4,000-node power-law graphs created with Inet-3.0 generator [10]. These graphs have an average degree of $d = 4.3$ (maximum degree equals to 855), while over 30% of the nodes have only one neighbor. Figure 18 shows how Λ varies with time for both replication strategies used in *APRE* using 1000 or 2000 nodes as requesters.

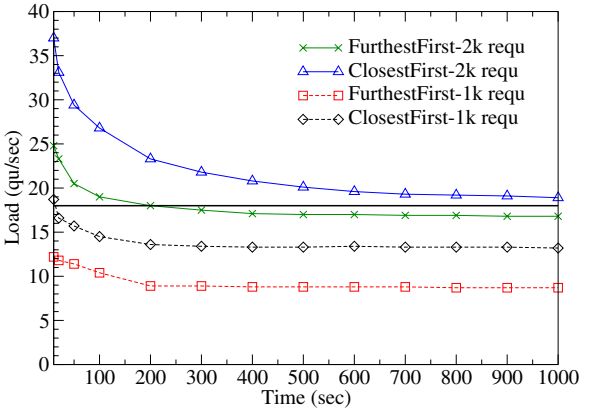


Fig. 18. Average load for 1k and 2k requesters in power-law topologies ($\lambda_r = 6/sec$)

These topologies noticeably affect performance compared to our previous tests. Even for average-range λ_r values, Λ moves close to the overload line, while expansion shows diminished ability to extend $\mathcal{S}_i$. This is consistent with results documented in previous work [5]. The tested topologies offer fewer paths between servers and clients, while a large percentage of the nodes only have one neighbor. This also explains why *FurthestFirst* outperforms *ClosestFirst*. Favoring replication close to the requesters quickly saturates available nodes due to lack of alternate paths. Nevertheless, its is worth to notice that our method still manages to keep Λ at lower levels. Even at the 2k-*ClosestFirst* run, where $\Lambda > Limit^{up}$, 14% of the servers are overloaded compared to 20% by *path-cache*.

We must note here that the replication protocol is not always responsible for overloaded servers. In many occasions, the amount of demand or the overlay connectivity cannot allow for more extensive or balanced replication. As we experiment with more densely connected graphs, *APRE* performs inside the load limits where it failed to do so over more sparse overlays.

In the accompanying technical report [11], we also present results on our method's behavior over variable maximum replication ratios and different values for the load limits, as well as load-balancing analysis based on a different metric.

5 Related Work

Replication is a well-known technique utilized to achieve high availability and fault-tolerance in large-scale systems. While applied to a variety of contexts, we focus in the area of distributed (P2P) systems.

Structured overlays (DHTs) balance routing between network nodes, due to the nature of the hashing functions used. Moreover, in systems like CFS [12] and PAST [13], each item (or chunk of it) is replicated on a set number of network nodes. DHTs take advantage of the routing structure, which in effect allows for almost-deterministic paths between two nodes, thus identifying "hot" areas easily. Nevertheless, DHTs are not

optimized for skewed access patterns and direct such traffic to few nodes responsible for popular content.

DHash [14] is a replication method applied on Chord [15]. The protocol allows for r copies to be stored at the r immediate successors of the initial copy's home. In [16], the authors propose the storage of at most R replicas for an object. Their location is determined by a hash function, allowing requesters to pro-actively redirect their queries. The work in [17] proposes replicating one hop closer to requester nodes as soon as peers are overloaded.

Lar [9] is a DHT-based approach similar to *APRE*, in that it adapts in response to current workload. Overloaded peers replicate at the query initiator and create routing hints on the reverse path. Hints contain some other locations that the content has been previously replicated, so queries are randomly redirected during routing. The method takes advantage of the DHT substrate in order to place the hints. Our scheme does not attempt to re-route queries or shed load to the initiator, but rather places replicas inside forwarding-intensive areas using multiple paths. Moreover, the state kept is accessible at any time, not only at the time of the query arrival.

HotRoD [18] presents a load-balancing approach for DHTs handling range queries in RDBMSs. It is based on a locality-preserving DHT and replication of overloaded arcs (consecutive modes on the DHT ring). [19] employes minimization function that combined high availability and low load to replicate video content inside a DHT. The approach requires knowledge of peer availabilities, workload and data popularity. In [20], the authors show that load-balancing based on periodic load statistics suffer from oscillation. By directing queries towards the the maximum capacity replica store, they show that both heterogeneity and oscillation issues are tackled. This method, nevertheless, assumes prior contact of an authority server which provides with a list of existing replicas. Moreover, replicas regularly advertise their maximum capacities to the network.

There has also been considerable amount of work on flash crowd avoidance. In [21], overloaded servers redirect future requests to mirror nodes to which content has been pushed. This approach does not tackle the issue of which node to replicate to. PROOFS [22] explicitly constructs a randomized overlay to locate content under heavy load conditions or unwilling participants. In effect, the method relies on the combination of custom overlay creation and a gossiping lookup scheme to locate objects and does not involve replication. Finally, the work in [23] discusses static replication in unstructured networks, given content popularity and random walks as a lookup method.

6 Conclusions

In this paper we presented our adaptive replication scheme for unstructured Peer-to-Peer systems based on probabilistic soft state. *APRE* aims at providing a direct response to workload changes, by creating server points in needy areas or releasing redundant servers in areas of low demand. Our approach couples lookup indices together with an aging mechanism in order to identify query intensive areas inside the overlay. Peers then individually decide on the time and extent of replication, based on local workload computation.

In this work, we show that it is important to couple replication with the search protocol in unstructured systems. Random replication performs poorly with informed lookup schemes, unless extra state is added to enhance searches. Applying *APRE* over a scheme such as *APS* mitigates this problem. *APS*-indices store local, per-object state to direct queries to objects. While peers only keep metadata about their neighbors, this information can be used to identify, hop-by-hop, where the queries are coming from. Moreover, our scheme is highly customizable allowing control of both the size and the location (as defined through reverse-indices) of replication.

Through thorough simulations, we show that *APRE* is extremely robust in eliminating server overloads while minimizing the communication overhead and balancing the load. Specifically, we show that replicating along the reverse path is an extreme case of our protocol. By effectively discovering all reverse paths, *APRE* manages to distribute content proportional to demand in a variety of overlays and workloads. Finally, we show that our method succeeds in creating a very stable server set with minimal amount of oscillation.

Acknowledgments

This material is based upon work supported by, or in part by, the U.S. Army Research Laboratory and the U.S. Army Research Office under contract/grant number DAAD19-01-1-0494.

References

1. Jung, J., Krishnamurthy, B., Rabinovich, M.: Flash crowds and denial of service attacks: Characterization and implications for CDNs and web sites. In: WWW. (2002)
2. Dilley, J., Maggs, B., Parikh, J., Prokop, H., Sitaraman, R., Weihl, B.: Globally Distributed Content Delivery. IEEE Internet Computing (2002)
3. Freedman, M., Freudenthal, E., Mazires, D.: Democratizing Content Publication with Coral. In: NSDI. (2004)
4. http://www.squid-cache.org/: Squid Web Proxy Cache
5. Tsoumakos, D., Roussopoulos, N.: Adaptive Probabilistic Search for Peer-to-Peer Networks. In: 3rd IEEE Intl Conference on P2P Computing. (2003)
6. Ripeanu, M., Foster, I.: Mapping the gnutella network: Macroscopic properties of large-scale peer-to-peer systems. In: IPTPS. (2002)
7. Medina, A., Lakhina, A., Matta, I., Byers, J.: BRITE: An Approach to Universal Topology Generation. In: MASCOTS. (2001)
8. Clarke, I., Sandberg, O., Wiley, B., Hong, T.: Freenet: A Distributed Anonymous Information Storage and Retrieval System. Lecture Notes in Computer Science (2001)
9. Gopalakrishnan, V., Silaghi, B., Bhattacharjee, B., Keleher, P.: Adaptive replication in peer-to-peer systems. In: ICDCS. (2004)
10. Jin, C., Chen, Q., Jamin, S.: Inet: Internet Topology Generator. Technical Report CSE-TR443-00, Department of EECS, University of Michigan
11. Tsoumakos, D., Roussopoulos, N.: APRE: A Replication Method for Unstructured P2P Networks. Technical Report CS-TR-4817, University of Maryland (2006)
12. Dabek, F., Kaashoek, M., Karger, D., Morris, R., Stoica, I.: Wide-area cooperative storage with CFS. In: SOSP. (2001)

13. Rowstron, A., Druschel, P.: Storage Management and Caching in PAST, A Large-scale, Persistent Peer-to-peer Storage Utility. In: SOSP. (2001)
14. Cates, J.: Robust and efficient data management for a distributed hash table Master's thesis, Massachusetts Institute of Technology, May 2003.
15. Stoica, I., Morris, R., Karger, D., Kaashoek, F., Balakrishnan, H.: Chord: A scalable Peer-To-Peer lookup service for internet applications. In: SIGCOMM. (2001)
16. Waldvogel, M., Hurley, P., Bauer, D.: Dynamic replica management in distributed hash tables. Technical Report RZ–3502, IBM (2003)
17. Stading, T., Maniatis, P., Baker, M.: Peer-to-peer caching schemes to address flash crowds. In: IPTPS. (2002)
18. Theoni Pitoura, Nikolai Ntarmos, and Peter Triantafillou: Replication, Load Balancing and Efficient Range Query Processing in DHTs. In: EDBT. (2006)
19. Poon, W., Lee, J., Chiu, D.: Comparison of Data Replication Strategies for Peer-to-Peer Video Streaming. In: ICICS. (2005)
20. Roussopoulos, M., Baker, M.: Practical load balancing for content requests in peer-to-peer networks. Technical Report cs.NI/0209023, Stanford University (2003)
21. Felber, P., Kaldewey, T., Weiss, S.: Proactive hot spot avoidance for web server dependability
22. Stavrou, A., Rubenstein, D., Sahu, S.: A lightweight, robust p2p system to handle flash crowds. In: ICNP. (2002)
23. Lv, C., Cao, P., Cohen, E., Li, K., Shenker, S.: Search and Replication in Unstructured Peer-to-Peer Networks. In: ICS. (2002)

Towards Truthful Feedback in P2P Data Structures

Erik Buchmann, Klemens Böhm, and Christian von der Weth

Universität Karlsruhe (TH), Germany
{buchmann, boehm, weth}@ipd.uni-karlsruhe.de

Abstract. Peer-to-Peer data structures (P2P data structures) let a large number of anonymous peers share the data-management workload. A common assumption behind such systems is that peers behave cooperatively. But as with many distributed systems where participation is voluntary, and the participants are not clearly observable, unreliable behavior is the dominant strategy. This calls for reputation systems that help peers choose reliable peers to interact with. However, if peers exchange feedback on experiences with other peers, spoof feedback becomes possible, compromising the reputation system. In this paper we propose and evaluate measures against spoof feedback in P2P data structures. While others have investigated mechanisms for truthtelling recently, we are not aware of any studies in P2P environments. The problem is more difficult in our context because detecting unreliable peers is more difficult as well. On the other hand, a peer can observe the utility of feedback obtained from other peers, and our approach takes advantage of this. To assess the effectiveness of our approach, we have conducted extensive analytical and experimental evaluations. As a result, truthful feedback tends to have a much higher weight than spoof feedback, and collaboration attacks are difficult to carry out under our approach.

1 Introduction

Peer-to-Peer systems (P2P systems) are distributed systems consisting of many nodes in open, coordinator-free communities. Peers typically are known by pseudonyms, which they can replace at little or no cost. P2P systems do not have a central instance that could observe the behavior of peers. Thus, reputation systems [1] to identify and penalize misbehaving peers are crucial building blocks of all kinds of P2P systems.

Reputation systems assign each peer a reputation value, be it positive or negative. A reputation value is an aggregate of positive feedback or complaints from other participants that have observed the behavior of the peer in the past. Clearly, we cannot expect that nodes issue only truthful feedback. A peer may wish to discredit others which have complained about it, or attackers could try to harm nodes by issuing spoof feedback. For instance, [2] has observed similar behavioral patterns at eBay. However, while others have investigated mechanisms for truthtelling recently [3,4,5], we are not aware of any studies in P2P environments.

This paper proposes and evaluates measures for truthful feedback for one particular kind of P2P system, namely P2P data structures (a.k.a. P2P overlay networks, distributed hash tables, etc. [6]). Such structures let a large number of peers share the data-management and query-processing workload. Designing mechanisms against spoof feedback in P2P data structures is challenging, more than for other P2P systems.

R. Meersman, Z. Tari et al. (Eds.): OTM 2006, LNCS 4275, pp. 498–515, 2006.

To lookup a data object, several peers must cooperate, and the lookup fails if only one of them is not reliable. With other P2P systems in turn, there typically is only one peer that carries out a service or a well-defined part of it. A related issue is that it is difficult to identify the defector if a lookup request is not processed properly. Consequently, generating truthful feedback is more difficult as well. Further, a lookup in P2P data structures consists of operations that are relatively simple. This means that reputation management must be relatively simple as well so that it does not become disproportionately expensive. Another issue is that P2P data structures have good scalability characteristics, and reputation management must not get in the way of this. In addition to these complications specific to P2P data structures, there are further 'more general' issues: Each node may change its behavior at any time and can behave differently with different peers. Thus, negative feedback on cooperative nodes and positive feedback on unreliable nodes typically exists. We cannot readily distinguish it from spoof feedback.

This paper makes the following contributions: First, we describe the particular requirements that an approach against spoof feedback in reputation systems for P2P data structures must fulfill. We then describe our approach. It is a characteristic of P2P data structures that a peer can observe the utility of feedback obtained from other peers, and our approach takes advantage of this. For instance, a peer which has forwarded a query to a certain node and has obtained a proper query result may conclude that complaints about that node were wrong and positive feedback was correct. With our approach, each peer uses such clues to derive weighting factors both regarding the issuers of feedback items and the peer the feedback refers to. Second, we provide an evaluation of our approach, both with an analytic model and by means of experiments. The analysis confirms that the differentiation between useful and less useful feedback is effective. We point out the analysis is rather general, i.e., leaves aside the details of the particular underlying reputation system. The experiments address issues which are difficult to examine analytically, such as collusion attacks and dynamicity issues. The experimental results are positive as well. For instance, our approach is effective against collusion attacks in realistic settings. Third, the article features a discussion of the applicability of our measures against spoof feedback to other reputation systems and application scenarios.

The remainder of this article is organized as follows: Section 2 describes the technical background, followed by a description of our approach in Section 3. Section 4 will analyze the approach, Section 5 will evaluate it. Section 6 reviews related work. Finally, Section 7 provides a discussion of the applicability of our approach, and Section 8 concludes the paper.

2 Background

This section briefly reviews the characteristics of P2P data structures and reputation systems and provides a short description of a reputation system which we will use as a basis for our experiments. The section also quantifies the damage spoof feedback may cause in P2P data structures without any countermeasures.

Content-Addressable Networks. P2P data structures (a.k.a. P2P overlay, distributed hashtable) administer huge sets of (key, $value$)-pairs on top of a large physical

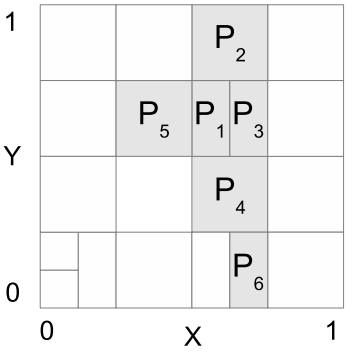


Fig. 1. Two-dimensional CAN

network. *Content-Addressable Networks (CAN)* [7] are a prominent variant of P2P data structures. Other instances are P-Grid [8], Viceroy [9] or Chord [10] which differ primarily with regard to contact selection and routing topology; cf. Section 6. We point out that the presented measures are independent from the specific P2P data structure. However, when presenting our results, we use a CAN for sample calculations and experiments.

A CAN is a distributed system that consists of many nodes (peers). Each peer can issue queries for any data object stored in the CAN, but it is supposed to store data and participate in the evaluation of queries as well. Each CAN node is responsible for a certain zone of the key space, and it knows all neighbors, i.e., peers responsible for adjacent zones of the key space. The key space is an n-dimensional torus of Cartesian coordinates in the unit space. It is independent from the underlying physical network topology. The assignment of zones of the key space to peers results from the CAN construction protocol. A peer which wants to join the CAN finds a random node that is already in the CAN. That node splits its zone, keeping one half and reassigning the other half to the new node.

The key space of the CAN in Figure 1 is two-dimensional. Node P_1 is responsible for Zone ([0.5; 0.5), [0.625; 0.75)) of the key space, i.e., it knows all $(key, value)$-pairs where $key \in$ ([0.5; 0.5), [0.625; 0.75)). The neighbors in the contact list of Node P_1 are Nodes P_2, P_3, P_4, P_5. Since the key space is mapped on a torus, Node P_2 is a neighbor of Node P_6.

Every data object maps to a point in the key space. Accordingly, each operation ($query$, $insert$, $update$, $delete$) in the CAN refers to a point in the key space. For example, a query is the key of a particular (key, value)-pair, and its result is the value of the pair. Query processing in CAN is a variant of *greedy forward routing*. A node that has issued a query first checks if it can answer the query from its zone. Otherwise, it forwards the query to the neighbor in its contact list whose distance to the query key is minimal. The procedure recurs until the query arrives at the peer that can answer it. The peer then sends the result back to the issuer. In a d-dimensional CAN with N peers, a number of $l = d/4 \cdot N^{1/d}$ participate in the processing of a lookup on average.

Incentives Mechanisms for Cooperation in Structured P2P Systems. Research on P2P data structures has tacitly assumed that peers follow the protocol. But participation actually is voluntary, and uncooperative behavior is the dominant strategy (in the economic sense of the word). An uncooperative peer does not follow the protocol of the P2P data structure, e.g., it drops incoming messages. In comparison to P2P systems based on flooding, e.g., Kazaa[1] or gnutella[2], P2P data structures are more vulnerable to uncooperative behavior. Few uncooperative peers can significantly reduce reliability of a P2P data structure. For example, in a CAN consisting of $N = 10,000$ peers with $d = 4$ dimensions, $l = d/4 \cdot N^{1/d} = 10$ peers on average forward a query (cf. [7]). Now suppose that the CAN contains $u = 500$ peers which do not forward any incoming query message. Then the probability to obtain a query result is only $(1-u/N)^l \approx 60\%$.

FairNet [11] is our proposal for a reputation system that renders uncooperative behavior unattractive. Peers in FairNet, the *feedback issuers*, generate and distribute *feedback items*. Such items are the observations of the feedback issuers regarding a particular transaction. Thus, a feedback item consists of a positive or negative statement and contains the identifiers of the issuer and the peer the feedback refers to (the *feedback subject*). Only peers with a number of positive feedback items above a threshold value are allowed to participate in the P2P data structure. Peers can obtain positive feedback in short time by carrying out proofs of work [12,13]. A proof of work is a problem that is easy to formulate, and the solution is easy to verify, but solving it requires a lot of resources.

Each peer maintains a private local *reputation repository* for feedback on its neighbors. The repository has a capacity of s feedback items per subject. If an item is added to a repository that is already full, one item in the repository will be replaced. Consequently, as soon as a peer starts to behave unreliably, negative feedback items tend to replace positive ones. Based on the feedback items in its local repository, each peer can derive an individual *reputation value* for each feedback subject. Peers do share feedback: A peer that has observed cooperative behavior of another peer generates a feedback item and stores it in its local repository. The next time the peer sends out a message, it attaches recent feedback items whose subject is a neighbor of the recipient of the message. Therefore the repositories do not only contain feedback generated by the maintainer of the repository, but also feedback forwarded by adjacent peers. In the following, we refer to such intermediate peers as *forwarders*.

Note that the ratio of positive and negative feedback items in a repository on an uncooperative peer does not exactly follow its failure probability. For example, if a peer does not handle 50% of all incoming messages, it does not obtain 50% positive and 50% negative feedback as well. One reason for this is that not only this peer, but also any other peer later in the sequence of forwarders can drop the query. This results in negative feedback on all peers in the sequence.

The Impact of Spoof Feedback. We know from previous work [11] that the measures outlined in the previous subsection are effective even against peers which process messages properly at a variable rate. For example, a peer which does not work off 10% of

[1] http://www.kazaa.com
[2] http://www.gnutella.com

all incoming queries ends up with more than the double effort compared to a peer that handles all queries properly. (The additional effort is the result of a higher number of proofs of work, in order to remain in the CAN.) However, the experiments show also that dishonest peers issuing spoof feedback can impair the effectiveness of the reputation system significantly.

Suppose that a peer maintains a repository of size s that consists of feedback on only one peer. If feedback is truthful, p_{pos} is the probability that an arbitrary feedback item in the repository is positive. The peer assigned to the repository is deemed reliable if the repository contains at least t positive feedback items, i.e., if $s \cdot p_{pos} \geq t$. Any feedback item has been issued by one of a peers. Now we wonder: How many dishonest peers are necessary to affect the reputation of one peer? Let $x \leq a$ be a number of dishonest peers issuing spoof positive feedback. The overall probability for positive feedback now changes to $\hat{p}_{pos} = \frac{a-x}{a} \cdot p_{pos} + \frac{x}{a} \cdot 1$. Equating this with $s \cdot \hat{p}_{pos} \geq t$, we obtain an estimate of the rate of $\frac{x}{a}$ dishonest peers required to induce positive feedback into the repository such that an uncooperative peer is above the threshold:

$$\frac{x}{a} \geq \frac{\frac{t}{s} - p_{pos}}{1 - p_{pos}} \tag{1}$$

For example, consider a FairNet instance with a repository size of $s = 10$ and a threshold $t = 6$. In this setup each uncooperative peer does not forward or answer 50% of all incoming messages. We know from previous work [11] that this results in a probability of $p_{pos} \approx 0.13$ for uncooperative peers. Equation 1 now tells us that an uncooperative peer is deemed cooperative if at least 54% of the peers it has interacted with issue spoof positive feedback.

3 Measures Against Spoof Feedback

This section motivates the requirements that an approach against spoof feedback in reputation systems for P2P data structures must fulfill. The section also describes our approach with its measures and data structures. We stress that the presented approach does not depend on particular implementations of reputation systems or P2P data structures. Instead, the peers just need to know the nodes which forwarded the feedback, the feedback subjects and the correlation between the feedback and the transaction outcomes; no matter how the implementation handle this. Section 7 provides a discussion on the applicability of our approach.

Approaches Against Spoof Feedback – Requirements. Measures to detect and avoid spoof feedback in a reputation system for P2P data structures must meet the following requirements:

Effectiveness. Obviously, the most urgent issue is the effectiveness of the detection of spoof feedback. Effectiveness means that it does not pay off to issue spoof feedback in any case. There are two worst-case scenarios that might impair the effectiveness: First, there are situations where the distinction between spoof feedback and truthful feedback is not feasible. If a transaction fails with a probability of 50%, any feedback is correct with a probability of 50% as well. Second, peers may run collusion

attacks to feed spoof feedback into the repositories of others. When such an attack takes place, it can be the case that the majority of the peers displays dishonest behavior.

Short response times. The time required to adapt to new situations is an important criterion in any P2P system where peers can change their behavior at any time. Peers can gain advantages during the period of time required to detect such changes. This period of time needs to be as small as possible.

Filtering 'wrong' feedback. There are several reasons why honest peers can sometimes generate wrong feedback. For instance, a cooperative peer may forward a message to an uncooperative neighbor only once, because it does not know any better as yet. If the peer obtains negative feedback but does not forward to the uncooperative neighbor again, the feedback could be seen as spoof. Thus, our approach should differentiate between spoof feedback and feedback that is wrong in spite of best intentions.

Tamper-resistant design. The measures must not introduce new 'holes' which dishonest and/or unreliable peers can exploit. Therefore the measures should rely on local operations as much as possible, in contrast to other peers, which could be dishonest.

Preserving trust relationships that already exist. The idea that a peer either generates spoof feedback all the time or not at all is too undifferentiated. For instance, a peer can generate spoof feedback on selected neighbors only, or wrong feedback could be the result of successful attacks. Thus, we strive for an approach that does not break existing trust relationships after having observed wrong feedback items from one forwarder. Instead, it differentiates between useful and spoof feedback from the same forwarder.

Low resource consumption. P2P data structures aim to process large numbers of small transactions. It is acceptable that a few transactions get lost due to unreliable peers. On the other hand, a measure against spoof feedback must not slow down the processing of transactions due to excessive resource consumption.

Overview. With our approach, each peer individually determines the weight of the feedback. In particular, a peer can assign different weights to each combination of subject and forwarder. The weights depend on the differences or similarities between the transaction outcomes observed and the outcomes predicted by the feedback. In P2P data structures, the feedback is used to identify a reliable peer to forward a query to. Here, the weights ensure that messages go to reliable peers only, even in the presence of dishonest peers issuing positive spoof feedback.

We now explain briefly the rationale behind our design decisions. A peer needs to associate feedback items with the forwarder they have come from. The assignment helps the peer to reduce the impact of spoof feedback and to determine the weight of future feedback coming from that peer. At first sight, we could have associated feedback with the issuer instead of the forwarder. Namely, the issuer is responsible for the feedback it has generated. However, the forwarder is able to manipulate incoming feedback items, and it can decide which feedback is forwarded and which one is not, i.e., apply some kind of censorship. In other words, the receiver of a feedback item can only pin down the last forwarder of the item with certainty, but not the issuer. Further, one might ask

why there are separate weights for each forwarder and each feedback subject. This is because a peer which forwards useful feedback on one feedback subject might forward spoof feedback on another one.

Data Structures. We now specify the data structures required to implement our approach against spoof feedback on top of an existing reputation system. Our approach introduces two variables individually maintained by each peer, *weighting factors* and *transaction logs*. The log is the history of all recent transactions handled by the peer, i.e., it contains the identifier of a transaction and the peer the query was forwarded to. A peer also maintains a weighting factor $w_{\sigma,\phi}$ in the interval $[0;1]$ for each feedback subject σ and forwarder ϕ.

In addition, two data structures implement the reputation system as described in Section 2, namely *feedback items* and *reputation repositories*. Assuming the presence of such data structures does not restrict the applicability of our approach: [14] indicates that the referred structures are common for most of the P2P reputation systems.

How to Weight Reputation Values. When a peer wants to compute the reputation value of a particular node, it first calculates several auxiliary reputation values, based on the feedback from the different forwarders. It then aggregates these auxiliary values using the weighted average. Let P_σ denote the set of all peers that have forwarded feedback for subject σ, and let $r(\sigma, i)$ be a function that computes the auxiliary reputation value for peer σ based on feedback from peer i.[3] The reputation value v then is as follows:

$$v_\sigma = \frac{\sum\limits_{i \in P_\sigma} r(\sigma, i) \cdot w_{\sigma,i}}{\sum\limits_{i \in P_\sigma} w_{\sigma,i}} \tag{2}$$

Equation 2 ensures that feedback with a low weight does not affect the reputation value significantly. Thus, in P2P data structures the messages go to reliable peers only, even in the presence of dishonest peers issuing positive spoof feedback.

Updating the Weighting Factors. Having observed the outcome of a transaction, each node can determine the utility of the feedback available to it. For example, negative feedback on a node that has handled the transaction properly has been less informative, therefore the weight assigned to the corresponding (forwarder, subject)-pair shall be decreased.

P2P data structures are dynamic systems where the peers are free to change their behavior at any time. Thus, there should be weights that allow to focus on recent transactions. In addition, single stochastic occurrences should not impact the weights. This – and the fact that it does without additional data structures – motivates the use of the exponential moving average over time to adapt the weights to new observations. Factor z with $0 \leq z \leq 1$ specifies the importance of recent information, i.e., larger values of z prefer new values. Let $a(F_{\sigma,\phi}, \theta)$ be a function to express the correlation between the

[3] In FairNet, the reputation value is the number of positive feedback items in the repository that refers to the peer in question.

transaction result θ and the set of feedback items $F_{\sigma,\phi}$ with Subject σ forwarded from Peer ϕ. The new weight w' is derived from the old weight w as shown in Equation 3.

$$w'_{\sigma,\phi} = (1 - z) \cdot w_{\sigma,\phi} + z \cdot a(F_{\sigma,\phi}, \theta) \tag{3}$$

In FairNet the transaction results and the feedback items are binary: a query is either answered or not, and the number of positive feedback items about a particular peer can only be above the threshold t or below. Let T^{pos}, T^{neg} be the sets of all successful and unsuccessful transactions, respectively. We now can develop the following correlation function $a(F_{\sigma,\phi}, \theta)$:

$$a(F_{\sigma,\phi}, \theta) = \begin{cases} 1 & \text{if } (\theta \in T^{pos} \wedge |F_{\sigma,\phi}| \geq t) \vee (\theta \in T^{neg} \wedge |F_{\sigma,\phi}| < t) \\ 0 & \text{if } (\theta \in T^{pos} \wedge |F_{\sigma,\phi}| < t) \vee (\theta \in T^{neg} \wedge |F_{\sigma,\phi}| \geq t) \end{cases} \tag{4}$$

Other reputation systems may depend on measures that express more sophisticated correlations between the transaction outcomes observed and the feedback. However, our experiments in Section 5 will show that a relatively simple solution leads to remarkably positive results already.

4 Analysis

This section provides an analysis of the measures proposed. The analysis is independent from the underlying reputation system and data structures. On the other hand, the analysis (not the experimental evaluation) is based on various assumptions. We will discuss the impact of these assumptions later in the paper. First, transaction processing takes place in rounds. In every round, each node issues one query and forwards or answers l queries on average. In addition, we assume that the system is in steady state, and the load of query processing and message forwarding is equally distributed among all nodes. Our formal analysis further assumes that uncooperative and dishonest peers are evenly distributed over the key space, i.e., there is not any cluster of neighboring peers that are unreliable and/or dishonest. Finally, we assume that the underlying reputation system handles the creation and distribution of feedback as follows: At the end of a transaction, each forwarder will be informed about its outcome. If a query remains unanswered, each forwarder generates negative feedback with the next forwarder in the sequence as feedback subject. In the other case, these peers generate positive feedback. The generated feedback will then be forwarded to all neighbors of the feedback subject.

The analysis only refers to the measures against untruthful feedback, not to the P2P data structure and the reputation system together with these measures. Hence, the analysis uses the quality of the feedback available, the frequency of successful transactions in the P2P data structure etc. as external parameters. In particular, the characteristics of P2P data structures are represented by two values: A query will not be processed successfully with probability g, and the processing of each query requires the cooperation of l peers that are not observable from the outside. In order to model the reputation system we use the following parameters: A peer forwards and handles transactions of another one only if it has at least t positive feedback items in its repository whose subject is the peer in question. The repository has the capacity to store s feedback items per

Table 1. Parameters used in the analysis

Parameters of the data structure	*Symbol*
Probability of an unsuccessful transaction	g
Number of peers that have to cooperate to process one transaction	l

Parameters of the reputation system	*Symbol*
Probability of positive feedback	p_{pos}
Number of feedback items in the repository of one peer	s
Threshold for the number of positive feedback items for a reliable peer	t

Parameters of the countermeasure	*Symbol*
Ratio of spoof feedback provided by dishonest peers	b
Smoothing factor of the Exponential Moving Average	z

subject. As a result of feedback generation in the reputation system, p_{pos} is defined to be the probability that an arbitrary feedback item issued by a honest peer on a reliable subject is positive. The values of p_{pos} then depend on the reputation system.[4]

The rate of spoof feedback b in the reputation system is the input variable of our analysis. A value of $b = 0$ denotes an honest peer that disseminates truthful feedback only, while dishonest peers forward spoof feedback at a rate of $b > 0$. Finally, the factor z specifies the smoothness factor of the Exponential Moving Average and can be customized according to the preference for newer values. Table 1 lists all parameters used in the analysis.

To examine the impact of spoof feedback, we first determine the expected average values of the weights. A dishonest peer issues spoof feedback with a rate of b and accurate feedback with a rate of $(1-b)$ that is positive with probability p_{pos}. Equation 5 gives the probability that an arbitrary feedback item generated by a dishonest node is positive.

$$p_{pos}^{dis} = \begin{cases} b \cdot 1 + (1-b) \cdot p_{pos} & \text{for spoofed positive feedback} \\ b \cdot 0 + (1-b) \cdot p_{pos} & \text{for spoofed negative feedback} \end{cases} \tag{5}$$

If at least t feedback items in a repository with s items are positive, the maintainer of the repository deems the peer reliable. Each honest peer (characterized by $b = 0$) generates positive feedback with probability p_{pos}. Therefore, it happens with a certain probability p_t that an honest peer generates less than t positive feedback items on a reliable node. In consequence, other nodes could suspect the peer to disseminate spoof feedback and reduce the weight of its feedback. The share of positive feedback items in

[4] See [11] where the value of p_{pos} is derived in one specific reputation system.

a repository follows a binomial distribution. Equation 6 now calculates the probability p_t for a repository containing less than t positive feedback items:

$$\begin{aligned} p_t &= P(\text{Number of positive feedback in the repository} < t) \\ &= \sum_{i=0}^{t-1} \binom{s}{i} \cdot (p_{pos})^i \cdot (1 - p_{pos})^{s-i} \end{aligned} \tag{6}$$

In order to determine the same probability for dishonest peers, we change the value in Equation 6 from p_{pos} to p_{pos}^{dis}, as shown in Equation 7:

$$p_t^{dis} = \sum_{i=0}^{t-1} \binom{s}{i} \cdot (p_{pos}^{dis})^i \cdot (1 - p_{pos}^{dis})^{s-i} \tag{7}$$

The expected average weight now is the probability that a transaction is not successful and that a repository contains less than t positive feedback items plus the probability of the opposite case. Equation 8 determines the weight of feedback issued by honest peers, Equation 9 does so for dishonest ones.

$$w^{hon} = p_t \cdot g + (1 - p_t) \cdot (1 - g) \tag{8}$$

$$w^{dis} = p_t^{dis} \cdot g + (1 - p_t^{dis}) \cdot (1 - g) \tag{9}$$

A participant in a P2P system is free to change its behavior at any time and with any frequency. For example, one peer might work hard to obtain a high standing in the eyes of others and try to disseminate spoof feedback afterwards. Thus, the time needed to adapt to new behavior is crucial. This time can be quantified as the number k of repository updates needed to adapt w to a new ratio of spoof feedback b. The weights are updated according to Equation 3. Therefore, k is a function of the smoothing factor z of the exponential moving average. Let a_k be the correlation $a(F_{\sigma,\phi}, \theta)$ between the transaction result and the set of feedback items at time k. We can now rewrite Equation 3 to Equation 10.

$$\begin{aligned} w_k &= z \cdot a_k + z \cdot (1 - z)^1 \cdot a_{k-1} + z \cdot (1 - z)^2 \cdot a_{k-2} + \cdots + (1 - z)^k \cdot a_0 \\ &= z \cdot a_k + a_0 \cdot (1 - z)^k + z \cdot \sum_{i=1}^{k-1} a_{k-i} \cdot (1 - z)^i \end{aligned} \tag{10}$$

The initial parameter a_0 is the value of w before the change in the behavior. To ease the calculation, the number of positive and negative feedback items in the repository and the rate of unsuccessful transactions after the change is assumed as constant, i.e., $a_1, a_2, \cdots, a_k$ are equal. Now Equation 10 is a geometric sequence and can be solved and rewritten to obtain the value of k, as shown in Equation 11.

$$k = \left\lceil \frac{\log(\frac{w_k - a_k}{a_0 - a_k})}{\log(1 - z)} \right\rceil \tag{11}$$

5 Evaluation

Having described the fundamentals of our approach independent from a concrete implementation of a P2P data structure or a reputation system, we will now evaluate the effectiveness with numeric methods and by means of experiments. Our intention with this section is to confirm that truthful feedback tends to have higher weight than spoof feedback even in worst-case settings, that the reputation system adapts quickly to changes in the behavior of nodes, and that it is effective against collaboration attacks. In order to obtain expressive results, the evaluation is based on FairNet.

Weights for Truthful and Spoof Feedback. The first question is whether spoof feedback may obtain a higher weight than honest feedback. We accomplish this by interpreting the formulae of the analysis. To do so, we use realistic values taken from a FairNet instance consisting of $10,000$ peers organized in a four-dimensional keyspace. In this setup, u is the rate of uncooperative peers. Each uncooperative peer does not forward or answer 50% of all incoming query. The rationale behind a failure rate of 50% is to analyze a 'more difficult' setting – even in the presence of spoof feedback, a completely uncooperative peer would be quickly discovered. In contrast, a failure rate of 50% is a worst-case scenario for settings with a small number of uncooperative peers u, because *any* feedback item is wrong with probability 50%. In this setting the probability for each uncooperative node to obtain positive feedback is $p_{pos} \approx 0.13$.[5] I.e., it is less than its failure probability of 50%. This is because FairNet generates more negative than positive feedback.

In settings with a large fraction u of uncooperative peers, the number of truthful positive feedback in the repositories goes against zero, as do the probabilities of successful transactions. Thus, it is easy to detect spoof positive feedback. On the other hand, if the probability of a successful transaction is about 50%, spoof feedback cannot be detected. Therefore, in a setting with few queries properly processed and a small p_{pos}, we expect the weighting factors to be smaller on spoof feedback, compared to a setting where p_{pos} is smaller than the probability of successful queries. However, we can already declare success if the weights of spoof feedback *never* are above the ones of nodes following the protocol.

We now determine the weight of a dishonest peer for a ratio of $b = 0$ to $b = 1$ spoof positive feedback generated on an uncooperative feedback subject. Figure 2 graphs the result of our analysis. The figure confirms our expectations: For values such as $u = 10\%$ or below, i.e., in our worst-case scenario with 50% successful transactions, the weight of honest feedback ($b = 0$) is only slightly larger than the one of a peer issuing spoof positive feedback only ($b = 1$). In contrast, in settings with a large number of uncooperative peers and a high rate of unsuccessful transactions, i.e., with high certainty regarding the accuracy of feedback, the weight of honest feedback is significantly larger than the weight of spoof feedback. Summing up, the analysis so far has shown that our approach assigns higher weights to honest feedback in any case. However, the difference between truthful feedback and spoof feedback might be small in worst-case settings. In

[5] Determining the probability p_{pos} in FairNet requires a complex algebraic model which we omit here for the lack of space. See [11].

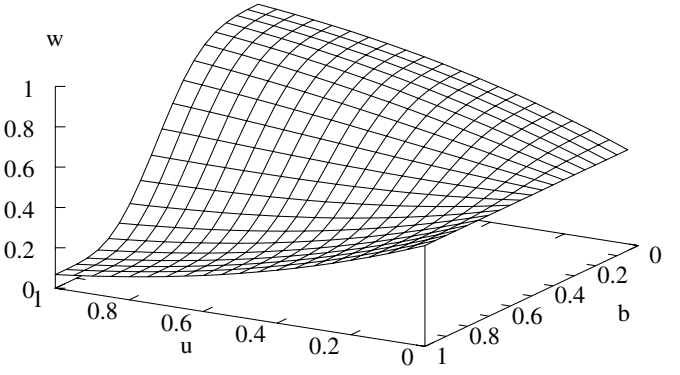


Fig. 2. Average weights for different shares of dishonest peers and spoof feedback

these settings however, truthful and spoof feedback would lead to the same decisions. We will address the applicability of our measures under worst-case conditions by means of realistic experiments later on.

Dynamicity. As a next step, we want to determine the number of updates k needed to adapt a weight w to a new ratio of spoof feedback b. The setup of our simulation is similar to the one used for Figure 2, i.e., the system consists of 10,000 peers in a four-dimensional topology. In order to have expressive results, the setup contains 50 uncooperative peers which do not handle 80% of incoming transactions.

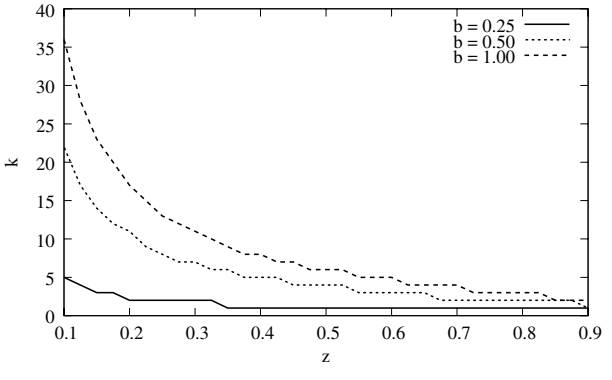


Fig. 3. Number of interactions to update the weight depending on the smoothing factor z

Equation 9 provides an estimate of w, depending on the ratio of spoof feedback b. Figure 3 graphs the number of updates k needed to decrease the weight of a truthful repository ($b = 0$) to 99% of the weight of a dishonest repository.[6] The number of

[6] Because of the exponential moving average, the weights asymptotically converge to the expected value. Hence, we are satisfied with a conformance of at least 99%.

updates is shown in comparison to the smoothing factor z and for three different ratios of spoof feedback. The exponential moving average replaces old information at a constant rate z. This explains why it requires more updates k to adapt to a repository with $b = 1$ in comparison to one with $b = 0.25$, as shown in Figure 3. However, in P2P data structures a peer usually interacts with its neighbors frequently. The example calculation of Section 2 has shown that it requires around 10 interactions between neighboring peers to forward one query from the issuer to the peer that can actually answer it in a setting with 10,000 peers and a four-dimensional key space. Thus, even though the value of $k = 35$ at $z = 0.1$ and $b = 1$ might seem to be large, it actually tells us that the weights are adjusted within less than four rounds. Larger smoothing factors shorten this period of time even more.

Robustness Against Collaboration Attacks. The last question that we have to address is: How useful is our approach in the presence of peers running a collaboration attack? In particular, does it pay off for a group of dishonest peers to 'boost' the reputation of an uncooperative peer by issuing and forwarding spoof feedback? A series of experiments addresses these questions. Our experimental setup consists of 1,000 peers in a setup where each peer has 26 neighbors. 50 uncooperative peers ignore 50% of the incoming queries. x dishonest peers surround each uncooperative peer. These x peers try to push the standing of the uncooperative one by disseminating spoof positive feedback items at a rate b. In other words, they generate honest feedback at a rate $(1 - b)$. In a series of 625 experiments, we varied the number of attackers from $x = 0$ to 25 and changed the ratio of spoof feedback from $b = 0$ to 1. As outlined in Section 2, only the neighbors can observe the behavior and generate feedback on a peer. Thus, the experiments with $x = 25$ identify extreme settings where dishonest nodes almost completely surround the uncooperative peer.

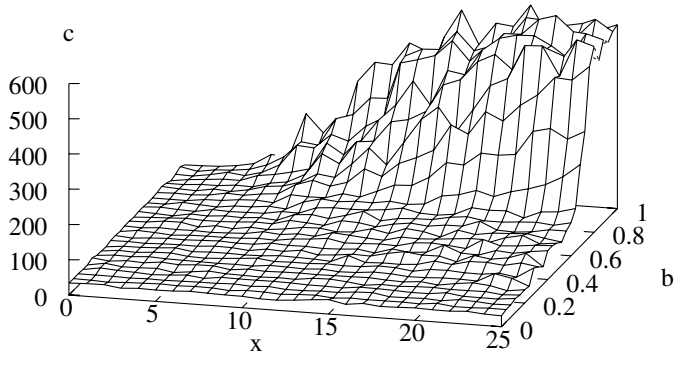


Fig. 4. Unhandled transactions in FairNet

Each experiment consists of 200,000 queries. The numbers are taken after an initialization period that allows the reputation system to reach a steady state. We measured the number c of queries dropped per round by all uncooperative peers, i.e., the total number of unanswered queries caused by 50 uncooperative peers, 'supported' by up to

950 dishonest peers issuing spoof positive feedback. Figure 4 shows the results of our experiments without our measures. It indicates that uncooperative peers drop a small fraction of queries even without the involvement of any dishonest peers ($x = 0$). This is because our experimental setup does not include data replication, and queries referring to keys in the zones of uncooperative peers are answered with a probability of 50% only. Except for this phenomenon, the reputation system does well without countermeasures against spoof feedback, even in the presence of dishonest peers. Only collaboration attacks where more than one third of the neighbors of an uncooperative node issue significantly more than 70% spoof feedback increase the number of unanswered queries. On the other hand, there have already been distributed attacks on the Internet with thousands of 'zombie computers' compromised by viruses and directed by a single attacker. Similar attacks on P2P data structures are conceivable as well. Thus, measures against spoof feedback are still necessary.

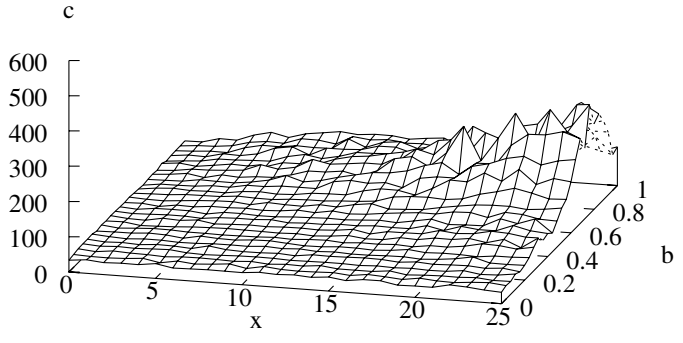


Fig. 5. Unhandled transactions in FairNet with weights

But how do the experimental results change with our approach? To investigate this, we replayed our series of experiments with our measures activated. Figure 5 contains the results. The figure shows two findings: The impact of the attacks has been largely reduced, and the number of collaborators required for a successful attack has increased considerably. At least 17 attackers have to disseminate more than 80% spoof positive feedback to increase the number of unanswered queries significantly. Moreover, the maximal number of messages lost resulting from $x = 25$ absolutely dishonest collaborators is approximately one third of the one observed in the experimental setup without our approach. Other experiments with spoof negative feedback on cooperative feedback subjects (omitted here for lack of space) yield similar results. Summing up, repository weights are an effective countermeasure against collaboration attacks in reputation systems for P2P data structures.

6 Related Work

This section reviews approaches that are related to our measure against spoof feedback in reputation systems for P2P data structures. The section starts with a (very) short

outline of related P2P data structures, followed by a review of P2P reputation systems. An overview on truthtelling mechanisms that are not specific to P2P systems concludes.

P2P Data Structures. P2P data structures address a core issue in data management: administering of huge sets of $(key, value)$-pairs under a high rate of parallel transactions. The various approaches [7,8,9,10] differ primarily with regard to *contact selection* and *path selection*, i.e., which are the peers a node can communicate with and forward messages to. The topology of the key space is closely related to contact selection and path selection. Common topologies include hypercubes (e.g., CAN [7]), rings (Chord [10]), virtual search trees (P-Grid [8]), and butterfly networks (Viceroy [9]); see [6] for an analysis of the impact of the topology on the characteristics of the data structures.

P2P Reputation Systems. All of these approaches assume that nodes readily follow the protocol. We think that this is not realistic. Reputation systems allow the peers to deal with unreliable nodes by collecting, distributing and aggregating feedback on the behavior of the participants in the past. One of the first reputation systems based on P2P data structures is [15]. The approach is based on complaints, i.e., negative feedback. Each peer stores the feedback it has generated in a global repository that is accessible by all peers. A peer assigned with more negative feedback than the global average is deemed unreliable. As a measure against spoof feedback, the approach proposes to check not only the number of complaints on the peer in question, but also the reputation of the peers which issued the complaints. But this does not help against a compromised global repository and comes with a large overhead. *EigenTrust* [16] is an approach to reputation systems that is based on a distributed eigenvector computation. The approach uses a P2P data structure to store a global trust vector. For each pair of peers, the trust vector contains a normalized reputation value, based on the number of satisfying and unsatisfying transactions. In order to avoid spoof feedback the reputation value of each peer is recursively weighted with the reputation of its 'observers'. However, cooperative peers are not forced to provide truthful feedback in settings such as ours. Another assumption that does not hold in P2P data structures is that an initial set of users is known to be trustworthy. *PeerTrust* [17] derives trust values from the satisfaction earned by each transaction, the credibility of the participating peers, the context of the transactions and community-specific issues. Similar to the other approaches, the trust model of PeerTrust depends on a secure, global data structure that stores feedback. Spoof feedback is addressed with a credibility factor derived from the assumptions that uncooperative peers tend to disseminate spoof feedback and cooperative peers usually issue truthful feedback. These assumptions may fail in the presence of groups of colluding peers which strive for 'strategic' goals, e.g., discrediting other nodes. A comparison of other P2P-based reputation systems is shown in [14].

It is challenging to secure global data structures against dishonest peers. A peer which wants to influence the reputation system could try to insert spoof feedback, tamper with feedback items it is supposed to forward and manipulate feedback in its local zone. FairNet [11], our reputation system for P2P data structures, avoids these vulnerabilities by introducing mechanisms that work on local data structures. In particular, the peers maintain local repositories and exchange feedback with every message that is sent out to another peer. With local repositories, an attacker that wants to modify a certain

reputation value is forced to compromise the repositories of many peers. However, local repositories without further countermeasures may still fall prey to spoof feedback.

Truthtelling Mechanisms. In addition to mechanisms designed for certain reputation systems, others have investigated approaches to incentivize truthtelling. The approaches do not depend on a specific implementation. *CONFESS* [3] aims at eliciting truthful feedback in buyer-seller situations. The idea is that buyers who appear repeatedly will build a reputation for truthtelling in equilibrium. The authors formally prove the effectiveness of the mechanism under the given assumptions. However, their solution is not readily applicable to our setting, for two reasons. First, CONFESS requires a central instance that all participants deem trustworthy. This is different from P2P architectures. Another issue is that uncertainty/subjectivism is not part of the model, at least currently: If a seller behaves cooperatively, the buyer will always notice this. If the seller does not, the buyer will notice this as well. Our approach in turn does without this assumption. The rate of such errors is an endogenous parameter of our approach.

Other proposals, e.g., *Bayesian Truth Serum* [5] and *Peer-Prediction* [4], pursue a different (i.e., not reputation-based) approach to the same problem, albeit in a slightly different setting. They compare the probability distribution of truthful answers to other probability distributions (the one of the answers of all participants in the case of Peer-Prediction, and the one predicted to be the distribution of the answers of all participants in the case of the Bayesian Truth Serum). This comparison allows to maximize the expected payoff of truthful answers, as formally shown in the respective publications. Unlike CONFESS, it does so without requiring repeated interactions. However, both approaches are not applicable to our setting as well. First, Peer-Prediction requires that the probability distributions of answers (of truthful feedback, to translate this to our setting) is known; Bayesian Truth Serum in turn requires that peers come up with an estimate of this distribution. Another issue is that, in spite of the name of one of the approaches, they are not Peer-to-Peer. More specifically, it is unclear how to implement them in an environment consisting of only the peers (and no other instances that could act as coordinators etc.). Finally, to the best of our knowledge, there have only been few experiments evaluating these approaches [18].

7 Discussion

The experiments presented so far have acknowledged that our countermeasure can be used with CAN and FairNet. But it remains to be discussed if our countermeasure are applicable to other reputation systems and application scenarios as mentioned in Section 6. Unlike many other approaches, FairNet does not depend on global data structures. Instead, the peers manage and exchange feedback locally with each interaction. However, our approach does not depend on the location where the feedback is stored. Instead, the peers just assign weights to the nodes which forward the feedback, according to the correlation between that feedback and the transaction outcomes observed. Thus, our approach is applicable to each reputation system where nodes exchange feedback items, e.g., [15,19] or (with some changes in the architecture) [17].

Our approach relies on mechanisms to detect spoof feedback with little resource consumption. The downside is that the approach requires several interactions before

adapting to the behavior of a node, according to our evaluation. Thus, our approach requires reputation systems characterized by a high throughput of feedback. However, this is generally the case in systems such as P2P data structures, and it is an attribute of many fields of application, e.g., semantic web or distributed search engines.

The experiment on collaboration attacks has indicated a significant improvement of the reliability in the presence of many peers issuing spoof feedback at a high rate. But the experiment has also shown that the countermeasure cannot prevent any transaction from being forwarded to unreliable peers. Therefore, our approach is only applicable in settings with many 'inexpensive' transactions where a few messages may get lost.

8 Conclusions

Spoof feedback is an important issue in any kind of reputation systems. Dishonest participants may wish to discredit others or try to take advantages from disseminating spoof feedback. The problem becomes even more difficult in distributed reputation systems for P2P data structures. Such settings are characterized by a high throughput of feedback and complex collaboration models with peers that cannot be observed from one instance. In this paper we describe the requirements that a reputation system for P2P data structures must fulfill and propose our new approach for truthful feedback. The approach takes advantage of the fact that each peer can observe the utility of feedback obtained from others after having observed the outcome of a transaction. The peers derive weighing factors of (feedback forwarder, feedback subject)-pairs.

We evaluate our approach with an analytic model and by means of extensive experiments. The analysis confirms that the differentiation between useful and less useful feedback is effective, irrespective of the particular implementation of the reputation system. The experimental evaluation demonstrates the applicability of our approach in realistic settings. It shows a significant reduction of the impact of collusion attacks where more than 90% of the peers issue spoof feedback.

References

1. Resnick, P., Kuwabara, K., Zeckhauser, R., Friedman, E.: Reputation Systems. Communications of the ACM (CACM) **43** (2000)
2. Khopkar, T., Li, X., Resnick, P.: Self-Selection, Slipping, Salvaging, Slacking, and Stoning: The Impacts of Negative Feedback at eBay. In: Proceedings of the 6th ACM Conference on Electronic Commerce (EC'05). (2005) 223–231
3. Jurca, R., Faltings, B.: CONFESS: Eliciting Honest Feedback without Independent Verification Authorities. Proceedings of the 6th International Workshop on Agent Mediated Electronic Commerce (AMEC'04) (2004)
4. Miller, N., Resnick, P., Zeckhauser, R.: Eliciting Informative Feedback: The Peer Prediction Method. Management Science **51** (2005) 1359–1373
5. Prelec, D.: A Bayesian Truth Serum for Subjective Data. Science **306** (2004) 462–466
6. Gummadi, K., Gummadi, R., Gribble, S.D., Ratnasamy, S., Shenker, S., Stoica, I.: The Impact of DHT Routing Geometry on Resilience and Proximity. In: Proceedings of the ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM'03). (2003)

7. Ratnasamy, S., Francis, P., Handley, M., Karp, R., Shenker, S.: A Scalable Content-Addressable Network. In: Proceedings of the ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM'01). (2001)
8. Aberer, K.: P-Grid: A Self-Organizing Access Structure for P2P Information Systems. In: Proceedings of the 9th International Conference on Cooperative Information Systems (CoopIS'01). (2001) 179–194
9. Malkhi, D., Naor, M., Ratajczak, D.: Viceroy: A Scalable and Dynamic Emulation of the Butterfly. In: Proceedings of the 21th ACM Symposium on Principles of Distributed Computing (PODC'02). (2002) 183–192
10. Stoica, I., Morris, R., Liben-Nowell, D., Karger, D.R., Kaashoek, M.F., Dabek, F., Balakrishnan, H.: Chord: A Scalable Peer-To-Peer Lookup Service for Internet Applications. In: Proceedings of the ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM'01). (2001)
11. Böhm, K., Buchmann, E.: Free Riding-Aware Forwarding in Content-Addressable Networks. International Journal on Very Large Data Bases (VLDB) (2006)
12. Back, A.: Hashcash - A Denial of Service Counter-Measure. http://hashcash.org (2002)
13. Jakobsson, M., Juels, A.: Proofs of Work and Bread Pudding Protocols. In: Proceedings of the 4th International Conference on Communications and Multimedia Security (CMS'99). (1999)
14. Dewan, P., Dasgupta, P.: Securing P2P Networks Using Peer Reputations: Is There a Silver Bullet? In: Proceedings of the 2nd IEEE Consumer Communications and Networking Conference (CCNC'05). (2005)
15. Aberer, K., Despotovic, Z.: Managing Trust in a Peer-2-Peer Information System. In: Proceedings of the 10th International Conference on Information and Knowledge Management (CIKM'01). (2001)
16. Garcia-Molina, H., Schlosser, M.T., Kamvar, S.D.: The EigenTrust Algorithm for Reputation Management in P2P Networks. Proceedings of the 12th International World Wide Web Conference (WWW'03) (2003)
17. Xiong, L., Liu, L.: PeerTrust: Supporting Reputation-Based Trust for Peer-to-Peer Electronic Communities. IEEE Transactions on Knowledge and Data Engineering (TKDE) **16** (2004)
18. Prelec, D., Weaver, R.G.: Truthful Answers are Surprisingly Common:Experimental Tests of the Bayesian Truth Serum. In: Proceedings of the Conference on Econometrics and Experimental Economics (CEEE'06). (2006)
19. Cornelli, F., Damiani, E., di Vimercati, S.D.C., Paraboschi, S., Samarati, P.: Choosing Reputable Servents in a P2P Network. In: Proceedings of the 11th International World Wide Web Conference (WWW'02). (2002) 376–386

Efficient Peer-to-Peer Belief Propagation*

Roman Schmidt and Karl Aberer

School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne (EPFL)
CH-1015 Lausanne, Switzerland

Abstract. In this paper, we will present an efficient approach for distributed inference. We use belief propagation's message-passing algorithm on top of a DHT storing a Bayesian network. Nodes in the DHT run a variant of the spring relaxation algorithm to redistribute the Bayesian network among them. Thereafter correlated data is stored close to each other reducing the message cost for inference. We simulated our approach in Matlab and show the message reduction and the achieved load balance for random, tree-shaped, and scale-free Bayesian networks of different sizes.

As possible application, we envision a distributed software knowledge base maintaining encountered software bugs under users' system configurations together with possible solutions for other users having similar problems. Users would not only be able to repair their system but also to foresee possible problems if they would install software updates or new applications.

1 Introduction

Peer-to-peer systems currently share local information by pairwise interactions in a cooperative way. The most popular application to date is file-sharing such as Gnutella and BitTorrent providing search functionality respectively efficient content distribution. Shared data is usually file-based and files are not correlated with each other, i.e., it is sufficient to find a desired file and to be able to download it. More sophisticated applications rely on correlated data probably spread out among several nodes and downloading each part for local processing can be too expensive. Another solution is to perform distributed inference directly in the network so that data remains at providing nodes and only small messages to process the inference are exchanged.

Distributed inference is already applied for various applications in other networks such as sensor networks [1] where network limitations are probably more

* The work presented in this paper was supported (in part) by the National Competence Center in Research on Mobile Information and Communication Systems (NCCR-MICS), a center supported by the Swiss National Science Foundation under grant number 5005-67322 and was (partly) carried out in the framework of the EPFL Center for Global Computing and supported by the Swiss National Funding Agency OFES as part of the European project NEPOMUK No FP6-027705.

R. Meersman, Z. Tari et al. (Eds.): OTM 2006, LNCS 4275, pp. 516–532, 2006.

obvious. We envision applications on top of a peer-to-peer system relying on knowledge provided by nodes and used to solve inference problems. A practical example is a distributed knowledge base for software bugs observed by users. Currently, bug reports are submitted on user acceptance to a central knowledge base for further processing. We assume that most of the bug reports are not submitted because users are afraid to reveal their identity. In our distributed scenario, the reports are inserted into a peer-to-peer system concealing the users identity. Distributed inference is then used to suggest solutions for occurred errors as known from central knowledge bases.

Belief propagation [2] enables distributed inference by a simple message-passing algorithm between nodes in a Bayesian network modeling correlations between variables. A node can represent any kind of variable, be it an observed measurement, a parameter, a latent variable, or a hypothesis. Belief propagation was first successfully applied in the domain of error correcting codes (Turbo Codes [3]), speech recognition, image processing and medical diagnosis. Recently, it was used in peer-to-peer systems in the context of content distribution [4] and in sensor networks [5]. The simplicity of the message-passing algorithm holds the risk of being not scalable towards large-scale networks because many small messages have to be sent between nodes. Approaches to reduce communication costs such as Generalized Belief Propagation [6] cluster nodes and build a hierarchy based on common variables of clusters. The message reduction comes with the drawback that the size of sent messages increases exponentially ($number\ of\ states^{nodes\ in\ the\ cluster}$) because the exchanged messages now contain the joint probabilities of all nodes and states in the cluster. What remains unsolved is how nodes are clustered in a distributed way requiring no global knowledge and coordination so that the communication costs are minimized.

In this paper, we will present a decentralized algorithm to cluster variables at nodes to reduce the number of physical messages sent over the network by not increasing message sizes. The overall number of messages to run Pearl's belief propagation algorithm remains the same but most of them are sent node-internally which does not induce any bandwidth nor latency costs. Our clustering algorithm is based on the spring relaxation technique used for example in peer-to-peer systems for virtual coordinate systems [7] and for path optimization in stream-based overlays [8] to find minimal energy configurations. In our case, we try to find the minimal configuration for variables stored on nodes organized in an P-Grid [9] overlay network. P-Grid provides us a distributed index of the Bayesian network and efficient lookup mechanisms.

In the following, we will first explain briefly the background and the basis of our approach, belief propagation in Section 2 and P-Grid in Section 3. Afterwards, we will present peer-to-peer blief propagation in Section 4 before we evaluate our approach in Section 5. The paper discusses related work in Section 6 and future work in Section 7. We conclude in Section 8.

2 Belief Propagation

Pearl's belief propagation [2], also known as the sum-product algorithm, is an iterative algorithm for computing marginal probabilities, "beliefs" about possible diagnoses, of nodes on a probabilistic graphical model such as Bayesian networks. A Bayesian network is an directed acyclic graph of nodes representing variables and edges representing dependence relations among the variables. If there is an edge from node A to node B, then node B's state depends on node A's state. This is specified by a conditional probability distribution for node B, conditioned on the state of node A. A Bayesian network is a representation of the joint distribution over all the variables represented by nodes in the graph. We assume that the joint probability distribution factors into a product of terms involving node pairs and single nodes. These factors are called edge potentials $\psi_{ij}(x_i, x_j)$ and local potentials $\phi_i(x_i)$. Evidence nodes are nodes with a known value. A node can represent any kind of variable, e.g., an observed measurement, a parameter, a latent variable, or a hypothesis. For example, consider the simple Bayesian network in Figure 1 consisting of 3 variables OS1, Driver1 and App1. The dependencies are as follows: if the hardware driver Driver1 is installed on the operating system OS1, the application App1 is likely to run smoothly with 90% probability. If the driver is missing, the application runs only to 40% and if OS1 is not installed, then the application does not run at all independent of the driver. If it is known that OS1 is installed, then its probability would be set to 1 and the probabilities for App1 to run would only depend on Driver1 thereafter.

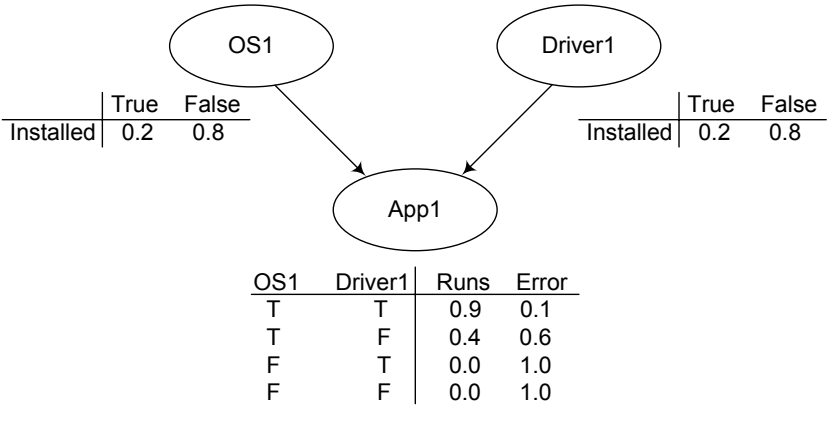


Fig. 1. Bayesian network example

The belief propagation algorithm is provably efficient on trees and experiments demonstrate its applicability to arbitrary network topologies using loopy belief propagation for loopy networks [10], which we will present in the following. The algorithm is currently used with success in numerous applications

including low-density parity-check codes, turbo codes, free energy approximation, and computer vision.

2.1 The Message Passing Algorithm

The algorithm passes messages across the edges in the graphical model, i.e., in each iteration, a node sends a message to an adjacent node if it has received messages from all of its other adjacent nodes at the previous iteration. In the first iteration, nodes send an initial message, usually set to 1, to all adjacent nodes. In subsequent iterations, messages passed from node x_i to node x_j are updated using the following rule:

$$m_{ij}(x_j) = \sum_{x_i} \phi_i(x_i)\psi_{ij}(x_i, x_j) \prod_{k \neq j} m_{ki}(x_i)$$

where $\phi_i(x_i)$ are the local potentials of node x_i and $\psi_{ij}(x_i, x_j)$ are the edge potentials. The product of messages excludes the message received in the previous iteration from node j, the node we are passing the message to. The messages $m_{ij}(x_j)$ and the local potentials $\phi_i(x_i)$ are vectors whose length corresponds to the number of states a node x_i can be in. The edge potentials $\psi_{ij}(x_i, x_j)$ are N x M matrices where N is the number of states node x_j can be in and M is the number of states for node x_i. Therefore, the message size of the belief propagation algorithm grows exponentially with the number of states of nodes.

Finally, the marginal probabilities of nodes, called the beliefs, can be computed by multiplying all received messages by the local potentials:

$$b_i(x_i) = \alpha\phi_i(x_i) \prod_{k} m_{ki}(x_i)$$

The beliefs are normalized by α to avoid numerical underflow. The algorithm converges if none of the beliefs in successive iterations changes by more than a small threshold. For singly connected graphs, it is proven [2] that beliefs at nodes converge to the marginal probability at that node, which is:

$$b_i(x_i) = \alpha \sum_{x_j / x_i} p(x) = p_i(x_i)$$

In networks with loops, evidence is counted multiple times. As all evidence is double counted in equal amounts, Pearl's belief propagation also provides good approximations of the marginal probabilities in loopy networks.

3 The P-Grid Overlay

The approach presented in this paper uses the P-Grid [9] distributed hash table (DHT). We assume that the reader is familiar with the general concepts of DHTs and will thus only address the specific and relevant properties of P-Grid.

In P-Grid peers refer to a common underlying binary trie structure to organize their routing tables. Data keys are computed using an order-preserving hash function to generate keys. Without constraining general applicability binary keys are used in P-Grid. Each peer constructs its routing table such that it holds peers with exponentially increasing distance in the key space from its own position. This technique basically builds a small-world graph [11], which enables search in $O(\log N)$ steps. Each peer $p \in P$ is associated with a leaf of the binary trie, i.e., a key space partition, which corresponds to a binary string $\pi(p) \in \Pi$ called the peer's *path*. For search, the peer stores for each prefix $\pi(p,l)$ of $\pi(p)$ of length l a set of references $\rho(p,l)$ to peers q with property $\pi(p,l) = \pi(q,l)$, where $\overline{\pi}$ is the binary string π with the last bit inverted. This means that at each level of the trie the peer has references to some other peers that do not pertain to the peer's subtrie at that level which enables the implementation of prefix routing.

Each peer stores a set of data items $\delta(p)$. For $d \in \delta(p)$ $key(d)$ has $\pi(p)$ as prefix but it is not excluded that temporarily also other data items are stored at a peer, that is, the set $\delta(p, \pi(p))$ of data items whose key matches $\pi(p)$ can be a proper subset of $\delta(p)$. Moreover, for fault-tolerance, query load-balancing, and hot-spot handling, multiple peers are associated with the same key-space partition (structural replication), and peers additionally also maintain multiple references $\sigma(p)$ to peers with the same path (data replication).

Figure 2 shows a simple example of a P-Grid tree consisting of 6 peers responsible for 4 partitions, e.g., peer F's path is 00 leading to two entries in its routing table: peer E with path 11 at the first level and peer B with path 01 at the second level. Further, peer F is responsible for all data with key prefix 00. A search initiated at peer F for key 100 would first be forwarded to peer E because it is the only entry in F's routing table at level 1*. As peer E is responsible for 11 and not for the key 100, peer E further forwards the query to peer D, which can finally answer the query.

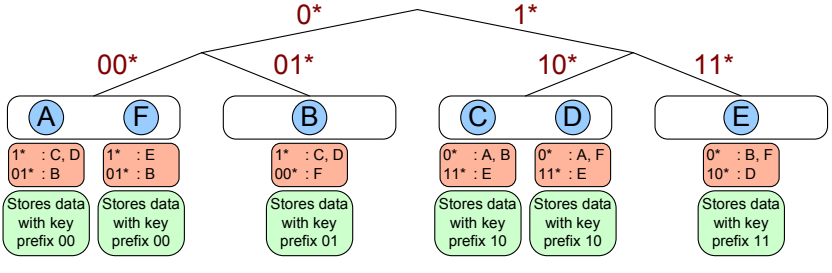


Fig. 2. P-Grid overlay network

4 Peer-to-Peer Belief Propagation

So far, we presented two independent approaches, on the one hand a distributed inference algorithm based on a simple message-passing algorithm and on the

other hand an overlay system to store and retrieve data. At first sight, those two systems have not much in common but we will show in this section that both can benefit from each other. Our focus lies thereby on P-Grids ability to improve the scalability of belief propagation as the application of belief propagation to improve P-Grid's load balancing is already shown in [12].

4.1 Distributed Knowledge Base Scenario

To motivate our idea of providing a large-scale peer-to-peer inference system, we will describe in this section our distributed knowledge base scenario for software dependencies. Examples for centralized knowledge bases are the one from Microsoft [13] and Mozilla [14] providing comprehensive information about their products and support for encountered bugs. Both use a bug report and tracking system, for example BugZilla [15] from Mozilla, to collect bug reports from users with their permission. In our opinion and from our personal experience, we assume that most users do not permit submitting bug reports to a central authority such as Microsoft or Mozilla because they do not want to reveal their identity and system configuration. A distributed solution would allow users to conceal their identity because bug reports are inserted and stored at various nodes of the peer-to-peer system making it more difficult to track user identities. Therefore, we hope to be able to collect more reports from users with our decentralized solution leading to a more comprehensive knowledge base.

A knowledge base stores data together with their dependencies usually for the purpose of having automated deductive reasoning applied to them. Belief networks are one way to define those dependencies and belief propagation is an appropriate probabilistic reasoning method. In our scenario, any kind of software such as operating systems, device drivers and applications are nodes in our Bayesian network and their dependencies and distributions are learned from bug reports. A simplified bug report could look like:

```
IF
    OS = ['Linux', '2.4.12.18'] AND
    PACKAGE = ['MySQL-server', '4.0.20-0'] AND
    PACKAGE = ['MyODBC', '2.50.39-18.1']
    ...
THEN
    ERROR = 'MySQL'
END
```

A bug report starts with the used operating system, in this case with Linux and the kernel version 2.4.12.18, followed by a list of installed packages on the system. The second part consists of the affected application (MySQL) causing the error. The bug report enables us already to learn that the given system configuration leads to problems for the MySQL application. Therefore, each bug report allows us already to create a small dependency graph, i.e., a Bayesian network, but we are still not able to identify responsible packages causing the

problem. Therefore, we need a larger number of bug reports with probably varying system configurations to identify the strength of package dependencies, e.g., if a version of a package always occurs in a bug report for an application, it is very likely to cause the problem. A solution proposal for our example bug could be to install a different version of MyODBC as this version is listed in many bug reports for the MySQL application.

4.2 The Inference Architecture

Our idea of providing a generic distributed inference system is based on two fundamental design decisions: (i) no central coordination of the variables in the system and their dependencies; (ii) no global knowledge and only pair-wise interactions between nodes. Both requirements are satisfied by the P-Grid overlay infrastructure and Bayesian networks and belief propagation. P-Grid is first used to maintain the Bayesian network by indexing all variables in the system and all dependencies between them. In our scenario, variables would be software components with their version number, and their dependencies would be derived from bug reports at their insertion. At this stage, nodes are able to derive small dependency graphs from the bug reports they store and all the variables they maintain locally. Those dependencies are already represented by Bayesian network and have now be connected with each other. Learning a Bayesian network structure and probabilities from distributed data is studied in various papers [16,17,18]. The bug reports itself are also stored in P-Grid together with solutions for bugs provided by users. Therefore, users have the possibility to help each other with solutions and if no solution exists, inference can help to restrict the cause of error.

So far, we have a system storing a Bayesian network derived from bug reports. Belief propagation requires multiple message-passing iterations between all nodes of the Bayesian network which are currently spread over physical P-Grid nodes. On a global scale, this can lead to scalability problems for our system because messages would be sent around the globe multiple times. To tackle this problem, we uncouple variable values, the local potentials, from the P-Grid index and allow them to be stored at different physical P-Grid nodes to improve the efficiency of belief propagation. The current location of a variable's local potential is stored with the variable's index entry. The problem remains how those local potentials are stored close to each other, in the best case even on the same physical P-Grid node, without central coordination and knowledge. Our proposed solution is based on the spring relaxation technique and presented in more details in the following section.

4.3 The Relaxation Algorithm

In this section we describe the developed relaxation algorithm based on the spring relaxation technique. In our case, Bayesian variables are connected by springs and the Bayesian network forms a spring network which has to be relaxed, i.e., the network has to be in a state requiring least possible energy. The

energy a spring requires is directly proportional to the distance between the two P-Grid nodes the Bayesian variables are stored at. The spring between two variables remaining at the same node requires no energy. Therefore, the optimal solution of the spring relaxation algorithm would be to place all variables at one node. This is of course not desirable because peer-to-peer systems are based on the idea of load sharing which is in contradiction with the optimal solution mentioned before. Thus, the spring relaxation algorithm also has to consider load balancing of variables among participating nodes. P-Grid provides already heuristic statistics about the current load of each level of the trie represented by a peer's routing table. These statistics are required by P-Grid itself to provide load-balancing of stored index information and are used in the following for our approach too. The statistics are based on periodic interactions with random peers of the routing table to sample the current load distribution. The periodic sampling enables peers to estimate the current load of a routing table level and the global average load.

The developed algorithm used to relax the Bayesian network is shown in Algorithm 1. The algorithm is executed by each node iteratively till no improvement is achieved anymore or a maximum number of iterations is reached. The following list provides an overview of the used variables in the algorithm:

- **localVars:** list of variables the local node maintains
- **avgLoad:** local estimate of the global average load
- **currentLoad:** the current load of the local node
- **routingTable:** the routing table of the local node
- **routingTable.levels:** the number of levels in the local routing table
- **candidate(j).tension(i):** the tension at level i for candidate variable j
- **candidate(j).tension:** all tensions at all levels for candidate variable j

First, in line 1 to 4, each node checks if it has "free" variables it can move to other nodes or not. Currently, nodes are allowed to move variables as long as they have more than $avgLoad/2$ variables. P-Grid obtains an estimate for the current average load in the system but the accuracy of this estimate is not crucial for the algorithm. In line 5, nodes determine those local variables which have a tension to other nodes remaining at the same level of the local routing table leading to one tension at one level. Ideally, variables have a tension to only one node and not to different nodes at the same level. If the local node can move variables and it found such unidirectional variables, it moves them directly to the corresponding level or node (line 6 to 10). Moving a variable always requires only one message between the two involved peers.

A node can try to balance the load in the system if it maintains above average many variables. It therefore uses all non-unidirectional variables, i.e., variables which have tensions at multiple levels (line 12 and 13). Next, the node tries to balance each level of its routing table, starting with the highest level, i.e., its closest neighbors (line 14). Starting with the closest neighbors allows nodes to balance load first locally before they try to balance load on peers further away from them, i.e., on peers stored in lower levels. If a level is underpopulated (line

Algorithm 1. The spring relaxation algorithm

```
 1: freeVars = length(localVars) − avgLoad/2;
 2: if (freeVars <= 0) then
 3:    return;
 4: end if
 5: undirVars = variables having a tension only at one level;
 6: while ((freeVars > 0) AND (length(unidirVars) > 0)) do
 7:    move variable to a peer from the level with the tension;
 8:    removeFirst(unidirVars);
 9:    freeVars = freeVars − 1;
10: end while
11:
12: multidirVars = variables having tensions to multiple levels;
13: while ((currentLoad > avgLoad) AND (length(multidirVars) > 0)) do
14:    for i = routingTable.levels to 1 do
15:       if (level i is underpopulated) then
16:          candidates = variables having a tension at level i;
17:          for j = 1 to length(candidates) do
18:             if (candidate(j).tension(i) >= max(candidate(j).tension)) then
19:                move variable to a peer from level i;
20:                remove(multidirVars, candidate(j));
21:                currentLoad = currentLoad − 1;
22:                if (currentLoad <= avgLoad) then
23:                   break;
24:                end if
25:             end if
26:          end for
27:       end if
28:    end for
29: end while
```

15), i.e., a level maintains below average many variables, then the node first selects candidate variables out of its local variables (line 16). Candidates are all variables which have a tension at the current level. Next, starting from line 17, the node checks if the tension at the current level for the candidate variable is the strongest tension the variable has considering all levels. This ensures that variables are moved to levels with their strongest tension. This process continues as long as candidates are available and the node has enough variables to move.

5 Evaluation

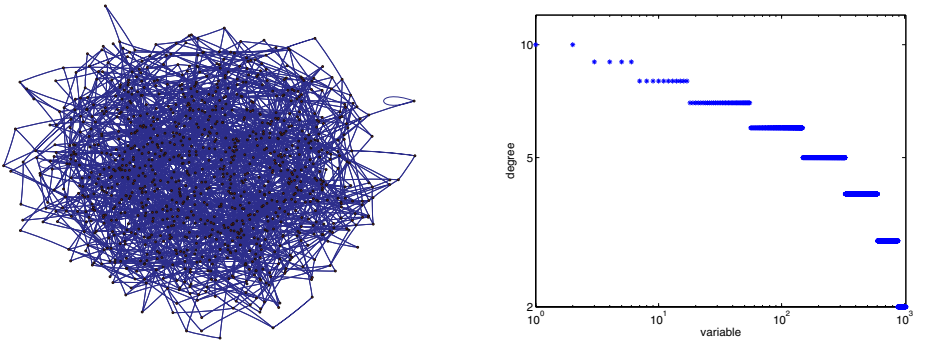
The algorithm presented in the previous section was implemented in Matlab and evaluated with diverse networks. We present results for random networks, binary trees and scale-free networks with up to 2048 variables in the Baysian network and 512 nodes in the P-Grid network. Considering our scenario we have in mind for our system, tree-based belief networks and scale-free networks

are the most realistic network topologies. The network size and the number of variables is difficult to estimate but the evaluation shows that our approach scales well even though no proof can be given so far. All experiments were repeated 10 times and the figures show the average of those 10 repetitions with their standard deviation. Each time a new belief network was created and variables were assigned randomly to nodes.

5.1 Network Topologies

We briefly describe some properties of the network topologies we used for our evaluation. The networks were visualized with the Pajek tool [19] using the 2D Fruchterman Reingold layout for random networks and the Kamada-Kawai layout for the others. Additionally, we show the node degree distribution by sorting nodes according to their node degree and plotting their degree in log-log scale.

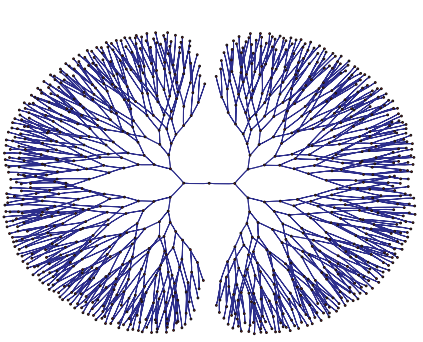
Random Networks. We constructed random networks by adding for each node degree/2 edges to other nodes with equal probability to reach the desired average node degree. Figure 3 shows a network of 1024 nodes with an average node degree of 4, nodes have between 2 and 10 edges. The degree distribution indicates that most of the nodes have a degree around the average.



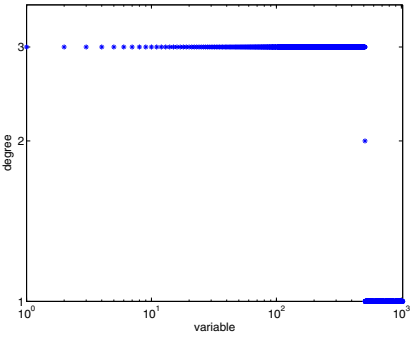
(a) Network visualization (b) Node degree distribution

Fig. 3. A random network: 1024 nodes with average node degree 4

Binary Trees. The second used topology is a binary tree with each node having exactly two children excluding leaf nodes. Each node has exactly one parent excluding the root of the tree. Therefore, the node degree varies between 1 and 3 with an average around 2. Figure 4 shows a binary tree with 1023 nodes. The degree distribution shows the leave nodes (half of the nodes) at the bottom with 1 edge, the root with 2 edges in the middle and the intermediate nodes with 3 edges at the top.



(a) Network visualization



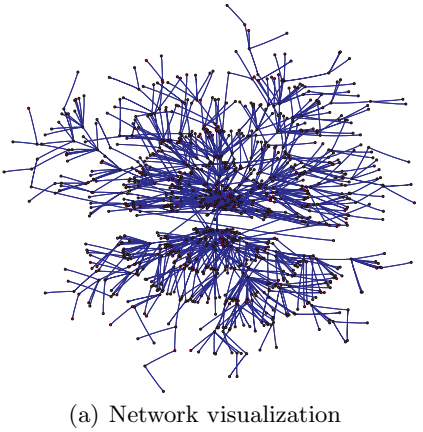
(b) Node degree distribution

Fig. 4. A binary tree: 1023 nodes

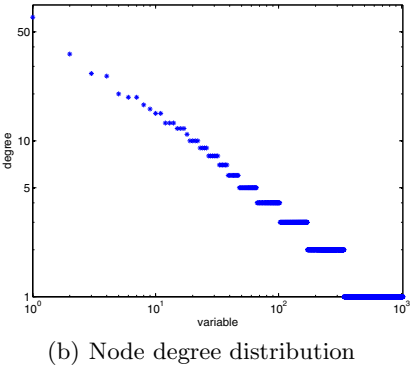
Scale-Free Networks. The last used network topology is a scale-free network with the property that the number of links k originating from a given node exhibits a power law distribution $\mathbb{P}(k) \sim k^{-gamma}$. The network is constructed by progressively adding nodes to an existing network and introducing links to existing nodes with preferential attachment so that the probability of linking to a given node i is proportional to the number of existing links k_i that that node has, i.e.,

$$\mathbb{P}(\textit{linking to node } i) \sim \frac{k_i}{\sum_j k_j}$$

Scale-free networks occur in many areas of science and engineering, e.g., including the topology of web pages (where the nodes are individual web pages and the links are hyper-links), and are therefore a good model for our scenario. Figure 5 presents a scale-free network on the left side with highly connected



(a) Network visualization



(b) Node degree distribution

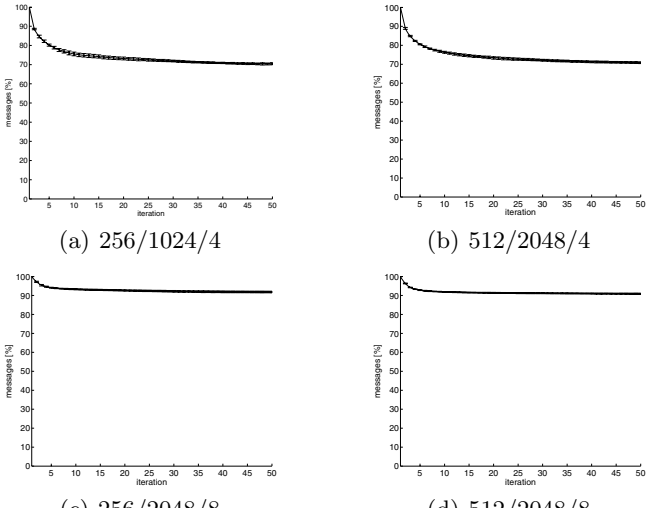
Fig. 5. A scale-free network: 1024 nodes with average node degree 4

nodes in the center and loosely connected nodes at the periphery. The node degree varies between 1 and 62 with an average around 4. The node degree distribution follows a power-law distribution.

5.2 Message Reduction

The most interesting evaluation criterion is of course the message reduction achieved by redistributing the variables close to each other in the P-Grid network. Figures 6 – 8 present the results obtained for the three network topologies. The plots show the achieved message reduction after each iteration of the spring relaxation algorithm by relating the number of required messages to run one iteration of the belief propagation algorithm. At the beginning, 100% of the messages are required, while after each iteration of the spring relaxation algorithm, less messages are required. The message reduction is given with the standard deviation of 10 repeated simulations for each setup.

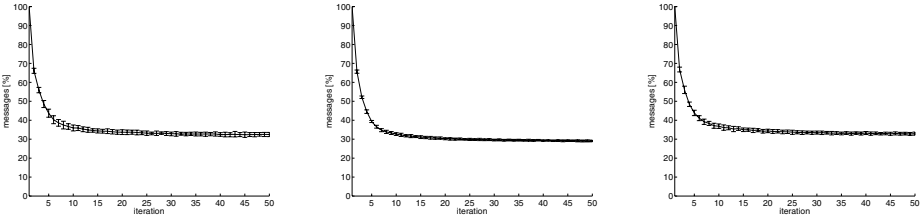
Figure 6 shows that the algorithm does not perform well for any evaluated random network as expected. The random correlations of variables in these networks makes it difficult for the spring relaxation algorithm to cluster variables close to each other to reduce the message effort. The average node degree seems to have the strongest influence on the achieved message reduction which is not larger than 25%. As random networks are not considered as the most realistic model for our use case, this result is tolerable in our opinion. For binary trees, see Figure 7, the relaxation algorithm is already able to reduce the number of required messages to around 35% of the initially required number before running the relaxation algorithm. The obtained results seem to be independent of the



(a) 256/1024/4 (b) 512/2048/4

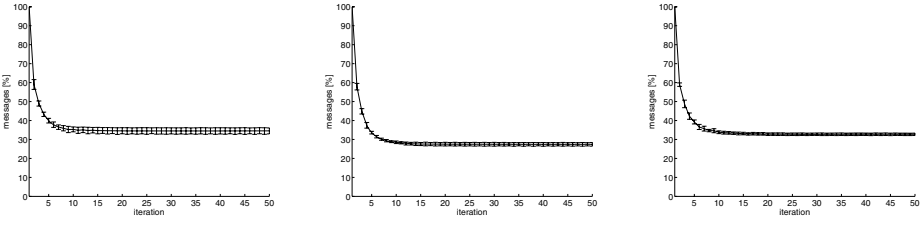
(c) 256/2048/8 (d) 512/2048/8

Fig. 6. Message reduction for random networks with different numbers of nodes/variables/degree



(a) 256/1024 (b) 256/2048 (c) 512/2048

Fig. 7. Message reduction for binary tree-based networks with different numbers of nodes/variables



(a) 256/1024 (b) 256/2048 (c) 512/2048

Fig. 8. Message reduction for scale-free networks with different numbers of nodes/ variables

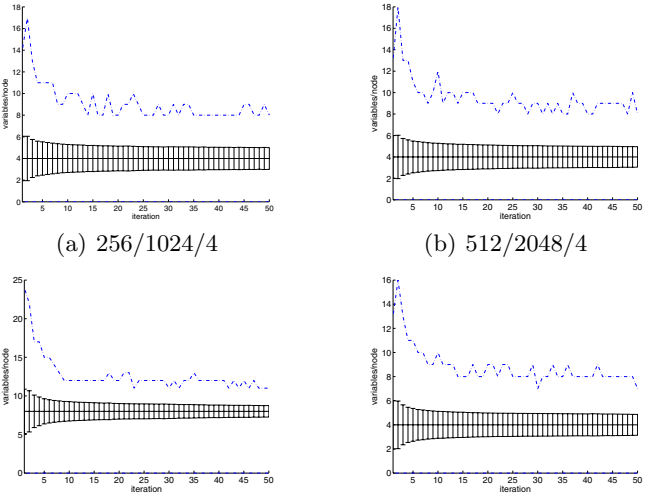
number of nodes and variables. Finally, we observe similar results for the scale-networks as shown in Figure 8. The relaxation algorithm is able to reduce the message cost by about 75% independent of the number of nodes in the P-Grid network and the number of variables in the Bayesian network.

The standard deviation is small for all network topologies and network sizes which is an indicator that the algorithm scales well. In all experiments, the algorithm was iterated 50 times but the main reduction is achieved in the first 10 iterations. Again, this seems to be independent of the number of nodes and number of variables in the networks.

5.3 Load-Balancing

Apart from the reduction of required messages for the message-passing algorithm, it is important that the load of variables is balanced among the participating nodes. Figures 9 – 11 present the corresponding results obtained again for random networks, binary trees and scale-free networks. All figures show the average variable load which remains constant over all iterations as the number of variables and nodes does not change. The standard deviation indicates the load balance in the system. Additionally, the maximum load of nodes is given by the dotted line.

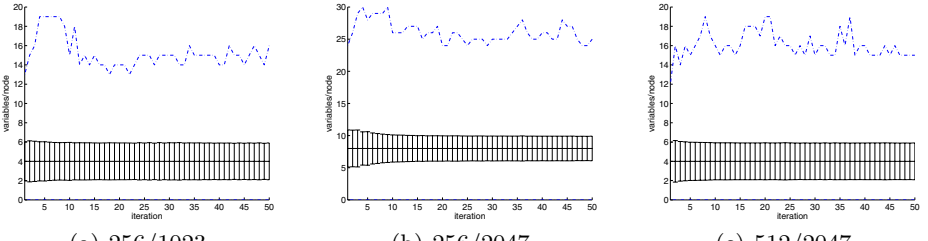
Whereas the relaxation algorithm did not perform well for random networks to reduce the number of required messages, it was more successful to balance the load among the nodes, as shown in Figure 9. The standard deviation is decreasing for all network sizes as well as the maximum number of variables per node (dotted line). Similar results were obtained for the binary tree-based networks (see Figure 10). Figure 11 shows that scale-free networks cause a slight increase of unbalance and a large increase in the maximum load for nodes. We think this is due to the fact that 1 or 2 nodes usually have very high degrees and therefore cause an overload at the P-Grid node they are currently maintained. Our relaxation algorithm is currently not able to handle this problem and we leave this as future work. A simple solution could be to allow nodes to decline maintaining further nodes by introducing an upper bound for the load.



(a) 256/1024/4 (b) 512/2048/4

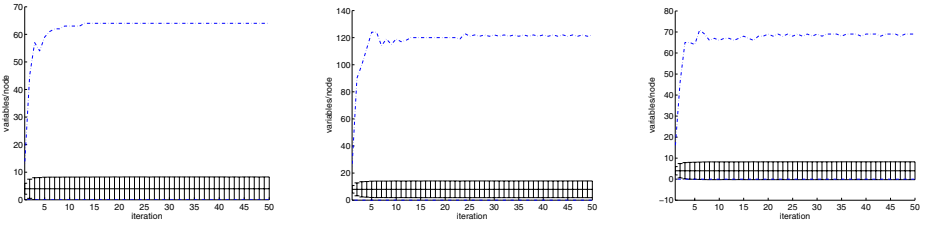
(c) 256/2048/8 (d) 512/2048/8

Fig. 9. Variables per node for random networks with different numbers of nodes/variables/degree



(a) 256/1023 (b) 256/2047 (c) 512/2047

Fig. 10. Variables per node for binary tree-based networks with different numbers of nodes/variables



(a) 256/1024 (b) 256/2048 (c) 512/2048

Fig. 11. Variables per node for scale-free networks with different numbers of nodes/-variables

5.4 Discussion

The results obtained from the first evaluations look very promising. One way to probably further reduce the number of messages apart from the relaxation algorithm is to combine messages to one large message if local variables have a relation with variables at the same remote node. It is usually more efficient to send less large messages than more small messages in a peer-to-peer system.

6 Related Work

6.1 Belief Propagation

Generalized Belief Propagation [6] reduces the number of messages by clustering correlated variables together and sending only one message between those clusters. This approach has three drawbacks: (i) the message sizes increase exponentially ($number\ of\ states^{nodes\ in\ the\ cluster}$) because the exchanged messages now contain the joint probabilities of all nodes and states in the cluster; (ii) the complexity of processing the messages and beliefs at nodes also increases considerable with increasing number of nodes in a cluster; (iii) it is not obvious for us how clusters are formed in a distributed way without central coordination and knowledge which is essential in peer-to-peer systems. Though Generalized Belief Propagation provides more accurate beliefs than Pearl's belief propagation, it is currently not applicable for large-scale networks.

Reference [1] presents an inference architecture for sensor networks based on message-passing on a junction tree. For this approach, a distributed algorithm is first used to form a spanning tree of nodes which is used later to construct the junction tree for inference. Junction trees group variables into cliques and their size determines the computation costs at nodes whereas the separator size between cliques determines the communication costs. The approach was evaluated with 54 sensor motes in a local experiment showing spanning tree optimizations and the communication costs of the junction tree. Inference on junction trees is exact and always results in the exact marginals at the cost of requiring building a tree with larger messages and higher computation costs. Belief propagation only provides approximate inference on lower overheads.

6.2 Spring Relaxation

Spring relaxation is used in various domains and we will only present two examples for peer-to-peer systems. Vivaldi [7] is a decentralized network coordinate system using a spring-mass model to position nodes in a virtual coordinate system according to their latencies. Nodes run the distributed spring relaxation algorithm as soon as a new latency measurement was performed to reduce the distance error between nodes. An application of Vivaldi is described in [8] to optimize the path in stream-based overlay networks. Services are placed on nodes close to each other in the virtual latency space.

7 Future Work

An open issue is the introduction of a stop criteria for the relaxation algorithm so that nodes detect that further iterations will not reduce the number of messages noticeably any more. This is crucial because a constant maximum number of iterations may influence the scalability of our approach. Further, the algorithm currently runs absolutely synchronized at all nodes. As this is not realistic in peer-to-peer systems, the influence of an asynchronous execution has to be investigated. We plan to implement our algorithm in P-Grid to evaluate it on PlanetLab, a global-scale testbed with real network characteristics.

8 Conclusions

We presented a relaxation algorithm making large-scale distributed inference possible in peer-to-peer systems. Our approach is based on belief propagation's simple message-passing algorithm to perform inference and the P-Grid overlay network to store and maintain the required Bayesian network. Nodes of the Bayesian network are redistributed among P-Grid nodes to cluster correlated nodes together to minimize the required message costs for inference. Our purely distributed approach does not require any central coordination nor global knowledge. Matlab evaluations show promising results with message reductions up to 70% for various network topologies and network sizes.

References

1. Paskin, M.A., Guestrin, C.E., McFadden, J.: A robust architecture for distributed inference in sensor networks (2005)
2. Pearl, J.: Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann, San Francisco, CA, USA (1988)
3. Berrou, C., Glavieux, A., Thitimajshima, P.: Near shannon limit error-correcting codes and decoding: Turbo codes. In: Proceedings of the IEEE International Communications Conference. (1993)
4. Bickson, D., Malkhi, D., Rabinowitz, D.: Efficient large scale content distribution. In: Proceedings of the Workshop on Distributed Data and Structures (WDAS), Lausanne, Switzerland (2004)

5. Ihler, A.T., John W. Fisher, I., Moses, R.L., Willsky, A.S.: Nonparametric belief propagation for self-calibration in sensor networks. In: Proceedings of the Third international symposium on Information processing in sensor networks, New York, NY, USA, ACM Press (2004) 225–233
6. Yedidia, J.S., Freeman, W.T., Weiss, Y.: Generalized belief propagation. In: Advances in Neural Information Processing Systems (NIPS). Volume 13. MIT Press (2000) 689–695
7. Dabek, F., Cox, R., Kaashoek, F., Morris, R.: Vivaldi: A decentralized network coordinate system. In: Proceedings of ACM SIGCOMM. (2004)
8. Pietzuch, P., Shneidman, J., Welsh, M., Seltzer, M., Roussopoulos, M.: Path optimization in stream-based overlay networks. Technical Report TR26-04, Harvard University, Cambridge, Massachusetts (2004)
9. Aberer, K.: P-grid: A self-organizing access structure for p2p information systems. In: Proceedings of the 6th International Conference on Cooperative Information Systems (CoopIS), London, UK, Springer-Verlag (2001) 179–194
10. Weiss, Y.: Correctness of local probability propagation in graphical models with loops. Neural Computation **12**(1) (2000) 1–41
11. Kleinberg, J.: The small-world phenomenon: An algorithmic perspective. In: ACM STOC. (2000)
12. Bickson, D., Dolev, D., Weiss, Y., Aberer, K., Hauswirth, M.: Indexing data-oriented overlay networks using belief propagation. In: Proceedings of the Workshop on Distributed Data and Structures (WDAS), Santa Clara, CA, USA (2006)
13. Corporation, M.: Microsoft help and support (2006) http://support.microsoft.com/.
14. Organization, T.M.: Mozillazine knowledge base (2006) http://kb.mozillazine.org/.
15. Organization, T.M.: Bugzilla (2006) http://www.bugzilla.org/.
16. Yamanishi, K.: Distributed cooperative bayesian learning strategies. In: COLT '97: Proceedings of the tenth annual conference on Computational learning theory, New York, NY, USA, ACM Press (1997) 250–262
17. Heckerman, D.: A tutorial on learning with bayesian networks. Technical Report MSR-TR-95-06, Microsoft Research, Redmond, USA (1995)
18. Chen, R., Sivakumar, K., Kargupta, H.: Collective mining of bayesian networks from distributed heterogeneous data. Knowledge and Information Systems **6**(2) (2004) 164–187
19. Batagelj, V.: Pajek - program for large networks analysis and visualization (2001)

Designing Cooperative IS: Exploring and Evaluating Alternatives

Volha Bryl, Paolo Giorgini, and John Mylopoulos

Department of Information and Communication Technology,
University of Trento,
via Sommarive 14, 38050 Povo (TN), Italy
{bryl, paolo.giorgini, jm}@dit.unitn.it

Abstract. At the early stages of the cooperative information system development one of the major problems is to explore the space of alternative ways of assignment and delegations of goals among system actors. The exploration process should be guided by a number of criteria to determine whether the adopted alternative is good-enough. This paper frames the problem of designing actor dependency networks as a multi-agent planning problem and adopts an off-the-shelf planner to offer a tool (P-Tool) that generates alternative actor dependency networks, and evaluates them in terms of metrics derived from Game Theory literature. As well, we offer preliminary experimental results on the scalability of the approach.

1 Introduction

During the requirements analysis and design of cooperative information systems one has to cope with the fundamental problem of planning, i.e. finding optimal/good-enough delegations to a set of system actors which collectively fulfill a given set of goals. These goals are initially assigned to the actors which may not have enough capabilities to satisfy them, so they are decomposed and delegated to other actors, thereby creating networks of delegations. The process ends when all initial goals can be fulfilled if all system actors deliver on their delegations.

Exploring the space of alternative dependency networks is a difficult design task. This is so because such networks represent complex socio-technical systems where organizational, human and system actors depend on each other to fulfill root-level goals. Moreover, there are no generic criteria to guide the design process by determining whether a solution is good-enough, or even optimal. Our ultimate goal is to identify suitable metrics for evaluating alternative sets of delegations to help a designer to select the best one.

The purpose of this paper is to propose a framework for the automatic selection and evaluation of alternative dependency networks, or design alternatives. The framework supports both the generation and evaluation of alternatives. Specifically, the framework adopts multi-agent planning techniques and uses an off-the-shelf planning tool. Alternatives are evaluated with respect to individual interests of system actors (i.e. their own goals). Ideas from Game Theory [14] are used to determine whether an alternative is an equilibrium. In particular, an alternative is an equilibrium if no actor can do better

R. Meersman, Z. Tari et al. (Eds.): OTM 2006, LNCS 4275, pp. 533–550, 2006.

with respect to its own goals by adopting a different strategy for delegating and accepting delegations. When combined together, these two steps support the designer of an information system in selecting alternatives that are in equilibrium with respect to the local strategies of each actor. An early version of this idea is used in [3] to propose a framework to generate alternative designs for secure systems. This paper goes further by describing a prototype tool that generates alternatives, presents some experimental results, and also proposes the evaluation techniques for alternatives based on game-theoretic notions.

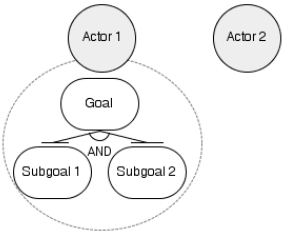
The process of the best alternative selection consists of the following steps:

1. Identify system and human actors, goals and their properties. Define goal decompositions and dependency relationships among actors.
2. For each actor identify criteria to evaluate alternatives.
3. Automatically explore the space of alternatives "on the upper level" to identify assignments of coarse-grained goals to actors.
4. Separately for each actor, automatically explore the alternative ways to satisfy the goals the actor was assigned at step 3. According to the above identified evaluation criteria, select "the best" alternative for each actor. During this step, alternative refinements of coarse-grained goals and delegation dependencies among actors are explored.
5. Evaluate the combined solution consisting of alternatives identified at step 4. In case it does not satisfy one or several system actors (e.g. they are overloaded with respect to others), return to step 4 to search for another alternative.

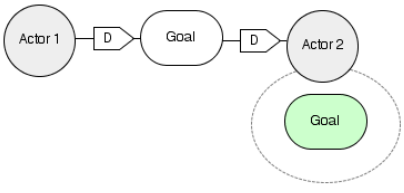
Ideally, the process stops after a number of iterations when the system structure is optimized enough to comply with the individual interests of the system actors. If no satisfactory alternatives can be generated at some step, the designer should return to steps 1 or 2, and revise either the initial structure, or the evaluation criteria.

Figure 1a presents a simple example of the problem of exploring design alternatives. Note that in this paper we use the *i**-like graphical notation [20]. The *i** modeling framework and an associated requirements analysis process (Tropos [2]) are based on the intentional concepts of an actor, a goal and a social dependency, and supports modeling and analysis during the requirements and design phases. So, in the example *Actor 1* has to achieve a *Goal*, which can be refined into two subgoals *Subgoal 1* and *Subgoal 2*. The actor can decide to achieve the goal by itself or delegate it to *Actor 2*. In both cases, there are a number of alternative ways that can be adopted. So, for instance, *Actor 1* can decide to delegate to *Actor 2* the whole *Goal* (Figure 1b), or a part of it (Figure 1c). Shaded goal in the circle of an actor means that the goal is the responsibility of this actor. Even for this primitive example, exploring all the alternatives is quite tedious, and a support for alternative generation and evaluation would be beneficial.

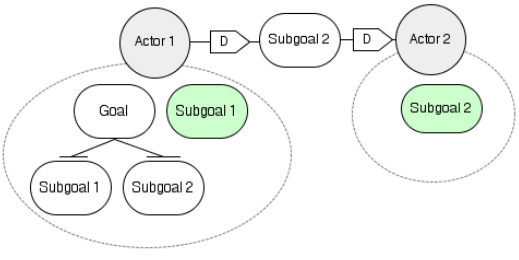
The rest of the paper is structured as follows. In the next section we introduce the example we use through the paper to describe our framework. In Section 3 the issue of alternative generation and evaluation is detailed. Section 4 describes the P-Tool, an implemented prototype tool to support the exploration of alternatives, and reports some experimental results. Finally, in Section 5 we describe the related work, and discuss conclusions and future work directions in Section 6.



(a) Sample problem



(b) 1st alternative



(c) 2nd alternative

Fig. 1. Sample problem and two alternative solutions

2 SDS System Example

Let us consider a small software development company, which typical projects are medium-scale web based information systems (like, e.g., online library catalog, or travel agency home page with online trip booking, etc.). Within the company there are three teams of developers each focused on its area: GUI development, web design, and database support. Each team can develop subcomponents and/or consult other teams on questions related to their expertise. A manager is supposed to divide the project into meaningful parts and perform the assignment of goals to achieve to the development teams. Note, that a manager can assess how the project goals are refined and what skills are required to satisfy each subgoal only on the coarse-grained level.

The company has decided to use a software development support system (SDS system or supporting system in the following), which will facilitate and report communication among actors, archive a library of reusable components, organize the search for such components, store and provide the information specific to the project under development (e.g. contain a glossary of domain specific terms, store domain specific

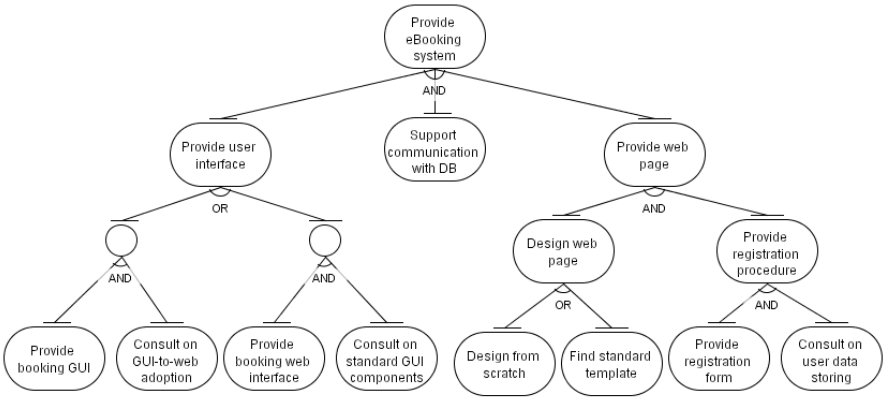


Fig. 2. Goal tree for eBooking project

classifications, etc.). Communication between manager and members of the development teams is supposed to be carried out only through the supporting system. Teams can communicate with each other in two cases: when one team wants to redirect a subgoal which requires the development skills these team does not possess to another team, and when one team needs to consult another one. The first type of communication is possible only through the supporting system, while the communication on consultancy can be done both through the supporting system and e-mail (or even personal communication).

To analyze the above described system let us consider a typical project it might deal with: web based eBooking system for the travelling agency. As it is represented in Figure 2, the high-level goal *provide eBooking system* is refined into three subgoals: *provide user interface*, *support communication with DB* and *provide web page*. In order to fulfil the high-level goal, all three subgoals should be satisfied. Two subgoals are further refined, e.g. *provide user interface* subgoal can be reached in two alternative ways: by developing eBooking GUI and consulting web designers to adopt it for web environment, or by developing web booking interface together with consulting GUI team on which standard components to use.

The OR-decomposition of the subgoals *provide user interface* and *design web page* introduces alternative solutions for the development of the eBooking system. One of the alternatives to achieve *provide user interface* subgoal is depicted in Figure 3. The goal is decomposed by *GUI team* actor, which selects one of the alternatives among the two or-subgoals. The selected subgoal is further decomposed into two subgoals: *consult on standard GUI components* and *provide booking web interface*. The former is satisfied by *GUI team*, while the latter is delegated to *Web design team*.

3 Exploring and Evaluating Alternatives

3.1 Formalization of the Planning Problem

As it was discussed in the Introduction, requirements to the information systems are conceived as networks of delegations among actors – at least within the frameworks

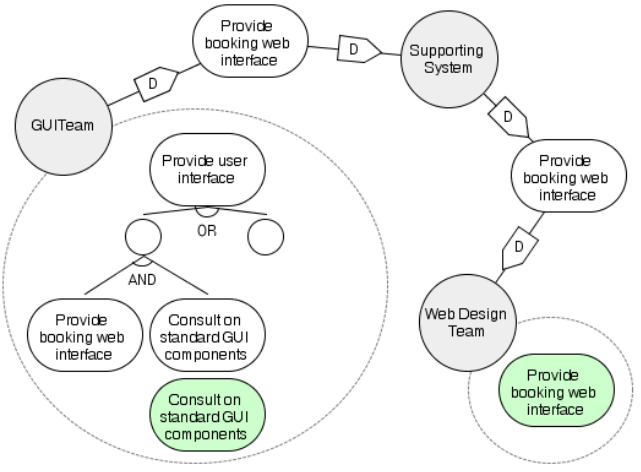


Fig. 3. An alternative way to achieve *provide user interface* subgoal

such as *i** [20], Tropos [2] and the like. Every delegation involves two actors, where one actor delegates to the other the fulfillment of a goal. The delegatee can either fulfill delegated goal, or further delegate it, thus creating another delegation relation in the network. Intuitively, these can be seen as actions that the designer/requirements engineer ascribes to the members of the organization and the system-to-be. Further, the task of constructing such networks can be framed as a planning problem: selecting a suitable system structure corresponds to selecting a plan that satisfies the goals of human and software agents.

Thus, we have chosen the AI (Artificial Intelligence) planning approach to support the designer/requirements engineer in the process of selecting the best design alternative. The basic idea behind planning approach is to automatically determine the course of actions (i.e. a plan) needed to achieve a certain goal where an action is a transition rule from one state of the system to another [19,15]. Actions are described in terms of preconditions and effects: if the precondition is true in the current state of the system, then the action is performed. As a consequence of an action, the system will be in a new state where the effect of the action is true.

Planning approach requires a specification language to represent the planning domain, i.e.

- the initial state of the system;
- the goal of the planning problem (i.e. the desired final state of the system);
- the description of actions;
- the axioms of background theory.

Once the domain is described, the solution to the planning problem is the (not necessarily optimal) sequence of actions that allows the system to reach the desired state from the initial state.

Table 1. Primitive predicates

Goal Properties
type(g : goal, gt : gtype)
subtype(child : gtype, parent : gtype)
and_decomposition$_n$(g : goal, g_1 : goal, . . . , g_n : goal)
or_decomposition$_n$(g : goal, g_1 : goal, . . . , g_n : goal)
satisfied(g : goal)
Actor Properties
can_satisfy(a : actor, g : goal)
can_satisfy_gt(a : actor, gt : gtype)
can_decompose_gt(a : actor, gt : gtype)
wants(a : actor, g : goal)
Actor Relations
can_depend_on(a : actor, b : actor)
can_depend_on_gt(a : actor, b : actor, gt : gtype)

To describe the initial state of the system we should specify actor and goal properties, and social relations among actors. We propose to represent the initial state in terms of predicates that correspond to

- the possible ways of goal decomposition;
- actor capabilities and desires to achieve a goal;
- possible delegation relations among actors.

The desired state of the system (or the goal of the planning problem) is described through the conjunction of predicates derived from the description of actor desires in the initial state. Essentially, for each desired goal a predicate is added to the goal of the planning problem.

Different types of logic could be applied for this purpose, e.g. first order logic is often used to describe the planning domain with conjunctions of literals specifying the states of the system. In Table 1 predicates used to describe the domain of information system design are introduced. Predicates take variables of three types: actors, goals and goal types. To typify goals, *type* predicate is used. Actor capabilities are described with *can_decompose* and *can_decompose_gt* predicates, which means that an actor has enough capabilities to satisfy a specific goal or any goal of a specific type, accordingly. Social dependencies among actors are reflected by *can_depend_on* and *can_depend_on_gt* predicates, which means that one actor can delegate to another actor the fulfilment of any goal or, in the latter case, any goal of a specific type. Predefined ways of goal refinement are represented using *decomposition* predicates, while with *can_decompose_gt* the scope of each actor can be represented: an actor can refine, or knows how to refine, only goals within his scope. Initial actor desires are represented with *wants* predicate. When the goal is fulfilled *satisfied* predicated becomes true for it.

In Figure 4, a part of SDS System example formalization is presented. The goal types *tConsult*, *tWDDevel* and *ManagScope* are used.

In *i**/Tropos approach, when drawing the model of a system, the designer/requirements engineer assigns goals to actors, defines delegations of goals from

```
type (ConsultOnGUIToWebAdoption, tWDConsult)
subtype (tWDConsult, tConsult)
can_depend_on_gt (GUITeam, WDTeam, tConsult)
can_depend_on_gt (WDTeam, GUITeam, tConsult)

type (ProvideBookingWebInterface, tWDDevel)
type (DesignFromScratch, tWDDevel)
type (ProvideRegistrationForm, tWDDevel)
can_satisfy_gt (WDTeam, tWDDevel)

type (ProvideEBookingSystem, tManagScope)
can_decompose_gt (Manager, tManagScope)
```

Fig. 4. Predicates for SDS System example

one actor to another, and identifies appropriate goal refinements among the predefined alternative refinements. Thus, the following actions will be used by a planner to find a way to fulfill the goals of the system actors.

Goal satisfaction. An actor can satisfy a goal only if the achievement of the goal is among its desires and it can actually satisfy it. The effect of this action is the fulfillment of the goal.

Goal delegation. An actor may have not enough capabilities to achieve its goals by itself, and so it has to delegate their satisfaction to other actors. This passage of responsibilities is performed only if the delegator wants a goal to be achieved and can depend on the delegatee to achieve it. The effect of this action is that the delegator does not worry any more about the satisfaction of the goal, while the delegatee takes the responsibility for the fulfillment of the goal and so it becomes its own desire to achieve it. The delegator does not care how the delegatee satisfies the goal (e.g. by its own capabilities or by further delegation), it is up to the delegatee to decide it.

Goal decomposition/refinement. As in different goal-oriented modeling frameworks (e.g. as in Tropos and KAOS [5]) two types of goal refinement are supported: OR-decomposition, which suggests the list of alternative ways to satisfy the goal, and AND-decomposition, which refines the goals into subgoals which all are to be satisfied in order to satisfy the initial goal. An actor can decompose a goal only if it wants it to be satisfied, and only in the way which is predefined in the initial state of the system. The effect of decomposition is that the actor which refines the goal focuses on the fulfillment of subgoals instead of the initial goal. It is assumed that different actors can decompose the same goal in different ways.

In addition to actions, axioms of the planning domain can be defined. These are rules that hold in every state of the system and are used to complete the description of the current state. For example, to propagate goal properties along goal refinement the following axiom is used: a goal is satisfied if all its and-subgoals or at least one of the or-subgoals are satisfied.

3.2 Evaluation Procedure

The alternative designs generated by the planner should be evaluated, amended and approved by the designer. The tricky point here is the solution evaluation which can be complex enough even for experienced designers with considerable domain expertise. Alternative requirements structures can be evaluated both from global and local perspectives, i.e. from the designer's point of view and from the point of view of individual actors. The optimality of a solution in the global sense could be assessed with respect to the following.

- *Length of the obtained plan.* The number of actions in the obtained plan is often the criteria for the planner itself to prefer one solution to another. Thus, it can be assumed that the obtained plan is already (locally) optimal in the sense of the length minimization.
- *Overall plan cost.* This is closely related with the idea of plan metrics introduced in PDDL 2.1 [8]. Plan metrics specify the basis on which a plan is evaluated for a particular problem (e.g. action costs or duration), and are usually numerical expressions to be minimized or maximized. However, the complexity of the problem of optimizing a solution with respect to the defined metrics is very high and the feature is still poorly supported by the available planning tools [8].
- *Degree of satisfaction of non-functional requirements.* E.g. in [12], a set of rules is proposed to identify application-specific parameters and functions to quantify impacts of different explored alternatives on non-functional goals (e.g. security, performance, usability) satisfaction.

In this paper we are not dealing with the global evaluation of generated alternatives. However, the first point, (sub)optimality of the solution with respect to the plan length, is automatically taken into account by the planner.

Local evaluation of the obtained plan is a much more complex task. Indeed, a challenging characteristic of modern information system design is that the human agents should be taken into account. They can be seen as players in a game theoretic sense as they are self-interested and rational. This means they want to minimize the load imposed personally on them, e.g. they want to constraint the number and the complexity of actions they are involved in[1]. In a certain sense non-human agents, i.e. system components, are players as well as it is undesirable to overload them. Each player has a set of strategies he could choose from, e.g. he could decide whether to satisfy a goal himself or to pass it further to another system actor. Strategies are based on the player's capabilities and his relations (e.g. subordination, friendship, or trust – all represented as possible dependencies in our framework) with other human and artificial agents in the system.

The substantial difficulty in applying game theoretic ideas to our problem is that all actors of an information system should work cooperatively as a solid mechanism satisfying the overall organizational goal. Differently from classical non-cooperative game theory, where all players choose their strategies independently and simultaneously

[1] In this work we focus on the load constraints only, and do not consider other factors which influence the player's decision to deviate, e.g. risk concerns.

Table 2. Costs for the SDS System example

Action	Cost	Actors and Goals
Satisfy	3	goals of type *tConsult* for *WDTeam*, *GUITeam* and *DBTeam*; goal *FindStandardTemplate* for *SupportingSystem*
	4	goals of type *tWDDevel* for *WDTeam*, goal *ProvideBookingGUI* for *GUITeam*; goal *SupprtDBCommunication* for *DBTeam*
Delegate	1	*SupportingSystem*; delegations between *WDTeam*, *GUITeam* and *DBTeam*
	2	all other actors and goals
Refine	2	all actors and goals

before the game, in our problem actors' choices are closely interrelated. A player cannot independently change his strategy because the new action sequence will very likely be unsatisfactory, i.e. it will not be a solution anymore. Thus, to satisfy the system goals it will be necessary to impose some additional load (to compensate the load this player tries to avoid) on some other actors – and it might happen that they will not be satisfied with the new solution, and will try to deviate from the strategy they were imposed, and so on and so forth. Thus, if one actor wants to deviate from the generated solution, the re-planning is needed to search for another alternative option, which is then evaluated, possibly, to be re-plan again. The process stops when a (sub)optimal alternative option is found. In our framework the following "replan-towards-optimality" procedure is used.

First, for all actors a_i, $i = \overline{1,n}$ and all goals g_k, $k = \overline{1,m}$, where n and m are the number of actors and goals, respectively, the costs are defined:

- cs_{ik} is the cost for the actor a_i of satisfying the goal g_k;
- cr_{ik} is the cost for the actor a_i of refining the goal g_k;
- cd_{ijk} is the cost for the actor a_i of delegating a goal g_k to the actor b_j.

In Table 2 the costs of actions for actors from the SDS System example are defined.

Then, the cost of a given alternative P for the actor a_i is calculated by summing up the costs of actions in P which a_i is involved in, and is denoted by

$$c\,(P, a_i) = \sum_{delegate(a_i,b_j,g_k)\in P} cd_{ijk} + \sum_{decompose_l(a_i,g_k,g_{k1},\ldots,g_{kl})\in P} cr_{ik} + \sum_{satisfy(a_i,g_k)\in P} cs_{ik},$$

where $decompose_l(a_i, g_k, g_{k1}, ..., g_{kl})$ stands for the decomposition of g_k into l subgoals $g_{k1}, ..., g_{kl}$.

If P is the alternative depicted in Figure 3, then $c\,(P, GUITeam) = 2+2+3+2 = 9$, $c\,(P, WDTeam) = 2 + 4 = 6$ and $c\,(P, SupportingSystem) = 1 + 1 = 2$.

Note, that in our framework we do not use the notion of utility, which is an important game theory construct. This is done mainly for the simplicity reasons. The utility of an alternative P for the actor a_i can be defined as the difference between maximum upper bound for the solution cost for actor a_i and $c\,(P, a_i)$. Basically, utility says how much an actor "saves" with the alternative P being selected.

After the costs are computed, for each actor the conditions are defined upon which an actor decides whether to deviate from an alternative P or not. The conditions could be either one of the following, or both.

- Actor a_i whose predefined upper cost bound c_i^{up} is less than $c\,(P, a_i)$ is willing to deviate from P.
- Actor a_i whose predefined upper bound $cdev_i^{up}$ on cost deviation is less than $c\,(P, a_i) - avg_i(c\,(P, a_i))$ wants to deviate from P.

In this work we consider only the first deviation condition – predefined upper cost bounds.

Finally, the evaluation procedure is the following.

- An alternative P is generated with the help of a planner.
- Cost $c\,(P, a_i)$ is calculated for each a_i.
- Actor a_{min} is identified which value of $c\,(.,.)$ is minimal among all actors which want to deviate from P.
- The first most expensive action d_{worst} (the one with the highest cost) is identified among actions of P in which a_{min} is involved.
- Negation of d_{worst} is added to the initial planning problem, and replanning is performed. If no plan can be found, the next d_{worst} is identified.

The process stops when an equilibruium-like solution is found, i.e. no actors are willing to deviate from it and the designer approves this solution. The designer remains in the process all the time, and can stop the iterations whenever he thinks the satisficing alternative is generated.

This evaluation procedure is used at the following steps of the selection of the best alternative, defined in the Introduction.

- At step 3, while selecting the best assignments of coarse-grained goals to actors.
- At step 4, separately for each actor, when exploring the ways to satisfy the goals the actor was assigned.
- At step 5, when evaluating the combined solution consisting of alternatives identified at step 4. Here the replanning is performed only for the alternative to which d_{worst} belongs to.

4 P-Tool and Experiments

4.1 Choosing the Planner

One important step we have performed during the implementation of the proposed framework, is choosing the "right planner" among off-the-shelf tools available. In the last years many planners have been proposed [15]. In order to choose one of them the following requirements were considered:

- The planner should not produce redundant plans. Under non-redundant plan we mean that, by deleting an arbitrary action of the plan, the resulting plan is no more a "valid" plan (i.e. it does not allow to reach the desired state from the initial state).

```
(: action Satisfies
   : parameters(?a − t_actor, ?g − t_goal
   : precondition (and
      (or(can_satisfy?a?g)
         (exists(?gt − t_gtype)(and(type?g?gt)
            (can_satisfy_gt?a?gt)))
      (wants?a?g)
   : effect (and
      (satisfied?g)
      (not(wants?a?g))))

(: derived
   (type?g − t_goal?parent − t_gtype)
   (exists(?child − t_gtype)
      (and(subtype?child?parent)(type?g?child))))
```

Fig. 5. Domain description using PDDL

- The planner should use PDDL (Planning Domain Definition Language) since it is becoming a "standard" planning language and many research groups work on its implementation.
- The language should support a number of "advanced" features (e.g. derived predicates) that are essential for implementing our planning domain, i.e. it should be at least PDDL 2.2. [6].

The first requirement is related to the question of the optimality of the generated design decisions. We argue that it is not necessary to focus on the optimal design: human designers do not prove that their design is optimal, why should a system do it? Instead, in our framework the plan is required to be non-redundant, which guarantees at least the absence of alternative delegation paths since a plan does not contain any redundant actions.

We have compared a number of planners with respect to above requirements (see [3] for the details). Finally, we have chosen LPG-td [13], a fully automated system for solving planning problems, supporting PDDL 2.2 specification for implementing our planning domain.

Then, we have implemented our planning domain in PDDL 2.2. Figure 5 presents the specification of one action and one domain axiom in PDDL 2.2.

Figure 6 shows the plans generated by LPG-td for satisfying *provide user interface* and *provide web page* subgoals. The former plan is illustrated in Figure 3, the latter – in Figure 7.

Preliminary experiments were conducted to test the scalability of the approach. A very simple "core" problem was considered, with three actors A, B and C and two goals, G_1 and G_2, which A wants to be achieved, and B and C can satisfy. Then "additional" actors with the dependencies among them were added to the problem, but they did not interfere at all with "core" subproblem. The idea was to check whether the search time of the plan to achieve G_1 and G_2 depends on the number of "additional" actors and dependencies among them. The experiments showed that, at least with respect

```
(OR_DECOMPOSES GUITeam ProvideUI ProvideUI1 ProvideUI2)
(AND_DECOMPOSES GUITeam ProvideUI1
    ProvideBookingWebI ConsultStandGUI)
(SATISFIES GUITeam ConsultStandGUI)
(PASSES GUITeam SupportingSystem ProvideBookingWebI)
(PASSES SupportingSystem WDTeam ProvideBookingWebI)
(SATISFIES WDTeam ProvideBookingWebI)
```

(a) Provide user interface

```
(AND_DECOMPOSES WDTeam ProvideWebPage
    DesignWebPage ProvideRegistrProc)
(AND_DECOMPOSES WDTeam ProvideRegistrProc
    ProvideRegForm ConsultOnStoreUData)
(SATISFIES WDTeam ProvideRegForm)
(OR_DECOMPOSES WDTeam DesignWebPage
    DesignFromScratch FindStandardTemplate)
(SATISFIES WDTeam DesignFromScratch)
(PASSES WDTeam DBTeam ConsultOnStoreUData)
(SATISFIES DBTeam ConsultOnStoreUData)
```

(b) Provide web page

Fig. 6. Plans for *ProvideUI* and *ProvideWebPage* subgoals

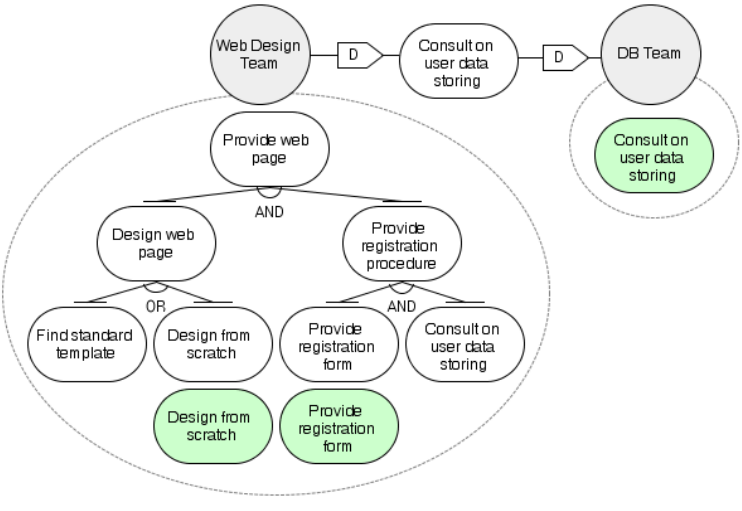
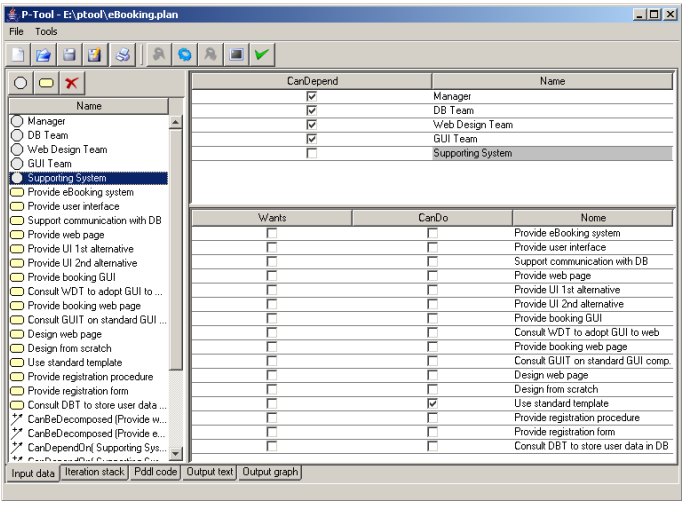
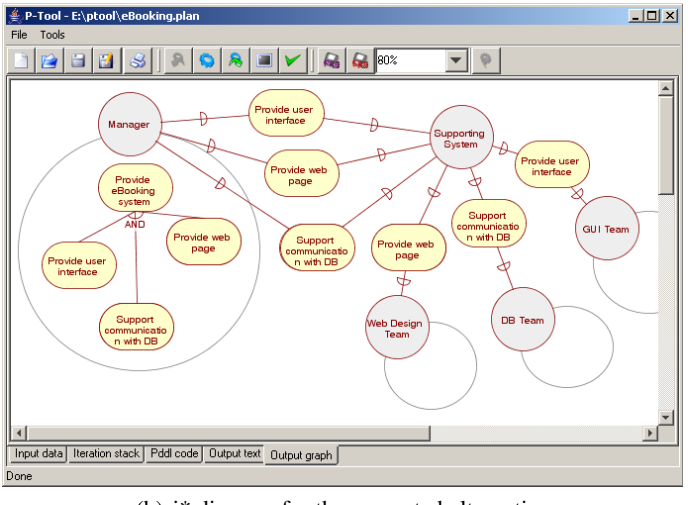


Fig. 7. Diagram for the plan for *ProvideWebPage* subgoal

to this example, the approach is scalable. Basically, the search time for the problem with 10 and with 120 "additional" actors is the same (less than one second), only the parsing time increases insignificantly. At the same time, search time for the plan with long delegation chains (more than 30 steps) is much greater (around 15 seconds). Of course, the scalability issue should be explored much more carefully (actually, this is



(a) Identifying actor properties



(b) i* diagram for the generated alternative

Fig. 8. P-Tool

one of our future work plans), but the above reported preliminary experiments have shown promising results.

4.2 P-Tool

We have developed P-Tool, an implemented prototype to support the designer/requirements engineer in the process of exploring and evaluating alternatives. The tool has the interface for the input of actors, goals and their properties, which can be seen in Figure 8a. LPG-td is built in the tool, and is used to generate design

alternatives, which are then represented graphically using i^* notation, see Figure 8b for an example.

In the following we will illustrate how the steps 3–5 of our approach (see Introduction) could be supported by the P-Tool. For the sake of simplicity we will leave out some details. Steps 1 and 2 are illustrated in Section 3. Predefined upper cost bound c_i^{up} for all actors a_i, $i = \overline{1,n}$ is equal to 14 units on step 4, and 18 units on step 5 (if no solution can be found at step 4, the constraints could be relaxed by increasing the upper cost bound up to 18).

Step 3. First, the planning "on the upper level" for *Manager* actor is performed. We will skip the process description. The resulting alternative can be seen in the screenshot in Figure 8b. *Manager* decomposes *ProvideEBookingSystem* goal into *ProvideUI*, *ProvideWebPage* and *SupportDBCommunication* subgoals, and passes them through the *SupportingSystem* to *GUITeam*, *WDTeam* and *DBTeam*, respectively.

Step 4. We will illustrate this step with exploring alternatives for the subgoal *ProvideWebPage* assigned to *WDTeam* actor. Firstly, an alternative presented in Figure 6 and Figure 7 is generated. For this alternative $c\,(P_1, WDTeam) = 2{+}2{+}2{+}4{+}4{+}1 = 15$, which does not satisfy *WDTeam* actor ($c_i^{up} = 14 < 15$), so it tries to decrease the imposed load. According to the evaluation procedure described in Section 3.2, the action (SATISFIES WDTeam ProvideRegForm) is selected as d_{worst}. When this action is negated, the planner is not able to find a solution. Thus, the next d_{worst} is identified, which is (SATISFIES WDTeam DesignFromScratch). New alternative is generated, see Figure 9, for which $c\,(P_2, WDTeam) = 2 + 2 + 2 + 4 + 2 + 1 = 13$. This last alternative is then fixed as it satisfies *WDTeam* actor.

Step 5. When partial plans are combined into the plan P and evaluated, it appears that $c\,(P, GUITeam) = 9$ and $c\,(P, WDTeam) = 13 + 6 = 19$. Actor *WDTeam* tries to deviate from the alternative P, and (SATISFIES WDTeam ProvideBookingWebI) of the plan depicted in Figure 6 is identify as d_{worst} and negated. By replanning we get an alternative presented in Figure 10, for which $c\,(P', GUITeam) = 2{+}2{+}1{+}2 = 9$ and $c\,(P', WDTeam) = 1 + 3 = 4$, thus the overall cost of a new combined solution for $WDTeam$ being equal to $4 + 13 = 17 < 18$. This new alternative satisfies both *GUITeam* and *WDTeam* actors.

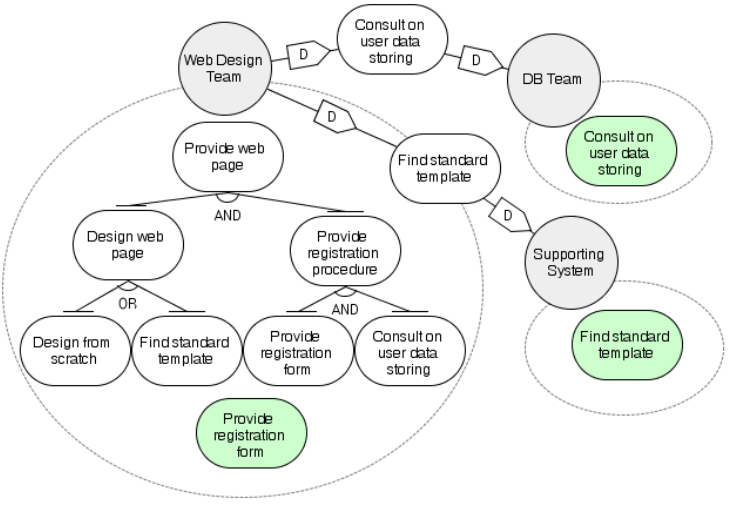
5 Related Work

Modeling requirements and designing information systems and organizations in terms of goals and their interdependences has been a topic of considerable research interest during the last decades [18]. A number of goal-oriented approaches for requirements representation and reasoning were introduced, e.g. KAOS [5]. Requirements engineering is considered to be a crucial part of software development process [18]. Careful elicitation and analysis of requirements help to develop a system that meets user's expectations, is trustful and robust.

The field of AI planning has been intensively developing during the last decades, and has found a number of applications (robotics, process planning, autonomous agents, etc.). Planning approach recently has proved to be applicable in the field of automatic Web service composition [15]. There are two basic approaches to the solution of plan-

```
(AND_DECOMPOSES WDTeam ProvideWebPage
    DesignWebPage ProvideRegistrProc)
(AND_DECOMPOSES WDTeam ProvideRegForm
    ConsultOnStoreUData)
(SATISFIES WDTeam ProvideRegForm)
(PASSES WDTeam DBTeam ConsultOnStoreUData)
(SATISFIES DBTeam ConsultOnStoreUData)
(OR_DECOMPOSES WDTeam DesignWebPage
    DesignFromScratch FindStandardTemplate)
(PASSES WDTeam SupportingSystem FindStandardTemplate)
(SATISFIES SupportingSystem FindStandardTemplate)
```

(a) Plan



(b) Diagram

Fig. 9. New plan for *ProvideWebPage* subgoal

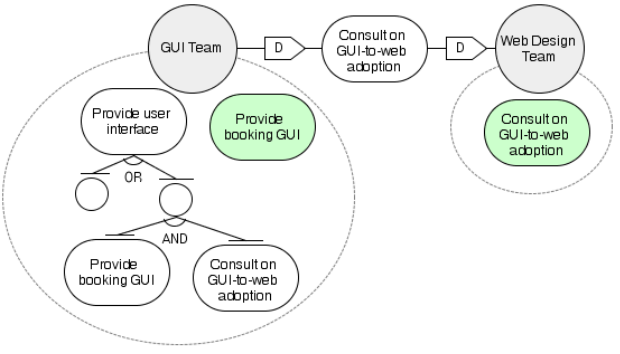
ning problems [19]. One is graph-based planning algorithms in which a compact structure, called Planning Graph, is constructed and analyzed. In the other approach the planning problem is transformed into a SAT problem and a SAT solver is used.

There exist several ways to represent the elements of a classical planning problem, i.e. the initial state of the world, the system goal, or the desired state of the world, and the possible actions system actors can perform. The widely used, and to the certain extend standard representation is PDDL (Planning Domain Definition Language), the problem specification language proposed in [10]. Current PDDL version, PDDL 2.2 [6] used during the last International Planning Competition [11], supports many useful features, e.g. derived predicates and timed initial literals.

A few works can be found which relate planning techniques with information systems requirements analysis and design. In [1] a program called ASAP (Automated Specifier And Planner) is described, which automates a part of the domain-specific software specification process. ASAP assists the designer in selecting methods for

```
(OR_DECOMPOSES GUITeam ProvideUI ProvideUI1 ProvideUI2)
(AND_DECOMPOSES GUITeam ProvideUI2
    ProvideBookingGUI ConsultOnGUIToWebAdoption)
(SATISFIES GUITeam ProvideBookingGUI)
(PASSES GUITeam WDTeam ConsultOnGUIToWebAdoption)
(SATISFIES WDTeam ConsultOnGUIToWebAdoption)
```

(a) Plan



(b) Diagram

Fig. 10. New plan for *ProvideUI* subgoal

achieving user goals, discovering plans that result in undesirable outcomes, and finding methods for preventing such outcomes. The disadvantage of the approach is that the designer still performs a lot of work manually while determining the combination of goals and prohibited situations appropriate for the given application, defining possible start-up conditions and providing many other domain-specific expert knowledge.

Castillo et al. [4] present an AI planning application to assist an expert in designing control programs in the field of Automated Manufacturing. The system they have built integrates POCL, hierarchical and conditional planning techniques (see [4,15] for references). The authors consider standard planning approaches to be not appropriate with no ready-to-use tools for the real world, while in our paper the opposite point of view is advocated. Another recent application of the planning approach to the design of the secure systems is proposed by Gans et al. [9]. The work is based on *i** modeling approach [20] and ConGolog (see [15] for description and references), a logic-based planning language. However, the authors focus more on representing/modeling trust in social networks, than on the design automation, and do not go far in explaining how they exploit the planning formalism.

Game theory is an established discipline which deals with conflicts and cooperation among rational independent decision-makers, or players. The key concept in classical game theory is the notion of equilibrium [14] which defines the set of strategies, one for each player, which none of the independent rational players wants to deviate from. By playing an equilibrium each player maximizes his utility locally, given some constraints. For example, playing the Nash equilibrium means that no player can benefit

when deviating from his equilibrium strategy given that all other players play the equilibrium.

Game theory is applied in various areas, especially in economics (modeling markets, auctions, etc.), corporate decision making, defense strategy, telecommunications networks and many others. Among the examples are the applications of game theory to so called network games (e.g. routing, bandwidth allocation, etc.), see [17] for references.

6 Conclusions

We have proposed a framework for automatic exploration of the space of alternative actor dependency networks that satisfy an initial set of system actor goals. The framework uses planning techniques to explore the space of design alternatives. A prototype tool (P-Tool) with a built-in off-the-shelf planner is used to generate alternatives. These are evaluated in terms of criteria founded on game-theoretic notions.

This is clearly a first step towards making more systematic and tool-supported the process of designing actor dependency models for a given set of initial stakeholder goals. More needs to be done to ensure the scalability of the P-Tool. In particular, we would like to include the use of heuristic (e.g., *A**-like [16]) techniques to reduce the space of alternatives under considering by filtering away early on alternatives that look bad. We would also like to adopt proposals for better structuring actor dependency models. One such proposal [7] is to make *i** models "service-oriented" by encapsulating composite actors and allowing delegations to it only through a well-defined service interface. Such proposals reduce dramatically the number of possible solutions to a given multi-actor planning problem.

Acknowledgements

We thank Alfonso Gerevini and Alessandro Saetti for the support on the use of LPG-td planner. This work has been partially funded by EU Commission, through the SENSORIA and SERENITY projects, by the FIRB program of MIUR under the ASTRO project, and also by the Provincial Authority of Trentino, through the MOSTRO project.

References

1. J. S. Anderson and S. Fickas. A proposed perspective shift: viewing specification design as a planning problem. In *IWSSD '89: 5th Int. workshop on Software specification and design*, pages 177–184, 1989.
2. P. Bresciani, P. Giorgini, F. Giunchiglia, J. Mylopoulos, and A. Perini. Tropos: An agent-oriented software development methodology. *JAAMAS*, 8(3):203–236, 2004.
3. V. Bryl, F. Massacci, J. Mylopoulos, and N. Zannone. Designing secure systems through planning. In *CAiSE'06*, pages 33–47. Springer, 2006.
4. L. Castillo, J. Fdez-Olivares, and A. Gonzlez. Integrating hierarchical and conditional planning techniques into a software design process for automated manufacturing. In *ICAPS 2003, Workshop on Planning under Uncertainty and Incomplete Information*, pages 28–39, 2003.

5. A. Dardenne, A. van Lamsweerde, and S. Fickas. Goal-directed requirements acquisition. *Science of Computer Programming*, 20:3–50, 1993.
6. S. Edelkamp and J. Hoffmann. Pddl2.2: The language for the classical part of the 4th international planning competition. Technical Report 195, University of Freiburg, 2004.
7. H. Estrada. Private communication.
8. M. Fox and D. Long. Pddl2.1: An extension to pddl for expressing temporal planning domains. *J. Artif. Intell. Res. (JAIR)*, 20:61–124, 2003.
9. G. Gans, M. Jarke, S. Kethers, and G. Lakemeyer. Modeling the impact of trust and distrust in agent networks. In *AOIS-01*, pages 45–58, 2001.
10. M. Ghallab, A. Howe, C. Knoblock, D. McDermott, A. Ram, M. Veloso, D. Weld, and D. Wilkins. PDDL – The Planning Domain Definition Language. In *AIPS-98*, 1998.
11. IPC-4 Homepage. International Planning Competition 2004. http://ls5-www.cs.uni-dortmund.de/ edelkamp/ipc-4/.
12. E. Letier and A. van Lamsweerde. Reasoning about partial goal satisfaction for requirements and design engineering. *SIGSOFT Softw. Eng. Notes*, 29(6):53–62, 2004.
13. LPG Homepage. LPG-td Planner. http://zeus.ing.unibs.it/lpg/.
14. M. J. Osborne and A. Rubinstein. *A Course in Game Theory*. MIT Press, 1994.
15. J. Peer. Web Service Composition as AI Planning – a Survey. Technical report, University of St. Gallen, 2005.
16. S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, second edition, 2002.
17. E. Tardos. Network games. In *Proceedings of the Annual ACM Symposium on Theory of Computing*, 2004.
18. A. van Lamsweerde. Requirements engineering in the year 00: a research perspective. In *ICSE-00*, pages 5–19. ACM, 2000.
19. D. S. Weld. Recent Advances in AI Planning. *AI Magazine*, 20(2):93–123, 1999.
20. E. S.-K. Yu. *Modelling strategic relationships for process reengineering*. PhD thesis, University of Toronto, 1996.

Natural MDA: Controlled Natural Language for Action Specifications on Model Driven Development

Luciana N. Leal[1], Paulo F. Pires[2], Maria Luiza M. Campos[1], and Flávia C. Delicato[2]

[1] DCC - IM/NCE
Federal University of Rio de Janeiro
P.O. Box 2324 - Rio de Janeiro - RJ - 20.001-970 - Brazil
[2] DIMAp
Federal University of Rio Grande do Norte
P.O. Box 1524 - Natal - RN - 59.072-970 – Brazil
lucianan1@posgrad.nce.ufrj.br, paulo.pires@dimap.ufrn.br,
mluiza@nce.ufrj.br, flavia.delicato@dimap.ufrn.br

Abstract. Current technologies are continuously evolving and software companies need to adapt their processes to these changes. Such adaptation often requires new investments in training and development. To address this issue, OMG defined a model driven development approach (MDD) which insulates business and application logic from technology evolution. Current MDD approaches falls short in fully derive implementation from models described at a high abstraction level. We propose a controlled natural language to complement UML models as an action specification language. In this article, we describe the language, its impact on systems development and the tools developed to support it. In order to demonstrate the language usability we present an application example.

1 Introduction

Modeling techniques have a long tradition in the software engineering area and their efficient use is a critical factor to the success of every enterprise-scale solution. Recent researches have tried to push modeling techniques a step further by focusing on two main points: (i) enhancing existent models with notations that allow users to express system perspectives with value added to software architects and developers; and (ii) proposing tools and techniques that allow models to be directly and fully mapped into a programming language code. One widely investigated proposal in this sense is Model Driven Development (MDD) [15]. MDD is a model driven system development approach where extensive models are progressively built and refined during project life cycle until there is enough information to generate code. Model Driven Architecture (MDA) is the OMG [23] view of MDD. The main goal of MDA approach is to make the application logic independent from the technology that will be used to implement the system. OMG tries to achieve this goal by specifying mechanisms for standardization of notations and tool interoperability [15].

The MDA approach adopts the Unified Modeling Language (UML) [24] as its system modeling language. MDA exploits the UML extensibility mechanisms, such

R. Meersman, Z. Tari et al. (Eds.): OTM 2006, LNCS 4275, pp. 551 – 568, 2006.

as stereotypes and tagged values, in order to extend it, raising its code generation capacity to a level where it is possible to generate artifacts of any target technology. However, in spite of the fact that the UML notation already includes mechanisms to represent system behavior, such mechanisms are not enough to express, in a complete and precise way, all the execution logics of a given system. By system behavior we mean a set of actions that must be executed to concretely realize a use case, i.e., the description of a given system behavior should refine and complement the corresponding use case. Using current UML techniques, it is not possible to fully automatize the source code generation from UML models. The central problem in specifications of system behavior is that the UML interaction diagrams are scenario-based [8]. A use case can be realized by a set of sequence diagrams, where each one describes each possible scenario. The problem that rises is to process the several diagrams which are related to a use case in a single implementation [27].

To overcome this limitation, in early 2000, influenced by discussions on action semantics theory [22], a working group was set up by the OMG to create a definition of action semantics that was first added to version 1.5 of UML™ in 2003 [24]. This initiative comprised a meta-model to support the formal description of actions and the proposition of associated languages. The goal of these languages is to make systems specification complete and independent from any given target technology. With this goal in mind, many action specification languages have been created such as the: Kabira Action Semantics [13], Object Action Language (OAL) [4] and Action Specification Language (ASL) [34]. By using these languages, all system implementation can be automatically generated since such languages have the same constructors that general purpose programming languages, thus facilitating transformations between them. As MDA approach states, the users of these languages are the system designers. However, we argue that current ASLs are too close to programming languages since they need explicit mechanisms to declare variables, to assign values and to create, destroy and associate objects. Therefore, they can not be assumed as having a high level of abstraction, mainly when considering the MDA goal, which is to be used as a design level language. As a consequence, they are not suitable as specification languages for business and system designers. To deal with these problems, we consider a solution that can bring the system behavior specification closer to the problem definition and, at the same time, is accurate enough to be processed and transformed into executable code.

In this paper, we discuss the use of the Natural MDA language, which was initially presented in [16, 17]. Natural MDA is an action specification language with a high abstraction level that is aligned to MDA objectives. The proposed language aims to raise the systems specification abstraction level, to complement UML and, as a consequence, to reduce the gap between the business domain objects and programming language elements. An application example is presented, exploring the language functionalities by using two associated tools specifically designed to augment MDD tools with Natural MDA language.

Since it complements UML and, at the same time, is directly translated to programming code, the proposed language has the frequently conflicting requirements of keeping a high abstraction level and being machine-processable. To meet such requirements, we consider the hypothesis where controlled natural languages can be used to specify systems and to transform these specifications into executable code.

Despite the existence of many controlled natural languages [6, 28], we have not found any that is aligned to MDA goals.

The next sections cover the following topics: (i) the grammar and the main features of the language; (ii) a brief discussion on the language shortcomings; (iii) the description of a tool developed to process the language; (iv) the description of a tool developed to aid the language usage; (v) the description and analysis of an example of application built to validate the language.

2 Model Driven Architecture

MDA is an OMG initiative whose goal is to provide a greater separation between the system specification and its implementation [15]. MDA approach includes three different system models:

- Computational Independent Model (CIM): this model represents the system independently from the computational model. It focuses on the system requirements. The structural and processing details are hidden or not described. This model is also known as the domain model.
- Platform Independent Model (PIM): this model is the representation of the system functionalities. It focuses on business rules, which are the part of the specification that do not depend on the technology.
- Platform Specific Model (PSM): this model is the representation of the technological details of the system implementation platform.

A key MDA feature is the adoption of transformation mechanisms. Through these mechanisms a high level model can be transformed in a lower level model. For instance, PSMs are generated by transformation mechanisms using as input PIMs and, in the same way, implementation code can be generated from PSMs.

MDA has adopted UML as the language for modeling PIMs and PSMs. By being based on a standard system modeling language, the generated specification is platform independent and can be transformed into a platform specific model.

Besides the existence of these different abstraction levels of models, there are six modeling maturity levels [33], indicating the role that models play on system development process:

- Level 0: there are no specification documents. The success of the project depends exclusively on the software developers.
- Level 1: the specification is textual and written by using natural language. Since natural language is ambiguous, software developers make their decisions based on their own interpretation of the text.
- Level 2: the specification is written in controlled natural language, with some diagrams to explain its structure.
- Level 3: the specification is composed of models, represented through both diagrams and text, with a well defined and specific meaning. The texts in natural language only explain the motivation and the context of the models and complement some details. The models are the most important artifact of the analysis and the design. At this level, software developers still make their own decisions, but such decisions have less influence over the system architecture.

- Level 4: the specification is composed of models with text written in natural language. The models are precise enough to have a direct association with the final implementation, despite being in different abstraction levels. This modeling level is the main goal of MDA. At this level, software developers can not make their own decisions. The models and the implementation keep themselves synchronized. Iterative and incremental development is made easy by the direct transformation between model and code.
- Level 5: the specification is based only in models. Models are precise and detailed enough to allow complete code generation and no interference is needed for this transformation. At this level, the UML code generators are as trustworthy as compilers, therefore there is no need to directly handle the generated code. The languages in which these models are written can be considered to be the next generation of programming languages.

2.1 Action Specification Languages

Object Constraint Language (OCL) [33] was the first step of UML to raise the models expressivity, defining business rules in a clear and unambiguous way. However, OCL is not sufficient to precisely define a system behavior, because it represents only static aspects of the system, and does not have mechanisms to express the occurrence and execution of events and actions. Despite ambiguity problems, textual descriptions are employed to capture those actions and events in UML models. The Precise Action Semantics for UML [21] which incorporates the theory of action semantics in UML 1.5 was proposed to fill this gap.

The main purpose of action languages is to complement the specification of operations with enough details to allow verification of models and to translate them into executable code [23]. OMG states that action languages should follow an industry standard, have a complete processing specification, be verifiable through simulation and generate complete code from UML models. OMG also determines some basic required features. Action languages should be:

- UML compatible
- Executable and complete
- Implementation independent
- At a level of abstraction above implementation

However, OMG does not specify nor recommends any action specification language. Some authors proposed their own action specification languages, like Action Specification Language [34] and Java like Action Language (JAL) [9]. These languages are compliant with the first three features listed above, but they are not at a level of abstraction above implementation. Both languages contain constructors directly related to programming such as: attribute reading and modification, variable declaration, and object manipulation. Another proposal is to extend OCL to include actions [14]. The main concerns of the authors are to keep the result language compatible with UML and to avoid overlapping modeling representations. Thus, the designers can specify dynamic requirements in a declarative way and still at an abstract level. However, these action languages are still very close to programming languages. We believe action languages should not contain specific details of

programming languages if they are intended to reach a separation between the specification and the implementation system.

Although there are some initiatives to implement action semantics [4,9,13,14,34], they are also not at the desired abstraction level. Since functionality specification is a designer role, the definition of a more suitable language for the design phase is necessary.

UML has many ways of representing behavior and there is little consensus on which technique should be used in the context of MDA. In [20] there is a classification of these techniques and a discussion on the strengths and weakness of each one. UML divides the universe of modeling artifacts into structural and behavioral models. A large amount of the infrastructure code for an application can be generated from structural models. A number of tool vendors provide code generators that create code skeletons, but the business logic has to be added by programmers. In case of relatively simple operations, some tools can generate the implementation of an operation from OCL post-conditions. But the dynamics of the system still can not be fully specified in OCL [15]. An alternative to the utilization of current languages is the adoption of controlled natural languages, which will be presented in next sections.

2.2 Controlled Natural Languages

A controlled natural language is a subset of natural languages that can be processed by computers and is expressive enough to allow intuitive use by non experts. Controlled natural languages seek the reduction or elimination of the complexity and ambiguity of natural languages [1]. Despite the contradictions that exist on the term "controlled natural language", because the term "natural" implies that it should not be "controlled" and vice-versa, we chose to adopt it because it is widely used in the literature.

Considering the conceptual differences of the application domain and software development, it is not a trivial task to specify a system. To reduce these differences, some works propose the use of controlled natural language as an application-specific declarative specification language. Thus, these specifications can be automatically transformed into some programming language and, therefore, become executable.

Some authors argue that the use of formal specification languages can eliminate problems related to natural language, such as ambiguity and imprecision. However, due to the necessity of communication and comprehension among the members of the development team, we can not always substitute documents written in natural languages by formal specifications. To improve the quality of specifications without loosing readability, some works propose controlled natural languages with well-defined syntax.

In [19] the use of controlled natural language is suggested to invoke previously developed components. To accomplish this, the components are described by an ontology. This ontology represents the components functionalities, their inputs and outputs and the types of these inputs and outputs. The contribution of this work is the usage of an interlingua, which is an universal language that can be expressed in different human languages [30]. The specifications can be expressed by many natural languages. Despite being targeted to software components reuse, this work is focused

on implementation reuse, and not on modeling reuse. On the other hand, MDA focuses on models.

Considering the need of a requirements specification language that is convenient for domain experts, but is also capable of being mapped to an implementation, in [6] a Two-Level Grammar (TLG) is defined. TLG is an object-oriented requirements specification language, based on natural language, with enough formalism to derive the corresponding implementation. The technique used on the transformation consists on identifying objects and relations on the problem domain based on nouns and verbs between them. This grammar was initially created as a language to specify programming languages. With the appearance of executable models, it became an executable specification language, allowing the transformation of requirements expressed in natural language to a formal specification.

Although TLG allows platform independent specifications, presenting a smaller gap to domain terms, its specifications are only text written in natural language, thus hindering the specification understanding. Our opinion is that a graphical notation, like UML, eases visualization and can be complemented by a natural language grammar, instead of completely replaced by it.

Natural language oriented models are widely used in requirements modeling. This kind of requirements model has to be reinterpreted by system analysts into a more precise representation [18]. Therefore, a semi-automatic transformation to map the requirements models into conceptual object models would be really useful. The author in [18] states that it would be necessary to incorporate linguistic approaches to achieve a better processing of the information.

3 Natural MDA: An Action Specification Language

In Section 2, we presented the main goals of the MDA approach and the main advantages and limitations of action specification languages and controlled natural languages. We realized that by combining the advantages of both controlled natural languages (to lay on an abstraction level above implementation) and action languages (to be UML compatible, executable, complete and platform independent) we can reach the main goal of MDA, which is the complete system modeling, in a precise and technology independent way.

This Section describes Natural MDA language. The design of the proposed language was based on two essential requirements: to provide a suitable abstraction level for the system design phase; and to be processable and complementary to UML (MDA compliant).

The main idea is that the language grammar should define only the valid sentence types; i.e., the sentence composition rules. Terms that are not keywords of the language should be represented as references to model elements (classes, attributes and operations) through their descriptions. We use the UML extension mechanism known as tagged values to associate descriptions to these model elements. These model element descriptions are used to describe the actions for operations.

According to the maturity level classification for modeling languages presented in Section 2, UML models augmented with Natural MDA language is a step further

towards a level 5 language. In the following section, we show the details of the proposed language grammar.

3.1 Action Specification Language Grammar

In Table 1 we introduce the grammar of the Natural MDA language. This grammar builds on the language presented by [28], which is based on end-user programming principles. The main change that we introduced in the grammar consists on replacing the domain ontology by UML model elements; i.e., the object references that are quoted on that work correspond to class and operation references that exist on a UML model. For the sake of simplicity, in this table we hide token declarations.

Table 1. Controlled Natural Language Grammar

```
1   …
2   rule : ((operations DOT) | (sentence) | (after_all) )+;
3   operations: (operation ((COMMA | AND) operation)*) |
4               (show_error_warning) | (return_statement);
5   show_error_warning : SHOW(ERROR | WARNING)ASPAS message ASPAS;
6   return_statement: RETURN alpha_numeric;
7   sentence: (conditional (OTHERWISE COMMA operations DOT)?) |
8             (loop) | (exceptional) ;
9   conditional : IF conditions COMMA
10                ((operations DOT) | loop);
11  loop: FOR EACH item OF THE collection COMMA
12        ((operations DOT) | (conditional )) ;
13  exceptional : WHEN event COMMA operations DOT;
14  after_all : AFTER_ALL alpha_numeric COMMA
15              ((operations DOT) | (sentence));
16  conditions: condition ((AND | OR ) condition)*;
17  condition : operand verb ( (equal | not_equal |
18              greater_or_equal_to | greater_than |
19              lower_or_equal_to | lower_than) ) operand;
20  equal : EQUAL TO ;
21  not_equal : NOT EQUAL TO;
22  greater_or_equal_to : GREATER OR EQUAL TO;
23  greater_than : GREATER THAN;
24  lower_or_equal_to : LOWER OR EQUAL TO;
25  lower_than : LOWER THAN;
26  verb : IS | ARE;
27  alpha_numeric : (CHARACTER | DIGIT)(DIGIT | CHARACTER)* ;
28  collection: alpha_numeric;
29  event: alpha_numeric;
30  operation: alpha_numeric ;
31  message: alpha_numeric ;
32  operand: alpha_numeric ;
```

The grammar main term is “rule”. As it can be seen in line 2, this term consists of a set of operations, followed by a dot, or sentences, or “after_all”. The set of operations, which is called “operations” in our grammar, consists of a sequence of operations, followed by the connector “and” or comma, and another operation; or error or alert (“show_error_warning”) messages, or a return statement. As it can be seen in line 5, an error or alert message is a sequence of the terms: “show”, followed by “error” or “warning”, and finally the message between double quotes.

A sentence can correspond to either a conditional, or iteration, or exceptional expression. A conditional expression is composed of a set of conditions and a set of operations that must be executed whenever the conditions are satisfied. The set of

conditions must be expressed in the form of a stream of conditions, with the connector "and" or "or" between each condition. Each condition is composed of two operands and an operator. Conditional expressions can include a set of operations to be executed whenever none condition is met. Line 7 shows a composition of sentences.

The allowed operators are: "equal", "not_equal", "greater_or_equal_than", "greater_than", "lower_or_equal_than" and "lower_than" (see lines 17-19). The iterations are represented by a collection which must be iterated and a set of operations and conditional expressions to be executed, as shown in lines 11-12. The expression "after all", which is described in lines 14-15, was created to fulfill the need to declare commands to be executed after the scope of the iteration. Event expressions should be used to deal with events, which correspond to exceptions declared in the model.

Up to this point we have described the basic features of the language used to describe actions. Next, we describe how the model elements (classes, attributes and methods) can be referenced within Natural MDA. To reduce the gap between business and model concepts, we used the UML extension mechanism known as tagged value. The designer must set the corresponding tagged value in each model element. This tagged value is used to link Natural MDA elements such as: operations, messages, return_statements, collections, etc. to model elements. Tagged values provide a more readable and natural name for designers to recognize the model elements, working as a mapping between model elements and business terms. The defined tagged values are:

- element.description: it assigns a description to model elements;
- return.description: it assigns a description to the return value of an operation;
- operation.behaviour: it assigns the actions of an operation.

4 Language Supporting Tools

After defining the proposed action language, we developed a set of tools to support it. These tools are the language processor (syntactic and semantic analyzer) and an editor to aid the language use while building action specifications. Fig.1a shows the conventional process of MDA tools, which generates skeleton code from structural models. Fig. 1b shows how our tools interact with existent MDA tools. The language editor does not interact with MDA tools; it only reads the XMI representation (to get the information to describe the action specifications) and saves action specifications in XMI files. On the other hand, the language processor must interact with a MDA tool. In order to generate the application code, the language processor must combine the skeleton code, generated by a MDA tool, with the body implementation of each operation previously specified by the designer using our language editor.

"Another Tool for Language Recognition" (ANTLR) [3] is an open source tool that, from a grammar description, generates a program capable of recognizing sentences written on that grammar. We used this tool to generate the grammar lexical analyzer. On the synthesis phase, we chose String Template [29], which is an open source tool that generates texts using templates. The advantage of using such kind of tool is that we can change the output language by changing only the templates, without modifying the transformation logic. Currently, Java code is generated, but

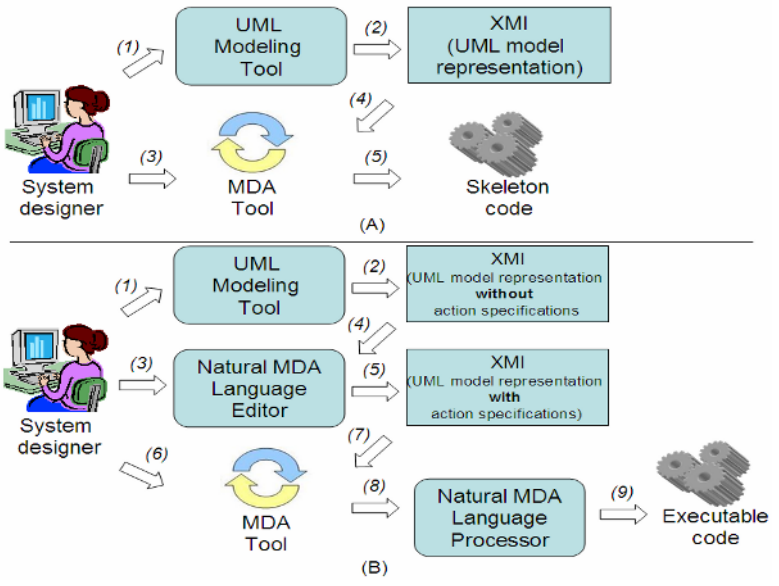


Fig. 1. MDA tool extended with Natural MDA Language Supporting tools

whenever it is needed, we can change the output language by only changing the templates.

Despite the existence of these tools, there is no integration between them capable of applying a transformation (defined by the templates) in a syntactic ANTLR tree. So, we had to implement a syntactic and semantic analyzer. Another reason to implement the semantic analyzer is to verify if the operation descriptions used on the action descriptions were specified on the UML model.

Since our goal is only to generate the operation implementation, we had to integrate our tool with another MDA tool to generate the complete and executable code. As we aim that our tool is independent from a specific MDA tool, we defined a set of interfaces to be used as contracts. We chose AndroMDA [2], an MDA open source tool which is extensible, well documented and widely used to develop our study. For the integration with AndroMDA, we had to create a set of classes to implement those interfaces. These classes work as an adapter. If we want to integrate our tool with another MDA tool, we have only to implement this adapter.

AndroMDA transformations are based on cartridges. A cartridge is a set of templates for a specific platform: Hibernate [12], Enterprise Java Beans (EJB) [10] and others. AndroMDA knows which cartridge must be applied by querying the model elements, stereotypes and tagged values and the transformation configuration for each project. Samples of predefined stereotypes are Entity and Service. Classes stereotyped as Entity can be transformed into Hibernate classes, for example. Classes stereotyped as Service can be transformed into Session Beans, for example. Whenever a class is marked as an Entity, for Hibernate cartridge, AndroMDA generates all artifacts (classes and mapping files) needed for CRUD operations (create, restore, update, delete). Whenever a class is marked as a Service, for EJB cartridge,

AndroMDA only generates the skeleton code and the deployment descriptor files. The skeleton code generated by AndroMDA is the point where we integrate our tool with AndroMDA. To implement this integration, we used one of the extension mechanisms of AndroMDA known as metafacades.

4.1 Syntactic and Semantic Analyzer and Synthesis

We implemented a parser that reads the action specification and builds a corresponding expression tree. Afterwards, we developed a transformer that navigates through this expression tree and translates it into code. In order to manage variable creation and scope during the translation phase, the transformer includes a context manager. The context manager stores the variables of the current scope. The translation context is always initialized with variables that represent each operation parameter. Then, the translation context is applied to the expression tree root node which in its turn applies the context to its children, and so on. Each node uses the translation context to discover which variables will be created or reused to generate the corresponding code.

These translation contexts implement lexical scoping - whenever a complex expression receives a context, it creates a new context based on the received context, and it propagates the newly created context to its children. After the translation of this complex expression, this new context is discarded. This behavior simulates the concept that variables created inside iteration can not be accessed outside iteration scope.

From action specifications, the parser generates an expression tree containing a logical sequence of expressions according to the defined grammar. These expressions consist of model operations invocations (method call) and control structures (conditional, iterations and event calls). After the expression tree is built, the transformer starts an in-depth navigation through it, where each expression is requested to update the translation context according to its type.

When the expression is a method call, the model is queried for a method that has the corresponding description. If the method is not found, an error is shown. When the method is found, the translation context is analyzed to discover if there is a class instance that owns the operation. If it exists, this instance is selected. Otherwise, it is checked whether the method is static or not. If it is static, the class is not instantiated to make the call. If it is not static, it is checked if the class has a service or entity stereotype. If it is a service class, its business interface is selected, and if it is an entity class, the corresponding data access class is created. Next, it is verified if the method has parameters and a return value. If it has parameters, the translation context is searched for variables that have the same description (more recently used variables have preference). If none variable is found, it is checked if the variables contain attributes or associations with the desired description (associations are recursively navigated, and stop conditions are defined to avoid an infinite loop). After all, if a variable is not found, a search by type is performed. The type search is analogous to the description search. When the method has a return value, it is added to the variable context, so that it can be used during the translation of the following expressions.

When the expression is an event call, the model is searched for an exception that has the corresponding description. If this exception is found, it is used to handle the event. If not, an error is shown.

When the expression is a conditional call, it is verified if there is more than one condition. Each condition can be a call expression or a logical expression. If the condition is a call expression, it is verified if the method exists in the model. If it exists, it is processed. Otherwise, an error is shown. If the condition is a logical expression, it is verified if the operands already exist in the translation context. If the operands are not found, an error is shown.

When the expression is an iterative call, it is verified if the collection to be iterated already exists in the translation context. If it exists, the type of each element of the collection is found by searching for dependencies of the operation at the model. Whether the collection is not found in the translation context or dependencies of method are not found in the model, an error is shown.

4.2 Language Editor

To facilitate the use of Natural MDA by system designers, a tool was developed at our research group [7]. The goal of this tool is to guide the designer to use, in an integrated manner, both the action specification language and a given target UML model. The tool takes UML models represented in XML Metadata Interchange (XMI) as its input. It reads the model and shows the model elements in a similar way to modeling tools. In addition, the tool executes a consistency check, indicating the valid sentences and suggesting fixes according to the grammar and the UML model. The users also have an option to save the descriptions that were specified. When this option is selected, each action description is saved back to the model on the tagged value "operation.behaviour" of the corresponding operation in the model.

Another feature is that designers can visualize business logic of action specifications graphically. Simple graphic elements for decisions, for structuring logic, and executing actions are used. Therefore, despite the textual nature of our language, through its editor, it can be visually presented.

5 Development Process

The usage of MDA approaches to build systems requires some changes in the software development process [5]. The development process is also affected by the use of Natural MDA language, because it requires the addition of new activities and the elimination of others in comparison to traditional development processes. In order to illustrate how Natural MDA language influences on system development process, we will see how Rational Unified Process (RUP) [26] workflows are affected by the use of Natural MDA.

RUP framework consists of phases and disciplines. Phases represent aspects of the process life cycle: Inception, Elaboration, Construction and Transition. Disciplines represent a logical grouping of activities: Requirements, Analysis and Design, Implementation, Test, Change and Configuration Management, Project Management and Environment Configuration. Each activity has a corresponding workflow. Next, we show the changes imposed by the use of our language in these workflows.

The use of Natural MDA language does not incur any changes in the requirements workflow, because such workflow is focused on capturing system requirements. At

the analysis and design workflow, we identified that model driven development along Natural MDA language usage bring some advantages. This workflow comprises implementing the system architectural proof of concept, which includes the most critical architectural decisions. Since it is possible to fully generate the system code from models augmented with Natural MDA specifications, the architectural proof of concept can be rapidly built, thus allowing testing different model transformations in order to choose the most appropriate ones to be applied to the critical system requirements. Therefore, we consider necessary to include a new activity whose objective is to define the most suitable transformations according to system needs. An additional change is that the designer must associate descriptions (through tagged values) to attributes and operations when creating the classes. Moreover, during use case design, the designer may describe the behavior using Natural MDA language or UML interaction diagrams. By using MDD, the database design phase is simplified, consisting in refining the schema generated from the domain model.

At the implementation workflow, the activity "Implement design elements" is considerably affected because a great part of the implementation is generated from the action specifications defined in the model. The code revision activity is also affected, since in the MDA approach it is more proper to revise the specification instead of revising the code. Furthermore, great part of code implementation does not need revision, because the transformations were already validated during the Elaboration phase.

At the test workflow, the activities remain unchanged because the current version of the language supporting tools does not generate automated tests. The use of Natural MDA language does not impact on the activities included in the change and configuration management workflow. At the project management workflow, the only change is that the team allocation activity must include a new role that is responsible for model transformations.

As the goal of the environment configuration workflow is to right-size the software development process according to the specific needs of the project and to provide a relevant and accessible process description to the members of the project, the Natural MDA language usage does not impose any changes to such workflow activities.

6 Application Example

In this section, we present an application example to illustrate the use of the Natural MDA language. The goal of this study was to analyze the expressiveness of the language, the level of difficulty on its use and its adherence to model driven development cycle, rather than measuring productivity gains derived from the language use. The complete observational study can be found in [17].

The following application example is an excerpt of a software system for academic management of the Federal University of Rio de Janeiro, whose goal is to manage administrative and academic activities. In this example, we will see only the use cases for class registration, grade registration and enrollment registration. Since our goal with the application example is to demonstrate the use of Natural MDA language, the activities related to requirement identification, schedule elaboration and project planning were not executed. In the Inception phase, we used the requirements already defined by the actual development team of the Academic System. Next, the candidate

architecture was defined and the proof of concept was implemented. We consider that the interface layer activates the components of the service layer, which in turn activate the domain layer. In the scope of the domain layer, we identified that a course has several subjects. A subject may have other subjects as requirements and they are offered through classes. A class is associated to a classroom and in each segment, the students must enroll to subjects.

The next step was to detail the elected use cases. Fig. 2 shows the initial service modeling derived from the use case analysis. The method "addEnrollmentOrder" of

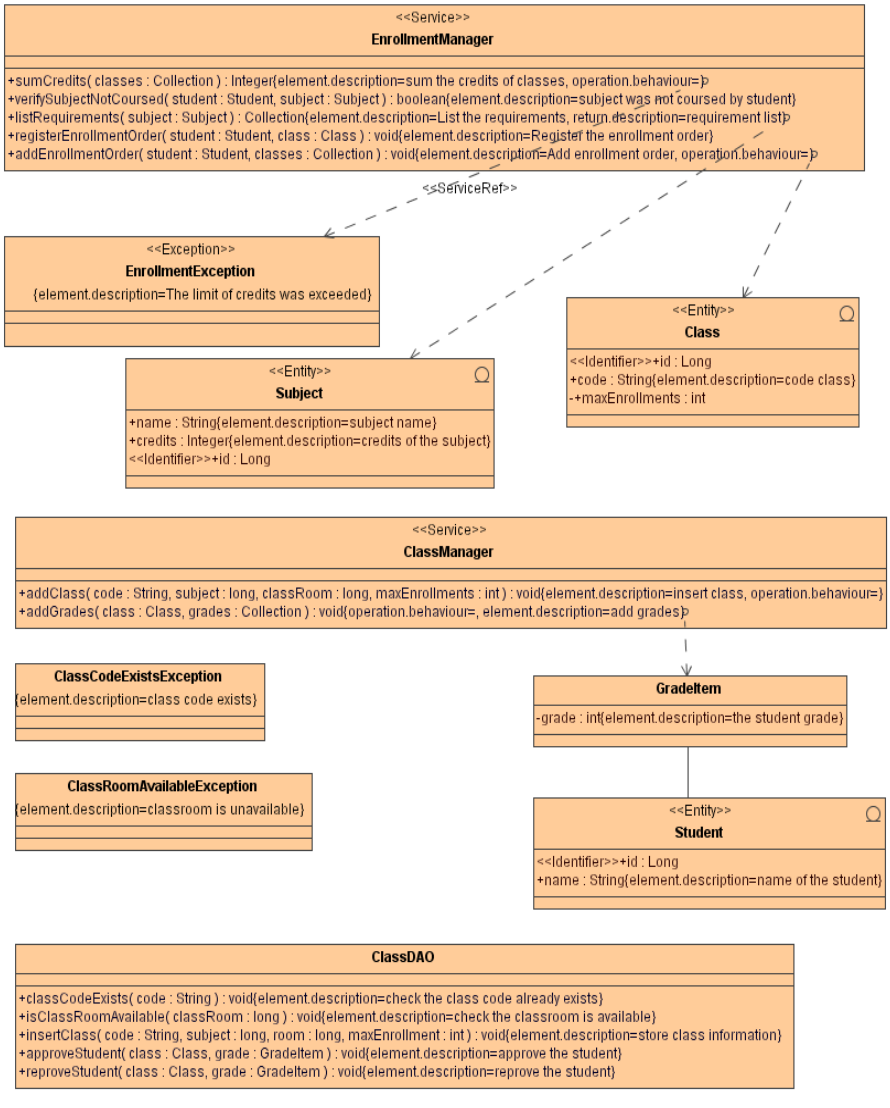


Fig. 2. Academic System service class diagram

the component EnrollmentManager realizes the use case of enrollment registration. The methods “addClass” and “addGrade” of the component ClassManager realize the use cases of class registration and grade registration. In order to specify the actions of these methods, other methods were defined to divide responsibilities. Note that all methods have an associated tagged value named “element.description”. The values defined in these tags are used in the action specifications.

Next, the use cases were refined using Natural MDA language and the language editor. In the editor, the user selects a model in order to describe the behavior of each class operation belonging to this model. After selecting the model, the tool shows all its classes grouped by package, in the same way they are grouped in UML modeling tools. When a class is expanded, the tool shows its operations, and when an operation is expanded, input parameters (arrow pointing right) and return value (arrow pointing left) are shown. When double clicking an operation, the corresponding action descriptions are opened in the top-right side. The tool provides auto-complete facilities to edit actions. When selecting the menu option “Translate”, the tool shows the corresponding source code for the current action description in the bottom-right side. When selecting the menu option “Validate”, the tool validates the actions according to the grammar definition and the UML model, and it shows the error or success messages on the bottom-right side. By selecting the menu option “Diagram”, the tool shows a flow chart that graphically represents the current operation description. Table 2 shows the action specifications for “addEnrollmentOrder” operation based on the model elements and the Natural MDA language grammar. To highlight both the grammar elements and the references to UML model elements, we used words in bold to refer to model elements, and words in italic to refer to grammar elements.

Table 2. Description of addEnrollmentOrder operation

```
1.  Sum the credits of classes.
2.  For each class of the class list, list the requirements and
3.  for each requirement of the requirement list, if subject was not
4.  coursed by student, show error 'Requirement not coursed'.
5.  After all requirement list, register the enrollment order.
6.  When the limit of credits was exceeded, show error 'The limit of
7.  credits was exceeded'.
```

In line 1 a method is invoked to verify if the limit of credits allowed for a term was exceeded. Next, we verify if the student has taken all the requisites of the selected subject and if the subject was previously taken. In case of success, the enrollment order is registered. When the limit of credits is exceeded, this event is handled as described in lines 6-7.

Table 3 shows the implementation generated from the action specification in Table 2. The steps executed during the translation are:

- Variable context initialization: the input parameters of the method to be translated are added to the context;
- Mapping between actions and operations: for each action used in the specification the corresponding operation that has the same description is searched. If the operation has an output parameter, it is added to the context;

Table 3. Generated code for addEnrollmentOrder operation

```
public void addEnrollment(br.ufrj.academic.domain.Student student,
        java.util.Collection classes) {
    try {java.lang.Integer var1 = this.sumCredits(classes);
        for (java.util.Iterator var2 = classes.iterator();
             var2.hasNext();) {
            br.ufrj.academic.domain.Class var3 =
                        (br.ufrj.academic.domain.Class) var2.next();
            java.util.Collection var4 =
                         this.listRequirements(var3.getSubject());
            for (java.util.Iterator var5 = var4.iterator();
                   var5.hasNext();) {
                br.ufrj.academic.domain.Subject var6 =
                       (br.ufrj.academic.domain.Subject) var5.next();
                java.lang.String var7 =
br.ufrj.academic.domain.Subject.findSubjectById(var6.getId());
                if (this.verifySubjectNotCoursed(student, var7)) {
                    logger.error("requirement not coursed"); } }
            this.registerEnrollment(student, var3);   }
    } catch (br.ufrj.academic.service.EnrollmentException var2) {
        logger.error("the limit of credits was exceeded");  }}
```

- Searching inputs and output of each operation: when it is necessary to call an operation that has input/output parameters, such parameters are searched by their type and description in the context;
- Dealing with iterations: the context is queried for the variable with the same description of the collection to be iterated. Then, for each element of the collection, a new variable with the corresponding type is added to the context;
- Dealing with events: an exception with the same description as the event described in action specification is searched in the model and it is used in the translation.

In order to ease the visualization of the action specification, the editor allows presenting the corresponding flowchart. It shows the iterations, conditions and operation calls in the same order as they occur.

The designer can refine the use case using the Natural MDA language, sequence diagrams or collaboration diagrams.

6.1 Evaluation

The example previously described highlights some advantages of using MDA augmented by Natural MDA language. First, it is noted that the main flow (lines 1-5) of the use case is kept apart from the alternative flow (lines 6-7), thus making the business rules more readable.

The second advantage is the reduction of the number of characters that must be written to represent the systems behavior compared to the number of code characters that are generated. Table 4 shows the number of characters in the specification and the implementation. Another important advantage is the improvement of readability due to the reduction of constructors directly related to programming languages. This is a major advantage since great portion of system code can be generated without the need of knowledge in any specific programming language. Considering the increasing complexity of current programming languages this feature of our proposal provides significant benefits in the system development. Moreover, this feature also facilitates

Table 4. Percentual of characters reduction

	Enrollment registration
Specification characters	300
Implementation characters	803
Percentual of characters (spec/impl)	37%

the interaction among the development team since team members without programming skills can understand the whole system specification.

7 Conclusion

The majority of current action specification languages for Model Driven Development (MDD) lack a suitable abstraction degree to leverage the task of system designers. We argue that an MDD action specification language should be closer to the abstraction level of system models and more independent from the underlying implementation syntax. To tackle this problem, we defined an action specification language based on controlled natural language. Besides having a higher abstraction level, the proposed language complements UML, is platform independent and can be automatically transformed into executable code. The proposed language seeks to increase the abstraction level while keeping the capability of being machine-processable. This goal is achieved by adopting an approach in which low level instructions closely related to implementation are replaced by of higher level constructions.

The proposed language is fully aligned to the MDA goals. The adoption of this language can foster the MDD process by reducing the gap between the problem domain and the implementation language, facilitating the communication between clients and designers. Moreover, since the proposed language is a natural controlled language it does not have the drawbacks of totally visual languages [32]. Furthermore, it seamlessly integrates with UML diagrams, augmenting their expressivity power.

Since Natural MDA is basically text-based, one can argue that a visual-based ASL would be more aligned to UML goals. We believe that as complexity increases, a diagrammatic representation does not scale well. Even UML, which is a visual modeling language, adopts textual languages like OCL and Action Specification Language (ASL) to describe restrictions and actions. Additional drawbacks of totally visual languages are discussed by the author of the programming language Visula [32]. Although Visula is a visual language, its author recognizes some advantages of textual languages: (i) they do not require a special notation; (ii) they do not require special editors; (iii) visual programming languages are hard to discuss inside discussion groups.

It is worthwhile mentioning that Natural MDA language can not be fully employed in all possible application domains. For instance, it is not suitable for complex or expert systems, since such kind of systems requires notations more appropriate than natural languages. We are now working on three main directions: (i) integration with an execution environment, in which the artifacts can be executed, debugged and

validated; (ii) grammar extension to include the usage of ellipsis and other linguistic resources to make the language closer to natural languages; and (iii) formal validation about the equivalence between inputs (specification) and outputs (implementation) of the transformation process.

Acknowledgements

This work was partially supported by the Brazilian funding agency CNPq.

References

1. Allen, J. (1995) "Natural Language Understanding", The Benjamin/Cummings Publishing Company Inc, ISBN: 0-8053-0334-0.
2. AndroMDA (2004) "AndroMDA: Getting started", http://www.andromda.org.
3. ANTLR (2004) "Another Tool for Language Recognition", http://www.antlr.org/.
4. BridgePoint (2005) "Object Action Language", http://www.acceleratedtechnology.com.
5. Brown, A. W.; Conallen, J. (2005) "An introduction to model-driven architecture – How MDA affects the iterative development process", http://www-106.ibm.com/ developerworks/ rational/library/may05/brown.
6. Bryant, B. and Lee, B. S. (2002) "Two-Level Grammar as an Object-Oriented Requirements Specification Language", XXXV Hawaii International Conference on System Sciences, IEEE Computer Society, Volume 9, p. 280. ISBN:0-7695-1435-9.
7. Cianni, N. M. and Cabeço, T. (2006) "Editor for Action Specifications in Controlled Natural Language", http://dataware.nce.ufrj.br:8080/dataware_en.
8. Damn, W., Harel, D. (2001) "LSCs: Breathing Life into Message Sequence Charts", Formal Methods in System Design, V. 19, p. 45-80, July, 2001. ISSN:0925-9856.
9. Dinh-Trong, T. (2005) "JAL: Java like Action Language", Specification 1.1, Department of Computer Science, Colorado State University, October.
10. EJB (2003) "Enterprise Java Beans", http://java.sun.com/products/ejb.
11. Fowler, M., Beck, K., Brant, J., Opdyke, W., Roberts, D. (1999) "Refactoring: Improving the Design of Existing Code", Addison-Wesley, ISBN 0201485672.
12. Hibernate (2006) "Hibernate", http://www.hibernate.org/.
13. Kabira (2002) "Kabira Action Semantics", http://www.kabira.com.
14. Kleppe, A. and Warmer, J. (2000) "Extending OCL to include actions", International Conference on the Unified Modeling Language, p. 440-450.
15. Kleppe, A., Warmer, J. and Bast, W. (2002) "MDA Explained: The Model Driven Architecture: Practice and Promisse", Addison-Wesley.
16. Leal, L. N., Pires, P. F. and Campos, M. L. M. (2006) "An Action Specification Language based on Controlled Natural Language", Proceedings of XX Brazilian Symposium on Programming Languages, Rio de Janeiro, Brazil (in portuguese).
17. Leal, L. N. (2006) "Natural MDA: An action specification language based on controlled natural language", Ms Thesis, http://dataware.nce.ufrj.br:8080/ dataware_en/public/thesi/ ObservationalStudy, UFRJ, Brazil.
18. Leonardi, M. C. and Mauco, M. V. (2004) "Integrating Natural Language Oriented Requirements Models into MDA", Workshop on Requirements Engineering, Argentina, http://www.sigmod.org/dblp/dp/conf/wer/wer2004.html.

19. Linhalis, F. and Moreira, D. A. (2005) “Execution of Imperative Natural Language Requisitions Based on UNL Interlingua and Software Components”, International Conference on Enterprise Information Systems (ICEIS).
20. McNeile, A. and Simons, N. (2004) “Methods of behaviour modeling: a commentary on behaviour modeling techniques for MDA”, http://wwwmetamaxim.com/download/documents/Methods.pdf.
21. Mellor, S. J. et al. (1998) “Software-Platform-Independent, Precise Action Specifications for UML”, Proceedings of UML´98.
22. Mosses, P. D. (1996) “Theory and Practice of Action Semantics”, BRICS Report Series. RS-96-53. ISSN 0909-0878. December.
23. OMG (1999) “Action Semantics for the UML – Request for Proposal”, http://www.omg.org/docs/ad/98-11-01.pdf.
24. OMG, Action Semantics Consortium (2001) “Action semantics for the UML”, http://www.omg.org/docs/ad/2001-03-01.
25. Pender, T. (2003) “UML Bible”, Wiley Publishers, ISBN: 0-7645-2604-9.
26. Rational (2006) “Rational Unified Process V7.0 Evaluation”, http://www-128.ibm.com/developerworks/downloads/r/rup.
27. Selonen, P., Systa, T., Koskimies, K. (2001) “Generating Structured Implementation Schemes from UML Sequence Diagrams”, Proceedings of TOOLS USA, Santa Barbara, California, USA, July-August (p. 317-328). IEEE Computer Society.
28. Silva, S. R. P. and Souza, C. S. (2002) “The Definition of an End-User Programming Language for Extensible Applications”, Proceedings of V Symposium on Human Factors in Computer Systems, Fortaleza, Brazil. Volume 1, p. 72-83.
29. String Template (2005) “String Template”, http://www.stringtemplate.org/.
30. Ushida, H. and Zhu, M. (2001) “The Universal Networking Language beyond Machine Translation”, International Symposium on Language and Cyberspace, September.
31. Visual Rules (2006) “Visual Rules”, http://www.visual-rules.de.
32. Visula (2004) “The Visula Programming Language”, http://visula.org.
33. Warmer, J. and Kleppe, A. (2003) “The Object Constraint Language - Getting Your Models Ready for MDA”, Addison Wesley. ISBN: 0-321-17936-6.
34. Wilkie, I. et al. (2001) “ASL Manual”, Kennedy Carter Ltd, http://www.kc.com.

Managing Distributed Collaboration in a Peer-to-Peer Network

Michael Higgins, Stuart Roth, Jeff Senn,
Peter Lucas, and Dominic Widdows

MAYA Design Inc.*
{higgins, roth, senn, lucas, widdows}@maya.com

Abstract. Shared mutable information objects called u-forms provide an attractive foundation on which to build collaborative systems. As we scale up such systems from small fully-connected workgroups to large, highly distributed, and partially disconnected groups, we have found that peer-to-peer technology and optimistic replication strategies provide a cost-effective mechanism for maintaining good performance. Unfortunately, such systems present well-known coordination and consistency problems. This paper discusses strategies for addressing those difficulties at different levels of the system design, focusing on providing solutions in the information architecture rather than at the infrastructure layer. Addressing problems at this higher layer allows greater freedom in design, and simplifies moving from one infrastructural base to another as technology evolves. Our primary strategy is to enable robust decentralized and asynchronous collaboration while designing architectures that do not rely on two users writing to the same u-form at the same time in different venues. Techniques are provided for simple messaging, collaborative maintenance of collections, indexing supporting rich query, and stand-off annotation and elaboration of third-party datasets. We outline the application of these techniques in a working collaborative system.

1 Introduction

This paper describes a variety of structures that enable robust peer-to-peer collaboration on many semantic levels. The strategy we pursue is to create information architectures that allow widespread replication but minimize the danger of conflicted data. This pattern pervades many of our techniques, from a simple asynchronous messaging protocol to the ability of many publishers to contribute content about the same phenomena.

For some time, MAYA Design and our collaborators have been building powerful collaborative applications by allowing users to share and modify mutable information objects called *u-forms* [1]. This sort of "shared memory" approach to managing collaboration is both elegant and powerful. As the breadth and reach of our systems has expanded, we have encountered a variety of challenges, both obvious and subtle. In brief:

* This work was supported by Defense Advanced Research Projects Agency grant SB031-008 for the Cluster-Based Repositories and Analysis project.

R. Meersman, Z. Tari et al. (Eds.): OTM 2006, LNCS 4275, pp. 569–586, 2006.

- Systems with many users generate large amounts of storage and retrieval cost.
- Systems with many users are rarely completely connected. Users are mobile: they come and go, and need information even when they have little or no network connectivity.
- Systems with many users generate large amounts of disagreement. These disagreements range from mismatches between versions of specific information objects to high-level differences of opinion on certain topics. Many such disagreements cannot and should not be resolved automatically.

We have found optimistic peer-to-peer replication to be a useful tool for attacking the first and second problems. Such a replication model permits a very scalable infrastructure, does not require a major datacenter investment, and tolerates disconnected and poorly connected users [2]. The difficulty of managing an optimistically replicated system is well-understood; it was most famously articulated in [3].

However, this model does nothing to address the third problem, and, if anything, exacerbates it by allowing concurrent and inconsistent updates to the system. Book-keeping to detect such concurrency problems can become significant, and the management of trust and security in the system is complicated. While good work has been done on managing and automatically resolving conflicts (see e.g., [4]), those who have worked hardest on this topic acknowledge how difficult it is to provide automatic conflict resolution in general, because the semantics of conflict resolution criteria are always heavily application dependent, and may depend on several replicated objects, not just one.

Our experiences building working applications and a shared *Information Commons* [5] collaborative space on top of a particular replication system (called *Shepherds*) has led us to evolve techniques to minimize and in some cases avoid these difficulties. Instead of trying to create rules for resolving all conflicts, we have taken a quite different approach, relying on careful information architecture to enable direct collaboration while minimizing the danger that two users will concurrently update the same replicated u-form.

It is our belief that the assumptions that our system depends upon are weak enough that our strategies will be applicable to other systems. There are many exciting recent developments in peer-to-peer systems that could support such collaboration strategies. Important systems include Chord [6], Pastry [7], and OceanStore [8]. Chord and Pastry supply only lookup and routing technology; OceanStore is a more complete system providing replication and object storage. Other higher-level systems include CFS (built on Chord) [9] and PAST (built on Pastry) [10]. Several of these systems use cryptographic hashes to identify immutable data as do we.

This paper describes the Shepherds system, the relevant engineering and design assumptions, and some interesting applications and collaborative structures. The paper is organized as follows. Section 2 discusses u-forms and the collaboration model they enable, as well as a brief survey of systems we have developed using this model. Section 3 discusses engineering and design issues, including

the particular choices and assumptions that our system makes, and the costs and trade-offs entailed. This discussion includes details about conflict detection, shepherdable indexes, trust management and digital signatures, and some information about our underlying replication and storage infrastructure, elaborating on the problem areas raised in this introduction. Section 4 describes how we layer higher level data structures upon this foundation to support messaging, shared maintenance of information collections, and collaborative publication and annotation, without running afoul of the pitfalls above. In addition, by layering our solutions at a higher level of the system, we give ourselves the freedom to more readily change infrastructure technologies. Finally, Section 5 ties these techniques together by briefly describing a *distributed collaborative geographic information system* that makes use of every layer of the system.

We conclude with a fundamental observation: optimistic replication systems scale well and provide a wonderful opportunity for very large-scale collaboration. But the guarantees provided by such systems are necessarily weak. Instead of trying to ensure that conflicting updates to the same u-form are resolved before users notice, we try to create information architectures that minimize the possibility of such conflicts occuring in the first place — while still empowering users to collaborate in real time and to make contributions about definitive data. Clever layering in the design and engineering of the system allow us to support very powerful applications without burdening the entire system with the maintenance of low-level guarantees.

2 U-Forms and the Visage Collaboration Model

All well-designed replication and collaboration systems depend on a fundamental unit of storage and replication, such as a file, a data table, or an individual data record. In Visage [11] and all its descendents, including our Shepherds system, this unit is called a *u-form* [12,1]. A u-form is an extensible bundle of attribute-value pairs identified by a universally unique identifier or UUID. U-forms are mutable (though sometimes changes to a u-form can be recognized as evidence of unauthorized tampering and rejected, as discussed in section 3.3). Attribute names are unique within a u-form. Attribute values can be any data: lists, dictionaries, binary blobs, and, of course, UUIDs of other u-forms. A UUID appearing as the value of an attribute is called a *relation*[1] from one u-form to another (sometimes thought of as a pointer or a reference). The ability to store a UUID as a value in a u-form is important: it allows us to construct complex structures of u-forms by reference. A simple example might be a u-form whose `members` attribute is a list of UUIDs. Such a u-form is called a collection. We

[1] This use of the word "relation" carries the traditional philosophical meaning of a relation between two *objects* (c.f. Aristotle, *Categories* Ch. 7) rather than the mathematical meaning of a relation between two *sets*, the meaning used in relational database systems. These uses are closely related: a relation between sets can be thought of as a *subset* of a Cartesian product, while a relation between objects is an *element* of a Cartesian product.

will see more sophisticated examples exploiting u-form relations throughout the paper.

The name u-form is an homage to Michael Dertouzos' e-form [13]. UUIDs are used quite commonly in many areas of computer science, particularly distributed systems. One contemporary reference relating UUIDs and the Web's URN system is [14]. A good overview of u-forms is found in [1]. A formal resemblance exists between the (UUID, attribute, value) triple in the u-form, the (URI, predicate, object) triple of RDF [15]. The u-form is an abstract datatype, and u-forms can be serialized using many formats including XML, though our implementations commonly use a recursive bytecode format called the Visage Standard Message Format, which is considerably more efficient than XML in practice.

Although u-forms *per se* are completely schemaless, schemata can be layered on top of the basic u-forms by including relations to special *role* u-forms [16].[2] Roles are u-forms that reserve the use of particular attributes and explain their intended interpretation. Storing relations to roles inside u-forms helps in making u-forms self-describing and introspective, which contributes to the effectiveness of replication and enables a single u-form to be a useful member of many heterogeneous datasets.

Updates to u-forms are simply changes to the attribute-value set. Changes to a single u-form can be made atomically in the local venue. The entire state of a

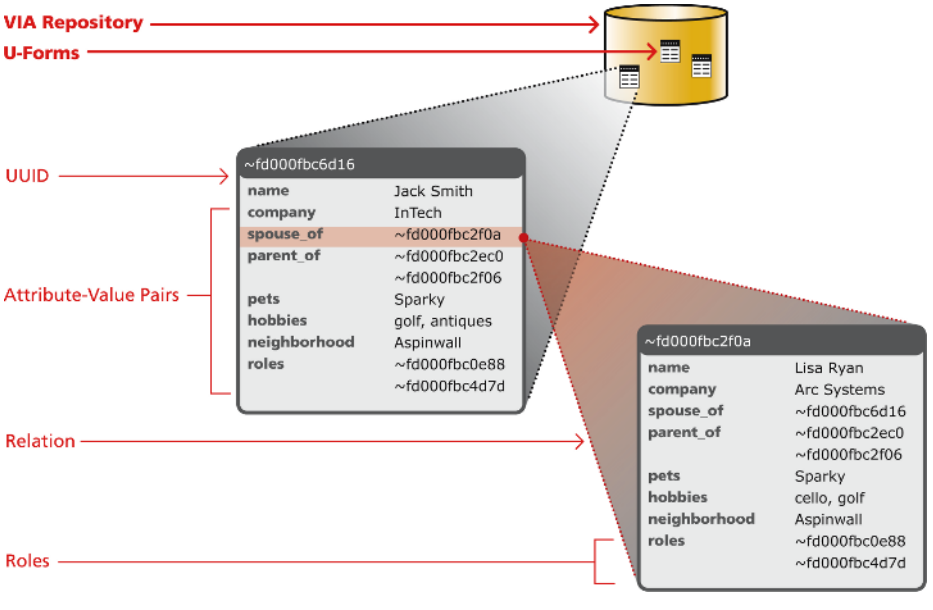


Fig. 1. U-forms and several related aspects of the system, including attributes, values, relations between u-forms, roles, and the VIA repository

[2] This use of a single syntax to describe both data and metadata is reminiscent of XML Schema's use of XML as its definition language. See [17].

u-form is replicated when replication occurs. Because users may be disconnected for a long period, we replicate only the state of a u-form, not a log of all operations on that u-form (which could be quite large).

U-forms support collaboration because they are mutable but have universal identity, so the underlying replication system allows a user to "subscribe to" a u-form: any changes are then propagated to that user. In this way multiple users can be interested in a shared set of u-forms. As they modify this shared set, the u-forms act as a common collaborative space and all the interested parties observe the evolution of that space.

2.1 A Brief History of the U-Form Style

It is useful at this point to give a quick overview of some of the systems we have built over the years using u-forms. This demonstrates the utility of a simple, sound foundation, and will give us a set of concrete touchstones to refer to during the development of some of the more abstract ideas later in the paper.

Visage and Visage2. We first used this collaborative style in an information visualization system called Visage (see [11] and [18]), so we have come to think of it as the "Visage collaboration model." Visage pioneered an interaction model called information-centricity; the idea was to simultaneously provide:

- An easy-to-use direct-manipulation user interface.
- Fine-grained access to small units of information (scraps of text and rows of tables, not just whole documents).
- Data strongly separated from presentation to encourage multiple visualizations of the same data.
- Easy collaboration despite the fact that different users might prefer very different visualizations of the same data.

This last feature was added for Visage2, and cemented the definition of u-form that we use today. In particular, the fact that we use UUIDs to identify data objects (rather than locally scoped primary keys or location-specific URLs) enabled us to envision much more flexible collaboration scenarios [19]. Our commitment to the information visualization application domain made it critical that collaboration was mediated through changes to the data space, not simple sharing of the user interface (see Figure 2 and further discussion in [20]).

Visage2's implementation depended upon client-server technology, not peer-to-peer technology. Therefore replication was limited and controlled. It is the increase in scale, and corresponding relaxation of our controls on replication, that has led to much of the present work.

CoMotion. Success in the laboratory and in limited deployments of the Visage system encouraged us to commercialize the technology. In 1998 we spun off a company called MAYA Viz to develop a commercial product. The framework they created is called CoMotion. Products built on top of CoMotion are used by the U.S. Military for command, control, and logistics applications [21]. CoMotion applications are also used in the medical and energy industries. MAYA Viz was acquired by General Dynamics in 2005 and is now GD Viz [22].

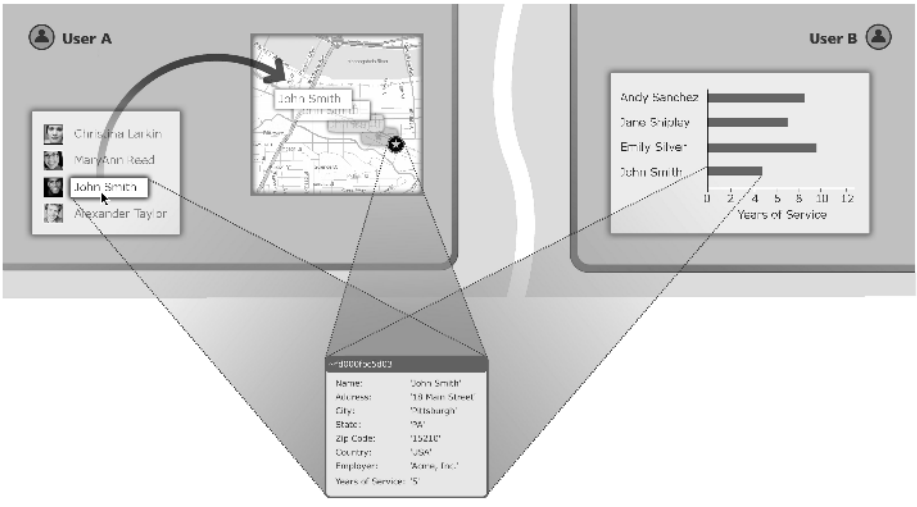


Fig. 2. Polymorphic visualization of a u-form

The Geobrowser. Meanwhile, we continued our research and became interested in very large scale, loosely coupled systems. This pushed us toward peer-to-peer replication as a scheme that can scale much better than client-server systems.

The Geobrowser [23] began as a collaborative GIS system built on our Shepherds peer-to-peer system, but has been evolving toward a more general information browsing and publishing system. The Geobrowser is discussed more thoroughly in Section 5.

The Community Directory and Buskarma. U-forms are not limited to use in exotic information management domains. We have built a variety of useful web applications on top of the Shepherds architecture. These include the Community Directory family of websites [24,25], winners of the Exemplary Systems in Government Award from URISA, an international organization of government information professionals. Another application is Buskarma, a public transportation website for the Pittsburgh area [26]. The underlying peer-to-peer nature of these sites is not very visible, since websites require a stable server to function, but the webservers themselves are connected through the Shepherds system to the same shared information space that the Geobrowser uses. This information space is called the Information Commons [5].

2.2 Some General Observations About U-Form Usages

U-forms are the unit of information in our system, but this same unit is often exploited for a variety of purposes, including:

– Conceptual units, in 1-1 correspondence with objects in the world. This perspective is about semantics and the world.

- Collaborational units (e.g., data that users share and update). This perspective is about interaction design.
- Trust units, for digital signatures and encryption. This perspective is about social cooperation and trust.
- Basic data storage and transport units. This perspective is about engineering and system efficiency.

Each of these perspectives will be important as we explore collaboration techniques, and there are often trade-offs among them. For example, computing digital signatures imposes a non-trivial computational cost per u-form, so there are sometimes pressures to put lots of related data from the same publisher into a single u-form. On the other hand, an index to a dataset is often a single conceptual unit, but for effective use of resources, and to serve the purposes of a variety of users with different interests, this conceptual unit needs to be broken up across several related u-forms.

3 Design and Engineering Issues and Background

This section gives an overview of three basic computer science areas that are necessary ingredients for our collaboration architectures. These are the shepherds (replications agents) themselves, shepherdable indexes, and digital security issues. We do not intend to provide a detailed description of all aspects of the engineering of the Shepherds system. It would be beyond the scope of this paper, and would distract from the general applicability of the techniques described later. For more information about the Shepherds system, consult [27]. It is nonetheless important to give some sketch of how the Shepherds system works, in order to make clear the kinds of guarantees we require from the low level portions of the system.

One major theme of this section is that while our system allows u-forms to be updated anywhere in principle, in practice this is problematic for a variety of reasons. The techniques described later in the paper will show how we can perform collaboration while attempting to minimize the number of venues in which a given u-form is updated.

3.1 Shepherd Agents, Version Vectors, and Conflict Recognition

Shepherds is a large-scale peer-to-peer system that uses optimistic replication to facilitate collaboration through shared u-forms. Let us quickly examine some facets of the system.

The system contains a potentially very large number of venues. There is an extremely large number of u-forms; no venue is likely to contain a significant fraction of all u-forms. U-forms can be created at will by any user in any venue. U-forms can, in general, be updated in any venue (but we will discuss practical restrictions). The system uses peer-to-peer agents called *shepherds* to replicate u-forms amongst the venues. The shepherd agents are guided by application-specific business rules, the specification of which are the job of the designers of a particular system.

The precise details of the replication schemes used are beyond the scope of this paper, and, indeed, have varied significantly over the history of the system. One of the strengths of the techniques described here is that they do not depend strongly on the underlying replication schemes. This means that as new peer-to-peer techniques arise we can use them without major application-level changes.

The essential characteristics of the replication scheme are these:

- It is optimistic. That is, it does not defer replication or updates in an attempt to guarantee consistency. Instead, it replicates under the optimistic assumption that inconsistent updates are rare and can be detected and repaired when they are detected.
- If a user has subscribed to updates on a given u-form, she will eventually receive all relevant updates (though the order and timeliness of those updates is not guaranteed).

We use *version vectors* to detect concurrent updates [28]. A version vector is a list of counters: one counter is kept for each venue in which a given u-form has been modified. A venue's counter for the u-form is incremented when the u-form is updated in that venue. Therefore the storage required for version vectors scales with the number of venues in which a u-form is updated. Version vectors are a rather old technique; perhaps the original reference is [28] (though we believe it to have been independently invented many times). Version vectors are known to be a minimal representation for detecting violations of causal history [29]. An alternative mechanism for capturing causal history is the Hash History approach [30]. While it provides some interesting advantages over version vectors, it must be periodically pruned and can therefore lead to dangerous numbers of false conflicts in a large system.

If, upon replication of a u-form, we discover that one replica's version vector does not strictly dominate the other replica's, then we can conclude that concurrent modification has taken place. In some cases, the shepherd agents may be able to determine that two concurrent updates are not incompatible, so they are merged. Generally, however, concurrent updates result in a conflict. The conflicted u-form is annotated with pointers to the alternate versions.

Users may manually resolve conflicts, or applications may have specific automatic conflict resolution rules. Manual conflict resolution, while sometimes necessary, imposes a burden on the user, and automatic conflict resolution is only reliable in particularly well-understood application contexts (as also demonstrated by the designers of the Bayou system [4]). These problems grow worse as the system scales in size and as inter-dependencies between objects grow. This experience is consistent with the analytic results obtained by Gray in [3].

It is often better to avoid conflicts—in essence, to try to justify the optimism of our replication strategy through appropriate design. We will see u-form data structures that achieve this goal in section 4.

3.2 Shepherdable Indexes

Our repository infrastructure provides only very limited expressivity: given a UUID, one can update the associated u-form or subscribe to others' updates.

There is no low-level provision for value-based search, only UUID-based lookup. This narrow expressivity gives us a great deal of freedom to choose different underlying storage and replication models, and improvements in storage and replication speed and reliability have a beneficial effect on all applications that use the Shepherds system.

Instead of hard-wiring a few value-based searches into the infrastructure, we build index structures that effectively enable the implementation of value-based searches by composing several UUID-based searches. An index is a data structure that efficiently maps a key or range of keys to one or more values (typically UUIDs).[3] We represent indexes by organizing u-forms into tree-like structures, and annotating the u-forms in the tree with key information dictating which sub-tree is relevant to the query at hand. For example, a typical "index node u-form" may contain information that an index reader will interpret as "follow relation u_1 for names beginning *A–L*, follow relation u_2 for names beginning *M–Z*."

A hallmark benefit of this approach is that index nodes are replicated on demand by the shepherds, just like any other u-forms. This means that once a user has performed a particular search while online, this search can be repeated in the future when offline. An index with this property is called *shepherdable*. For a detailed account of shepherdable index structures see [31]. We have implemented and routinely use B-tree style structures [32] for handling one dimensional queries and R-tree style structures [33] for multi-dimensional and geo-temporal queries.

One important use of indexes in collaborative systems is the support of what we call *virtual relations*. Recall that a relation is formed when a u-form contains a UUID (or set of the UUIDs) as the value for one of its attributes. It is not always desirable (for all the usual reasons: access control, scalability, etc.) to directly encode relations in one of the u-forms implicated. Instead, we can construct an index whose keys and values are both UUIDs. Such an index is a mapping from UUIDs to UUIDs: thus, a virtual relation. Virtual relations are used extensively in the Universal Genetics Database [34], a research project that uses the Shepherds system to represent and share publicly available genetic databases, and enables researchers to annotate and reuse particular genes from these databases for special purpose projects, without writing to the centrally administered database. We will use this technique in section 4.3 to support standoff annotation and commentary.

3.3 Digital Signatures, Security, and Trust

We have seen that modifying the same u-form in many venues increases the likelihood of conflicts, and increases the amount of book-keeping information we must store in the form of longer version vectors. These facts act as forces that encourage us to keep the number of venues modifying a particular u-form small.

In practical systems, another consideration also plays a major role. In many cases, not every user has the authority to modify every u-form in the system. It is not necessary for a user to consistently use the same venue, but it is common

[3] In some sense a collection is a trivial sort of index with linear query performance.

that a given user will only use a small number of venues. Therefore, if the number of users permitted to modify a u-form is small, the number of venues in which it will be modified is also likely to be small.

To protect u-forms from being maliciously or accidentally modified, we employ digital signatures. A good comprehensive introduction to technologies such as digital signatures, cryptographic hashing, and encryption can be found in [35]. Signatures are attached to u-forms as normal attribute values, and public key credentials are published in the system as u-forms. This allows shepherd agents and applications to verify that u-forms are only updated by their proper owner.[4]

Fully- or partially-immutable u-forms can also be created by using cryptographic hashes of the u-form content as part of the UUID for the u-form. These are of somewhat less interest in highly dynamic collaborative systems, but are very useful for storing certain kinds of data.

4 Collaborative Structures

Our experience with the Visage project demonstrated to us the value of using u-forms as a shared collaborative space that operates by allowing a set of users to update a shared set of u-forms. We wish to design publicly available systems that do not presuppose investment in major datacenters, and that also support collaboration between users with poor or intermittent connectivity. These requirements guided us towards investigating optimistic replication and a peer-to-peer approach.

We have seen, though, three distinct pressures that require us to minimize the number of venues in which a given u-form is modified. One is the bookkeeping associated with version vectors and conflict detection; a second is the difficulty of lazily reconciling conflicts; and a third is the natural tendency to need to restrict write access to a subset of users. On the other hand, a legitimate worry is that restricting the original update-anywhere-anytime policy will impair collaboration.

In this section we examine techniques that help us to have our cake and eat it too. We describe a series of multi-u-form structures, along with conventions for using them, that let us perform many kinds of collaboration without needing to update the same u-form in many venues.

4.1 The Carrier Pigeon Protocol

One useful tool for collaboration is simple messaging: the ability for one user to send another user a message. Not only are the messages themselves handy

[4] There remain a variety of potential attacks that we will not discuss in this paper. One is a spoofing attack that we call a "land grab" in which a malicious user creates a new u-form with a pre-existing UUID. This results in two apparently valid but competing signed u-forms. Such an attack must resolved through human arbitration or high quality automated reasoning. Another class of attacks involves denial-of-service, and is best addressed in the replication layer.

for instant messenger or email style applications, but having such a technique available allows a distributed workflow: one user can request another user or automated agent to perform an action on his behalf.

As we have seen, our replication system does not presuppose that any two users are simultaneously connected. If each user is only intermittently connected, it may be that there is never transitive connectivity between them in the underlying network. However, our system does guarantee that replication eventually makes progress. Therefore the solution is to mediate messaging through u-form updates.

The most obvious way to do this is to have the two users who wish to communicate simply read and write to a shared u-form. This, however, leads to frequent conflicts.

Instead, we arrange matters so that each user has an outbox. User A writes to the A outbox, and listens to updates on the B outbox, while User B writes to the B outbox and listens to updates on the A outbox. We arrange the protocol as follows:

1. User A desires to send a message to user B. So User A writes an attribute into his outbox called `message_X`, where `X` is any locally unique token he chooses (a number is a reasonable choice). The content of the message is simply the value of the attribute `message_X`. User A also writes an attribute called `current_message_id` whose value is `X`.
2. Since User B is listening for updates on User A's outbox, User B will eventually see the message. She can then decide upon a response and write an attribute `response_X` into her own outbox, where `X` matches the message identifier token chosen by user A. The value of the attribute `response_X` is the response to User A's message.
3. Since User A is listening for updates on User B's outbox, he will eventually see the response. At that point, he can remove the `message_X` attribute from his outbox, keeping the outbox size reasonably small.
4. User B will eventually notice that User A has removed the `message_X` message. Consequently, User B can remove the `response_X` response from her outbox.

The important thing to notice about this technique is that it is guaranteed to eventually succeed as long as the underlying replication scheme—no matter what it is—can make progress. Moreover, no single u-form is modified by more than one user, so no conflicts occur, no security policies are violated, and no extra version vector housekeeping is incurred.

4.2 Collections and Recursive Collections

As we mentioned earlier, it is easy to construct a u-form that holds references to many other u-forms. We call such u-forms collections, and they are very useful for defining datasets, that is, sets of u-forms that a given user community is interested in.

Many of our visualization applications depend strongly on shared collections. For instance, a military commander may be examining a collection of his aircraft in a map visualization, while, simultaneously, a logistics officer is viewing the same set of aircraft in a chart showing fuel and ammunition supplies. It is very useful for these visualizations to be encoding the same collection u-form, so that if it is modified (say, to add or remove an aircraft), both visualizations are updated.

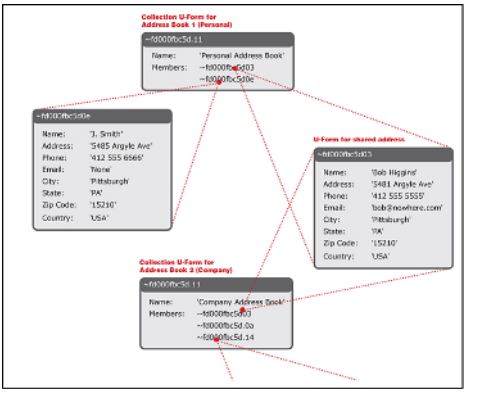


Fig. 3. Several collections

Collections, however, are very difficult to update consistently in a distributed fashion if many writers are involved. There is no unique "ground truth" method for producing a single collection by resolving several edits in different venues, especially if a globally consistent ordering is to be preserved. Moreover, the expense of version vectors is incurred and many users must have the authority to update the collection u-form.

One solution to the problem is to make one user the owner of the collection, and have any other user that wishes to modify the collection send a message (see section 4.1) to that user. This, in effect, converts our multi-master distributed system into a single-master client-server system *for this particular piece of data*. The nice thing about this approach is that any collection semantics can be supported, and we can assign ownership however we like for various pieces of data and datasets. The downside is that if the owner of the dataset is unreachable then no updates can be performed.[5]

Another solution is to factor a single collection u-form into a *recursive collection*. A recursive collection is a collection u-form whose member u-forms are themselves collections, together with some annotation to distinguish between the intent of adding a single collection or adding all the *members* of that collection. We can assign ownership of each "child" collection to a different user, and we can interpret the recursive collection as containing the union of the members of its children.

This works well provided ordering is not crucial to the application, and provided no single user needs to be able to fully delete an item from the recursive collection (since she cannot delete it from other people's "child" collections, only her own). Version vectors are kept small and digital signatures are easy to manage, because there is a one-to-one relationship between users and the collection u-forms they modify. In some situations it can be argued that managing the

[5] It should be noted, though, that the carrier-pigeon message approach will automatically queue messages until connectivity is restored, so updates are not lost, only deferred.

security policy, the conflict resolution, and the large version vector on a single collection provides superior performance to tracking all the child collections in a recursive collection. It must be noted, though, that safety demands that version vectors be kept forever, whereas only currently interested users need maintain child collections.[6]

Setting up the recursive collection can be handled using a single-writer-via-carrier-pigeon approach, since changes to the set of interested users (who also need to write) are probably comparatively infrequent.

It is likely that recursive collections can be generalized with additional annotations to support approximate or limited guarantees on the ordering of elements, but we have not yet experimented with this concept.

4.3 The Publication and Annotation Mechanism

Virtual relations, described in section 3.2, arm us with a powerful tool. A user can associate one u-form with another without requiring write access to either u-form. Instead, the user simply needs write access to a virtual relation index that supplies the mapping or, more commonly, someone to add the virtual relation on the user's behalf.

We have established a convention for using such virtual relations to perform collaborative enrichment of information. (The application of this mechanism to annotation of linguistic corpus data is described in [36].)

We define an *author* to be a user who creates u-forms and may be interested in annotating existing u-forms.

A *publisher* is some agent who vouches for the veracity of a piece of information. Each publisher also maintains a well-known virtual relation index. An author may be his own publisher, or may cooperate with a well-known publishing organization.

A *theme* is a u-form defining an area of interest for some community of users. These could be created in an ad-hoc fashion, or worked out in a standardization process. It depends on what the publishers and users find most convenient; the system doesn't care what mechanism is used to agree upon the meaning of themes—they are treated as unique labels.

An *annotation* is a u-form that comments upon another u-form with respect to some theme. (Because an annotation is itself a u-form, it might be the target of further annotations. The same goes for themes themselves.)

Each publisher maintains an index whose keys are UUIDs of u-forms that are being annotated (it is thus, by our earlier definition, an index of virtual relations). The values in the index are UUIDs of collections that store UUIDs of annotation u-forms, sorted by theme.

A user who wishes to publish an annotation on an existing u-form creates the annotation u-form, and uses a carrier-pigeon channel to ask the publisher to

[6] Technically, it is possible to prune version vectors if a safe distributed transaction can be performed over every venue in the system. Unfortunately, for internet-scale systems with intermittent connectivity such a transaction is essentially impossible.

relate the new annotation to the existing u-form through the publisher's index[7]. To maintain the integrity of the index, a publisher will typically copy the user's annotation to u-form that is signed by the publisher, and publish this version, rather than publish the user's original annotation (this avoids "bait and switch" abuses).

Similarly, if a user knows of a publisher's index, and a source u-form of interest, she can efficiently find all the themes and annotations published by that publisher for that source u-form by looking up the source u-form UUID in the publisher's index.

In practice, this technique supports rich stand-off annotation and commentary without requiring users to modify each others' u-forms. We will discuss an existing application of this mechanism in section 5. An advantage to using stand-off annotation over direct modification of data is that competing viewpoints and opinions can be captured and expressed. This is a much more natural mode of discourse in many respects than a tug-of-war over a single mutable information object.

Stand-off annotation is only a recent description of the much older scholarly practice of citation with commentary. This technique can be found in ancient Greek writings and in early scriptural scholarship. The tendencies apparent in such practices are the same as those we see today: the more definitive a text, the more people wish to comment upon it, and the less likely they are to be able to edit the definitive version with inline annotation or markup.

5 A Comprehensive Application: Collaborative, Distributed GIS

Our Geobrowser application [23] leverages all these techniques to support collaborative, distributed GIS. What do we mean by that? A Geobrowser user can browse and explore a large shared dataset of u-forms describing political and physical features of the Earth. (This dataset, called the Information Commons Gazetteer, is quite complex and is built from many freely available sources. Its structure is described in detail in [37].) The user can also create her own data and datasets and share them. She can also enrich existing datasets through the publication mechanism.

Each Geobrowser contains a database node that is part of the Shepherds system. Consequently, the Geobrowser application works well whether or not it is currently connected to the internet.

Let's walk through some of the techniques we have discussed and identify how they support specific Geobrowser features.

– U-forms are used to represent all of the data and much of the implementation of the Geobrowser application. It is thus "self-updating" since the underlying replication mechanism will update relevant u-forms transparently.

[7] The publisher, of course, is free to pursue any policy for deciding whether or not to publish an annotation. And, of course, a user may self-publish if no publisher is willing to work with him.

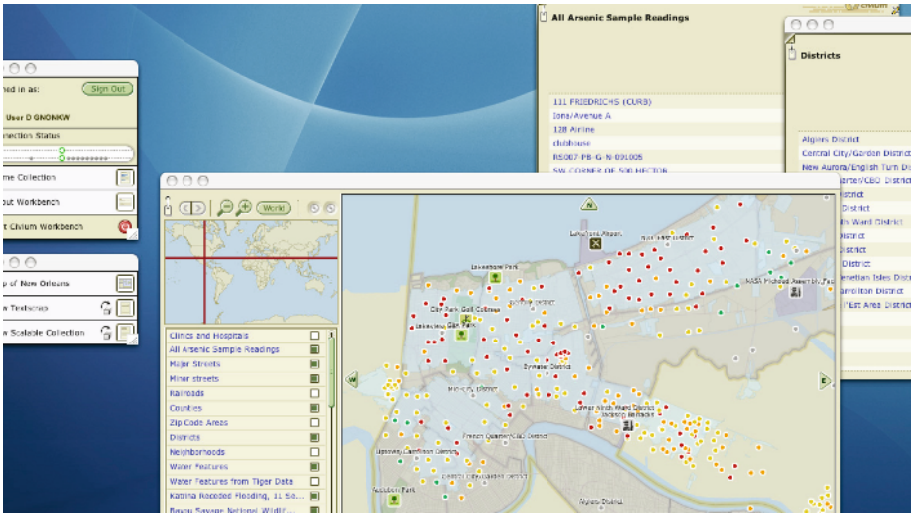


Fig. 4. The Geobrowser

- Digital signatures are created and checked automatically by the tools in the application using the active user's credentials.
- Carrier-pigeon protocols are used to support creating new Geobrowser users (a master index of users and public key credentials is maintained).
- Collections are used extensively to structure and organize data. Users can easily create and share new collections (as well as other sorts of u-forms).
- U-form based indexes are used to accelerate access to over 5 million physical and political features of the world. Both geo-temporal and string-search indexes are provided.
- The publication mechanism can be employed by users to provide free-text comments on any u-form in the system. The publication mechanism is also used to associate more structured data from multiple sources: for instance, data from the Wikipedia has been associated with the Information Commons Gazetteer through the publication mechanism.

The Geobrowser is in active development and is becoming a tool that reaches far beyond GIS. It is being used to as a content management system for the Community Directory websites [24,25], and to support research and collaboration in the biomedical domain [34].

6 Conclusion

U-forms as a shared collaborative space are a valuable and powerful tool. Optimistic peer-to-peer replication allows us to increase the reach of u-form based systems by improving scalability, and by allowing us to operate in disconnected or poorly connected environments.

This capability comes at a price, however. Managing conflict-detection bookkeeping in the form of version vectors can grow expensive over time as u-forms are updated in many venues. Lazy conflict resolution increases in difficulty as the number of conflicts grows and as the complexity of u-form structures grows. As the number of users grows, the desire for access control to protect data from accidental or malicious tampering also argues against overpermissive editing policies.

To continue to support rich collaboration despite these challenges, we have developed a variety of higher-level structures and protocols on top of our basic u-form storage and replication system. The high-level techniques demand only very weak guarantees from the underlying system, allowing us the freedom to exploit technological progress as it becomes available.

These high-level structures support simple user-to-user messaging, multi-writer datasets, and rich multi-user annotation and elaboration of data. We believe that other researchers interested in large-scale collaborative systems may also find these patterns efficient and effective.

Future work will continue to improve the efficiency and utility of these structures without compromising their ability to scale. Simultaneously, we are continuing to explore new replication and storage technologies for the underlying system. We are also developing end-user applications and tools for a variety of domains, including bio-informatics and GIS.

References

1. Lucas, P., Senn, J., Widdows, D.: Distributed knowledge representation using universal identity and replication. Technical Report MAYA-05007, MAYA Design (2005)
2. Saito, Y., Shapiro, M.: Optimistic replication. ACM Computing Surveys **37** (2005)
3. Gray, J., Helland, P., O'Neil, P., Shasha, D.: The dangers of replication and a solution. In: Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data. (1996) 173–182
4. Terry, D.B., Theimer, M.M., Petersen, K., Demers, A.J., Spreitzer, M.J., Hauser, C.H.: Managing update conflicts in Bayou, a weakly connected replicated storage system. In: Proceedings of the 15th ACM Symposium on Operating Systems Principles (SOSP-15), Copper Mountain Resort, Colorado. (1995)
5. Lucas, P.: Civium: A geographic information system for everyone, the Information Commons, and the Universal Database. In: Vision Plus 10, Lech/Arlberg, Austria (2003)
6. Stoica, I., Morris, R., Karger, D., Kaashoek, F., Balakrishnan, H.: Chord: A scalable Peer-To-Peer lookup service for internet applications. In: Proceedings of the 2001 ACM SIGCOMM Conference. (2001) 149–160
7. Rowstron, A., Druschel, P.: Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. Lecture Notes in Computer Science **2218** (2001) 329
8. Kubiatowicz, J., Bindel, D., Chen, Y., Eaton, P., Geels, D., Gummadi, R., Rhea, S., Weatherspoon, H., Weimer, W., Wells, C., Zhao, B.: Oceanstore: An architecture for global-scale persistent storage. In: Proceedings of ACM ASPLOS, ACM (2000)

9. Dabek, F., Kaashoek, M.F., Karger, D., Morris, R., Stoica, I.: Wide-area cooperative storage with CFS. In: Symposium on Operating Systems Principles. (2001) 202–215
10. Druschel, P., Rowstron, A.: PAST: A large-scale, persistent peer-to-peer storage utility. In: Proceedings of HOTOS (Hot Topics in Operating Systems). (2001) 75–80
11. Roth, S., Lucas, P., Senn, J., Gomberg, C., Burks, M., Stroffolino, P., Kolojejchick, J., Dunmire, C.: Visage: A user interface environment for exploring information. In: Proceedings of Information Visualization, San Francisco, IEEE (1996) 3–12
12. Lucas, P., Senn, J.: Toward the Universal Database: U-forms and the VIA Repository. Technical Report MTR02001, MAYA Design (2002)
13. Dertouzos, M.: What Will Be. Harper, San Francisco (1997)
14. Leach, P., Mealling, M., R.Salz: A UUID URN namespace. Technical report, The Internet Society (2004) Current draft, awaiting approval.
15. Manola, F., Miller, E.: RDF primer (2004)
16. Lucas, P., Widdows, D., Hughes, J., Lucas, W.: Roles in the universal database: Data and metadata in a distributed semantic network. Technical Report MAYA-05009, MAYA Design (2005)
17. van der Vlist, E.: XML Schema. O'Reilly (2002)
18. Higgins, M., Lucas, P., Senn, J.: VisageWeb: Visualizing WWW Data in Visage. In: Symposium on Information Visualization (Infovis), IEEE (1999) 100–107
19. Lucas, P.: Mobile devices and mobile data: Issues of identity and reference. Human Computer Interaction **16** (2001) 323–336
20. Bishop, D., Lucas, P.: Polymorphic collaboration: Beyond relaxed WYSIWIS in Visage-Link. Technical Report MTR-02007, MAYA Design (2002)
21. General Dynamics: Command post of the future (CPOF) (2005) http://www.darpa.mil/ato/programs/CPOF/DT.htm.
22. General Dynamics: GD Viz (2005) http://www.gdviz.com/.
23. MAYA Design, Inc.: Civium Workbench (2002) http://civium.maya.com/.
24. Allegheny County Department of Human Services: HumanServices.net (2006) http://www.humanservices.net/.
25. A-Plus Schools: Pittsburgh After School (2006) http://www.pghafterschool.com.
26. MAYA Design, Inc.: Buskarma (2002) http://www.buskarma.com/.
27. Higgins, M., Roth, S.: Shepherds and shepherd spaces. Technical Report MAYA-06009, MAYA Design, (prepared for DARPA) (2006)
28. Parker, D., Popek, G., Rudisin, G., Stoughton, A., Walker, B., Walton, E., Chow, J., Edwards, D., Kiser, S., Kline, C.: Detection of mutual inconsistency in distributed systems. IEEE Transactions on Software Engineering **SE-9** (1983) 240–247
29. Charron-Bost, B.: Concerning the size of logical clocks in distributed systems. Information Processing Letters **39** (1991) 11–16
30. Kang, B.B., Wilensky, R., Kubiatowicz, J.: Hash history approach for reconciling mutual inconsistency in optimistic replication. In: 23rd IEEE International Conference on Distributed Computing Systems (ICDCS'03). (2003)
31. Higgins, M., Widdows, D., Balasubramanya, M., Lucas, P., Holstius, D.: Shepherdable indexes and persistent search services for mobile users. In: 8th International Symposium on Distributed Objects and Applications (DOA 2006), Montpellier, France (2006)
32. Sedgewick, R.: Algorithms in C. Addison-Wesley (1990)
33. Guttman, A.: R-trees: a dynamic index structure for spatial searching. In: Proceedings of SIGMOD. (1984) 45–47

34. Widdows, D., Barmada, M.: The Universal Genetics Database: Information sharing in genetics and beyond. BioTech International **18** (2006) 11–13 (Byline article).
35. Schneier, B.: Applied Cryptography. 2nd edn. John Wiley and Sons (1996)
36. Balasubramanya, M., Higgins, M., Lucas, P., Senn, J., Widdows, D.: Collaborative annotation that lasts forever: Using peer-to-peer technology for disseminating corpora and language resources. In: Fifth International Conference on Language Resources and Evaluation (LREC 2006), Genoa, Italy (2006)
37. Lucas, P., Balasubramanya, M., Widdows, D., Higgins, M.: The Information Commons Gazetteer: A public resource of populated places and worldwide administrative divisions. In: Fifth International Conference on Language Resources and Evaluation (LREC 2006), Genoa, Italy (2006)

Developing Collaborative Applications Using Sliverware

Seth Holloway and Christine Julien

Mobile and Pervasive Computing Group
The Center for Excellence in Distributed Global Engironments
The University of Texas at Austin
{sethh, c.julien}@mail.utexas.edu
http://mpc.ece.utexas.edu

Abstract. Despite computers' widespread use for personal applications, very few programming frameworks exist for creating synchronous collaborative applications. Existing research in CSCW (computer supported cooperative work), specifically approaches that attempt to make current application implementations collaboration-aware, are difficult to implement for two reasons: the systems are focused too narrowly (e.g., on Internet-only applications), or the systems are simply too complicated to be adopted (e.g., they are hard to set up and adapt to concrete applications). Enabling real-time collaboration demands lightweight, modular middleware—*sliverware*—that enables the fine-grained interactions required by collaborative applications. In this paper, we introduce sliverware and give a specific example in the guise of a *distributed keyboard* that multiplexes input from several users into a single stream that each user receives just like input from a normal keyboard. The result is simple, real-time collaboration based on a shared, distributed view of data that enables rapid development of highly coupled coordinating applications.

1 Introduction

Even though computers worldwide are connected, the computer is still largely a personal experience; CSCW (computer supported cooperative work) research has focused on expanding the limitations of traditional computers to allow for more collaboration amongst users. However, CSCW has mainly provided *information exchange* rather than *information sharing* [6]. While both allow for collaboration, information sharing provides real-time interaction with a consistent global view of shared data. Despite the large volume of work that has been done in connecting people, simultaneous collaboration and information sharing are still relegated to face-to-face meetings and telephone or video conferences. Existing research into enabling collaboration is too narrowly focused to be usable by everyone; for example, available solutions may only work online or with a specific operating system, or they may rely on a particular programming framework. Additionally, the approaches often rely on heavyweight, unchangeable middleware systems which provide excessive functionality; the vast amount of functions cannot be easily understood and fully utilized yet they dominate the software size.

R. Meersman, Z. Tari et al. (Eds.): OTM 2006, LNCS 4275, pp. 587–604, 2006.

This paper introduces *sliverware*, which aims to simplify the development of collaborative software. A sliverware component (a *sliver*) is a thin slice of a modular middleware that provides a specific functionality appropriate to implementing collaborative software. Each piece of sliverware includes not only the programming abstractions a developer uses to create the particular interaction, but also the protocol implementations required to realize that interaction. By hiding the details of this implementation from the application programmer, sliverware allows developers of collaborative software to focus on *interactions* instead of low-level communication. An application requiring multiple modes of interaction pulls together exactly the required sliverware components in a modular fashion to create a tailored middleware layer that implements the interactions' underlying behavior. By providing full application stacks within one sliver, collaborative programs can be constructed out of sliverware with very little knowledge of distributed programming. Common components among slivers are automatically optimized at compile time to provide the smallest-possible implementation. Traditional middleware provides a horizontal abstraction, but sliverware introduces the notion of lightweight, cross-cutting, vertical abstractions.

This paper details the sliverware concept including an overview of the approach, details of each abstraction layer, and a description of how to program using sliverware. We then show an example to demonstrate sliverware's flexibility. This example, the distributed keyboard, logically multiplexes multiple users' inputs into one input stream. By replacing the normal keyboard listener with a distributed keyboard listener, an application developer can program real-time text-based collaboration. There are numerous applications in which the distributed keyboard can play a critical role, including interactive presentations—lectures in which the professor's material is broadcast to students instantly while students create annotations on the fly—and simultaneous text editors that allow multiple people to author a document simultaneously. These possibilities make the abstraction ideally suited for educational software and software development tools although many more potential applications exist. Results show that the approach is easy to implement and scalable.

In Section 2 we motivate this research and define the problem that sliverware solves; Section 3 details sliverware and its approach to program construction. Section 4 covers the implementation details for one example of sliverware—the distributed keyboard, and in Section 5 we discuss work related to sliverware and collaborative software. Finally, we conclude with Section 6.

2 Motivation and Problem Definition

Real-time collaboration is long overdue in software; for nearly 40 years people have only had access to single-user applications. With the advent of Internet-based communication, many people saw the power of multi-user applications. Today, almost everyone communicates via a chat program such as AOL Instant Messenger [3]. However, while everyone is chatting with peers, they are

working on single-user applications [13]. For increased collaboration, group members often meet face-to-face rather than using their computers directly. This is due largely to the fact that existing collaborative software provides only coarse-grained interactive capabilities. The types of tasks we commonly cooperate on demand varying levels of granularity and sharing of dynamic data in real time.

If such flexible interaction primitives are available, software could further enhance collaboration. For example, real-time collaboration in a shared word processor may improve productivity by allowing multiple authors to contribute simultaneously: everyone editing the document would see exactly the same up-to-date version. In a setting where collaboration is employed, contributors have a shared goal and work towards producing higher quality work in less time than is possible using only non-collaborative word processors. Another example is a collaborative software development environment that utilizes simultaneous text editing similar to the collaborative word processor. Agile development methods such as extreme programming (XP) [5] very nearly demand collaborative software environments. Rather than waiting for code to be checked into a version control system, programmers could view one another's progress in real time. The synchronous development environment may be an ideal collaboration target because the developer is also the end user, thus the end user has a solid understanding of the capabilities of software which avoids many of the traditional groupware pitfalls [8].

There is also great promise in enabling interactive lectures, particularly in an educational or industrial meeting. A growing number of lectures currently utilize computers through standard methods like slide shows and more diverse means such as electronic whiteboards [1]. Many of these technological improvements have removed several of the interactive qualities of traditional classroom environments. Enabling digital collaboration during lectures would foster an interactive environment where students personally augment the lecture with relevant annotations in real time. This method of note-taking gives students the opportunity to absorb the lecture and clarify notes without having to focus solely on copying the lecturer's words. The synchronicity also allows the audience to review the lecture and understand how different parts relate as opposed to seeing notes scattered in the margin. Real-time feedback from students enables lecturers to dynamically adapt the lecture's content and pace. These combined effects would allow for a truly interactive lecture with a room of active participants.

There are many CSCW ideas and applications related to this research, but none have received widespread usage due to several fundamental problems. First many collaborative approaches are complex and hard to use—the basic idea is incomprehensible or the development method is ill-defined. In addition, there is a general lack of programming tools tailored to enabling development of collaborative applications—the granularity of programming constructs is too low-level, focused on communication routines, and not tailored to the high-level interactive qualities of collaborative applications. The CSCW approaches are also too specific so they do not integrate with other systems easily. These combined challenges scare even seasoned programmers away.

In summary, existing programming constructs do not adequately address collaborative software developers' needs, including: expressive coordination and group membership primitives, an extensible and easy to use programming framework, and responsive communication-based interactions.

The work undertaken in this paper addresses the above challenges and presents the following specific essential characteristics:

- *Tailored collaborative abstractions*: we introduce abstractions that make collaborative programs easier to write and deploy quickly by encapsulating interactive capabilities within reusable programming constructs.
- *Extensible programming framework*: our approach allows the system to be modified by collaborative application developers at various levels of complexity.
- *Lightweight software footprint*: we provide a simple, efficient solution that dynamically minimizes the resources required by applications.

The next section introduces sliverware, which stands to overcome barriers to enabling collaboration in numerous ways. Sliverware supplies easy-to-understand abstractions tailored to collaboration functions so application developers can construct applications from existing components rather than having to program applications from the bottom up. This also frees the programmer from writing complicated, low-level communication methods. Also, the real-time interactions provided by sliverware help to solve the mismatch between face-to-face and mediated communication. Finally, sliverware can be extended and used in three distinct ways, providing tailorable abstractions for programmers of all skill levels.

3 Sliverware: A Constructive Programming Model

Sliverware is thin modular middleware that offers the functionality necessary for writing collaborative software. Application developers do not need to know vast amounts of distributed computing solutions before using sliverware; instead the programming model provides a hierarchical collection of components that each implement a specific functionality necessary to collaborative software. The sliverware approach abstracts all aspects of collaborative programming into three layers: collaborative services (the collaborative functionality and application programming interface, or API), group membership, and network communication. A specific sliver teams three components (one from each layer) to provide an abstraction that allows programmers to develop collaborative applications quickly and effectively. Novice programmers can use particular slivers through the slivers' high-level programming interfaces, while more advanced developers can combine components in a different manner to create new slivers.

Sliverware provides all the functionality application developers have come to expect from middleware while also allowing for execution on more resource-constrained devices. The latter benefit stems from the fact that sliverware's modularity enables an application developer to incorporate only the middleware

components required for the particular application instance. The framework implements many common collaboration tasks through the reuse of components. The model is flexible because it can provide many varying potential solutions from a vast library and ultimately trims the system at compile time. Sliverware is also specifically collaborative-oriented which allows it to tailor the programming interfaces and functionality underlying them to a specific set of tasks. This simplifies the complexity of the development task while still enabling a large number of similar applications as described in the previous section. Finally, sliverware is component-based which provides developers the ability to mix and match components at varying levels of abstraction.

Fig. 1 depicts the sliverware-program interaction model and shows that sliverware interacts with four layers in the application stack: the network, group, service, and API levels. Fig. 2, the sliverware component model, shows the essential components of a single sliver. High level sliverware functionality is decomposed into pieces for each of the three layers. An underlying network protocol is necessary to enable coordinating partners to exchange information; this is handled by the network communication layer. It plugs into the sliver via an *adapter* that provides the communication interface, described in Section 3.3. Collaboration also requires the coordinating partners to be organized into a group that contains all of the distributed application components that must be kept consistent. The group membership layer provides support for organizing and coordinating the group itself, and may provide specialized logic for distributing information to all group members. This group membership and the group interface for connecting group policies to collaborative services is described in Section 3.2. Finally, we think of the implementation and API layers as a single entity that encapsulates the collaborative functionality which is separate from the coordination or distribution method. This layer, described in Section 3.1, incorporates algorithms that provide collaborative support for the local application such as the methods to allow information sharing (e.g., creating periodic snapshots to build a consistent global view or processing snapshots from other group members). In the remainder of this section, we describe the responsibilities of each of these component layers in more detail.

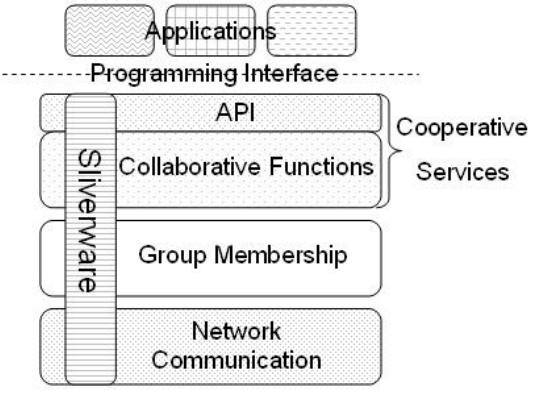


Fig. 1. Sliverware-program interaction model. Sliverware extends vertically through three layers: the network communication, group membership, and cooperative services layers. The sliverware hooks into the application via an API.

We then describe how developers program with slivers and how slivers can be combined automatically to optimize the resultant, tailored sliverware.

3.1 Collaborative Services

The uppermost layer in the sliverware model logically combines the collaborative functions and an API that presents the programming model to the developer to create a suite of *collaborative services*. This layer is founded on the premise that programming collaborative applications can be made easier if the programming primitives focus on functions integral to collaboration while hiding the necessary underlying group cooperation and network communication protocols. Implementations at this layer provide discrete fine-grained services that enable different pieces of functionality required for collaborative applications. For example, collaborative applications may exchange input information (e.g., keyboard or mouse events) occurring across a set of distributed devices, diffs of the same file open concurrently on multiple machines, simple idle/active status information of participants, or even screen shots for each connected user. For each such function, a collaborative service is defined that implements exactly the behavior dictated. A collaborative application can then be defined by integrating one or more such collaborative functions and providing application-specific behavior on top. This application-specific behavior is not part of the sliverware model, and includes determining how shared information is displayed and how often information should be exchanged. Some collaborative services may also automatically collect context information from the user or device generating data and attach it as part of the collaboration information.

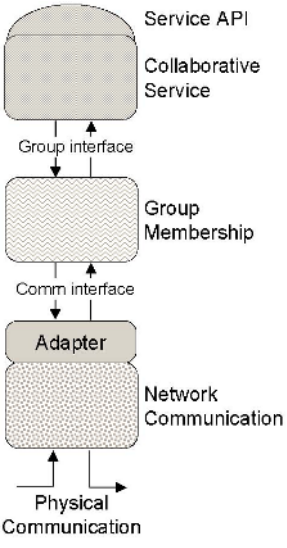


Fig. 2. Sliverware component model. An individual sliver combines one component from each of the three layers.

Section 4 gives an example of a specific collaborative function, the distributed keyboard. In this collaborative service, every participant in the collaboration group has attached a keyboard listener to a component in the application. When a keyboard event occurs in that component, the event is not only displayed appropriately on the user's local device (e.g., by displaying the letter typed), but the event is also automatically distributed to all other group participants. The distributed keyboard's API is defined to interact with the application in both directions (event generation and event reception). The information exchanged when interacting with different collaborative services likely differs greatly (e.g., interacting with a distributed keyboard listener is clearly quite different from interacting with a distributed mouse listener). In the sliverware model, we present this service interaction through a common `serviceListener`

that can be subclassed by a sliver component developer to be made to resemble local interfaces with which the developer is familiar. All context collection and event distribution is hidden from the application programmer and handled by the implementation of the collaborative service, and, ultimately, by the underlying communication and coordination protocols described next.

3.2 Group Membership

One of the most important aspects to supporting the collaborative services described above is the ability to support collaborative groups. Without defining the group that is collaborating (even if this group contains only two participants), then it is difficult to understand and specify the collaborative behavior (in the layer above) or to implement the collaborative behavior (which relies on the communication protocols described in the next section). For these reasons, group membership specification and implementation is the second, middle component, of any sliver in our model, and this component handles all steps necessary to store, collect, and learn group identities. The group membership policy that dictates how members are added to the group can present varying levels of complexity depending on how restrictive, consistent, and/or private the group should be. For example, the simplest group policy allows anyone who hears a communication to be part of that group. This policy offers no consistency guarantees however, because there is no information provided about which parties have received the message. More complicated group policies may require potential members to explicitly join, they may require members to periodically update their status to remain active, or they may even require members to know a shared password to participate.

Regardless of their semantics, groups are represented by well-defined data structures that contain the following components, if applicable:

$$group \triangleq (groupName,\ groupAddress,\ members)$$

This group data structure is maintained independently by each member; unless explicitly implemented by the group policy, no guarantee is provided that every group member has the same view of the current group. The *groupName* is a unique group identifier, *groupAddress* contains the group IP address for multicast purposes, and finally *members* is a list of all group participants. Any of these fields can be empty; though such null values make some sliver combinations impossible (for example, if the multicast address is not set, but the underlying communication is provided by a multicast protocol, there is no way to contact the other group members). These subtleties, their implications, and how our sliverware model and framework handles them are discussed in the next section. The *members* portion of the *group* data structure is a list of individual members each presented in the form:

$$members \triangleq (name,\ address,\ attributes)$$

where *name* is the group member's identifier, *address* is the group member's IP address, and *attributes* contain any other information that the program may

track. As with the *group* data structure, none of these aspects are required; if the underlying communication protocol is using a multicast address, it is possible that an IP address for each individual member is superfluous. The *attributes* are largely open-ended, and different group policies may use them for different purposes. As one example, if the group policy requires periodic *heartbeats* from each group member, the time to the next expected heartbeat would be included as an attribute for each member. Context information about the node, such as physical location, can also be included in the attributes.

The actual implementation of a group membership component depends on the particular policy employed by the group membership protocol. In our prototype, we present two widely applicable group membership policies: `announce` and `n-heartbeat`. The first of these, `announce`, provides open access to the group that allows anyone to join by simply announcing their intent. The policy does not attempt to prune unproductive members and instead relies on explicit departures from the group. To join the group, an application component's group membership component broadcasts a `join` request for the group. This is received by every node in the network, and, if the node supports an application participating in the specified group, the associated group data structure is updated with information about the new member. At this point, when any member of the group sends collaborative information, it is received by all other members specified in the group data structure *if they are still connected*. When an application component wishes to depart the group, it sends a `depart` message to all nodes it knows to be part of the group.

While the `announce` policy is useful for supporting many applications that are relatively static, more complex policies are necessary to support more sophisticated applications. For example, more stringent joining processes or more rigorous exit criteria may be programmed to suit the application. This brings us to our second prototype group membership policy, `n-heartbeat`, which requires that all participants periodically broadcast heartbeat messages; if a member does not communicate within `n` heartbeat periods, the member is assumed to have disappeared and is removed from each member's group data structure. Each member performs this removal independently; there is no guarantee that all members will have the exact same lists at all times, but if a node is no longer sending heartbeat messages, every active group member will eventually remove that node from the group data structure. To join a group, a new member sends a `join` request as above, but in addition, the node must periodically send a short message indicating it is still present and active. Still more sophisticated group membership policies are possible, for example, based on relative physical location and its implied notion of future connectivity [9].

Because of its modular design, the sliverware framework allows for more group membership policies to be integrated quickly and easily. A group membership component developer simply needs to provide functionality for joining the group and any constraints on the group interactions; this allows support for managing user group life cycles. The interface the group membership layer presents to the collaborative service layer contains a `send(Message m)` method (for calls coming

from a collaborative service above) and a `GroupListener` for delivering group events from other group members to the collaborative service. Before invoking the collaborative service's listener, the group membership implementation is responsible for ensuring that the message received is in fact destined for this group and that the sender is a part of the group. The interface the group membership protocols use to interact with the lower, network communication layer contains some additional methods and is discussed in more detail next. In summary, to add a new group policy to the framework, the policy developer is responsible for understanding how to use the underlying communication interface, for providing an implementation of the `send` method called by the collaborative service, and for invoking any registered group listener to deliver messages to a registered collaborative service.

3.3 Network Communication for Collaboration

Communication is a necessary part of collaboration—without proper communication there is no collaboration. With real-time collaboration in mind, communication should generally include every group member, be immediate, and ensure that every contribution that is made is registered. However, many factors influence the quality and cost of communication, including the degree of synchronicity, priority requirements, timeliness, etc. Combinations of these factors lead to numerous ways to provide communication for collaborative applications. Complicating this is the fact that different applications have different performance requirements and communication approaches perform differently when faced with different operating constraints. Given these discrepancies, it becomes apparent that there is no common communication approach that is the best choice for all applications or even for a particular application in every scenario. For this reason, the sliverware programming model simply defines a common interface, or *adapter* [7], that communication approaches must adhere to in order to be usable by the other, higher level, sliverware components. This is similar to the requirement above that new group policies adhere to the specified interface, but in this case we make the particular interface an explicit individual component because we do not modify the communication protocols themselves. Because a standard interface is used, an application developer can swap in different communication implementations without altering any of the higher level implementation components.

The basic interface for collaborative communication in our sliverware model allows both communication with a single member of the collaborative group and simultaneous communication with the entire group. While reliable multicast [12] appears at first glance to be an obvious widely applicable solution, it is difficult to implement in the dynamic, lightweight, wireless systems we are targeting [11]. A reliable multicast implementation may be the correct choice in some cases, but, depending on the situation, a simpler flooding-based broadcast may be sufficient. In other cases, more tailored protocols may be appropriate. For example a cooperative text editor that will only involve a select group of individuals in the same boardroom (such as in a meeting) could utilize simple flooding, while the

same application connecting a small group within in a large audience (such as a conference) may use ad-hoc on-demand distance vector routing (AODV) [15] or multicast ad-hoc on-demand distance vector routing (MAODV) [16]. In either case, the application developer needs only to select the proper network communication component for the sliver and the common send interface (defined by the adapter) will distribute the necessary data to the group.

Method	Description
`join(String name)`	called by the group membership layer on the network communication layer to enter the callee into the group with the specified name
`depart(String name)`	called by the group membership layer on the network communication layer to remove the callee from the specified group
`send(Address a, Message m)`	called by the group membership layer on the network communication layer to send the specified message to a single recipient: the node indicated by the specified address
`sendAll(Group g, Message m)`	called by the group membership layer on the network communication layer to send the specified message to all members of the specified group
`receive(Message m)`	called by the network communication layer on the group membership layer to deliver the specified message; implemented within the `CommunicationListener`

Fig. 3. The interface between group membership and network communication

The methods comprising the interface between group membership and network communication are shown in Fig. 3. The `join` and `depart` methods were discussed previously; they are used by some group membership protocols that need to explicitly join and leave groups. At first glance, one might think these methods ought to be implemented within the group membership layer itself and not involve the network communication layer, but some join activities require participation of the communication protocol (e.g., joining a multicast group) while others do not (e.g., communication that relies on broadcast for every message). Because our foremost goal is pushing all knowledge of communication to the network layer, we require the group membership layer to know only the name of a group in which it participates, and the protocol delegates knowledge of how the communication protocol operates to whatever implementation resides in the network communication layer of a particular sliver. In this manner, the group membership protocol performs exactly the same behavior regardless of the nature of the underlying communication protocol.

The remaining three methods allow data to be exchanged among group members. The common element in these three methods, `Message`, includes the *group name*, the *source*, and the *data* to be transmitted. The functionality required

to send a message to everyone in the group changes with the networking protocol, but the interface distances the application developer from low level details. Overall multicast functionality is provided by the `sendAll(group, message)` method; here, *group* refers to the data structure presented earlier that contains the *group name*, *group address*, and *members* list. Recall, however, that the *group* data structure does not have to contain each of these elements for the communication protocol to be able to function correctly. For example, AODV does not support multicast, so the *group address* provides no useful information. However, if the elements of the *members* list are omitted, AODV cannot reach the other group members since it relies on unicasting the message to each of them. In this case, the method triggers a `NoMulticast` exception. On the other hand, MAODV is a multicast protocol that automatically provides group-wide communication. In this case we expect the *group* element to contain a *group address*, and we trigger an exception if the element is undefined. It is important to note that we do not alter the communication protocol implementations themselves and instead implement these behaviors in the adapter interface. As an example, the code implementing `sendAll` for an AODV adapter looks like the following:

```
void sendAll(Group g, Message m) throws NoMulticast, UnreachableHost {
   if(g.members == null){
      throw new NoMulticast();
      return;
   }
   for(int index = 0, index < g.members.size(); index++){
      if(g.members[index].address != null){
         AODVSend(g.members[index].address, m);
      }
      else{
         throw new UnreachableHost(g.members[index]);
      }
   }
}
```

The above is an example of the simple interfacing code that a sliver developer must write to be able to include a new communication protocol as a component in the network communication layer.

If the network communication is implemented by a brute-force broadcast protocol, the `sendAll` method will deliver the message to every connected node, even those that do not support members of this particular group. As the group membership layer is communication agnostic, so the network communication layer is group agnostic. When the network communication layer receives a message, it delivers it to the group membership layer. It is then the group membership protocol's responsibility to filter these messages and ensure only proper messages are delivered to the collaborative service and ultimately the application.

There may also be a need for communication between single members of the group; this may be used to implement `join` and `depart`, but it may also be useful for sidebar coordination within the group. This unicast style of behavior is achieved through the `send` method in Fig. 3. In AODV this method is the

base functionality, so little extra work is necessary. In the case of MAODV and flooding we rely on the group membership layer at the receiving end to filter all messages where the target address does not match the host address.

All incoming messages trigger the `receive` method in the `CommunicationListener` interface. This method's implementation is likely quite simple in all cases; it is a basic pass-through of information from the network to the group membership layer. If the message received was not sent by a member of the group, or if the group identifier in the message does not match a group id of the receiver, the group membership layer filters the information rather than passing it to the application layer.

Our prototype contains an implementation for flooding, unicast, and multicast based protocols; a sliver component programmer can easily insert another protocol by creating an adapter for that protocol. In no cases is it necessary for the programmer to modify the existing communication protocol implementation; the adapter simply changes the interface to conform to our framework. Therefore, this system allows very low-impact changes to the network layer while providing extreme flexibility and extensibility.

3.4 Programming with Sliverware

Sliverware is an extensible, modular model for enabling collaboration in applications. The abstractions provide collaborative services that are easy to understand and use. Referring back to Fig. 2, each sliver provides a discrete collaborative function and comprises the local implementation of that function, the implementation of a group membership policy that defines who participates in that collaboration, and a network communication component that facilitates physical message exchange. Programming collaborative applications with the sliverware model is straightforward and simply requires a developer to select, use, and combine slivers that provide the high-level collaborative functions the application demands. This programming may result in a sophisticated application that encompasses several collaborative functions and may even combine views of information from multiple different groups. For example, in a classroom support application, a group may be defined for the entire class so the teacher can share materials with every student, and there may be additional groups defined for a team project. A single student's display may simultaneously show information from both of these collaborations side by side.

From the programmer's perspective, each of these behaviors is programmed independently as a single sliver. When the application is compiled, the sliverware framework optimizes the resulting implementation by combining functionality that is duplicated across slivers. Fig. 4 shows this process in a bit of detail. The left of the figure pictorially represents an application that combines three separate slivers: a distributed keyboard shared with all of the members of the class, a distributed keyboard shared with the team members, and a distributed mouse shared with the team members. The classroom application may have a dedicated window for keyboard events shared with the class and a separate window for keyboard and mouse events shared with the team. When a keyboard event occurs,

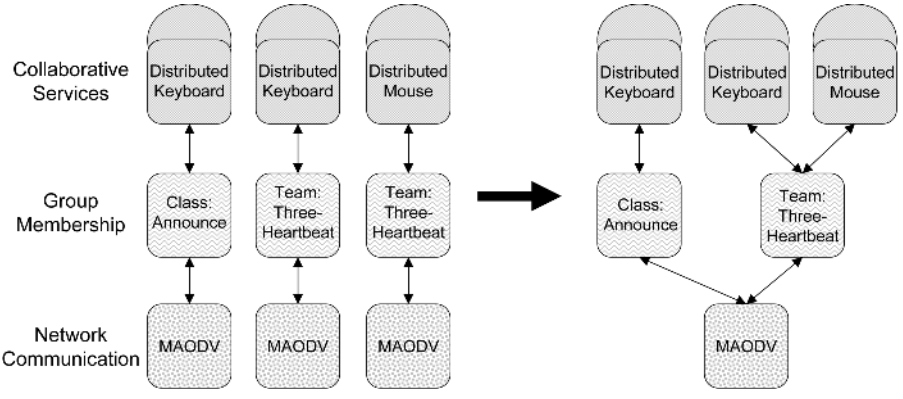


Fig. 4. A visualization of the sliverware simplification; components that are reused are combined to create a minimal set of functionality that is lightweight yet functional

the collaborative application implementation (not shown) first determines which window had the focus and then triggers the appropriate collaborative service.

When compiled, the actual middleware deployed in support of the application is shown on the right of Fig. 4. The duplicate components (i.e., the MAODV implementation used and the group membership policy implementation for the team) have been combined for efficiency. When a message is received at a node, the MAODV implementation passes it to both group implementations (because it doesn't know how to read or process group information). Each group processes only the messages destined for its members (as indicated by the *groupName* carried in the message). The team group implementation passes all group messages to both collaborative services (i.e., both the distributed keyboard and the distributed mouse), but each of these services only passes along events whose data portions match the type for that service. In general, duplicated network communication protocols are always combined. Ultimately, the compiler optimizes the set of slivers to create a tailored lightweight middleware. This is in contrast to traditional approaches where an entire middleware system must be deployed and invariably contains functionality that will not be employed by *every* application. Sliverware's approach leads to a much leaner execution environment which can translate into increased performance and greater support for heterogeneity.

The flexibility described above illustrates an important property of the sliverware programming model: the ability to create new slivers by adding and exchanging components. Overall, sliverware can be programmed by three distinct classes of developers with increasing levels of programming expertise. First are the collaborative application developers who access suites of available preconfigured slivers and combine them into applications as described above. Second are the sliver developers who combine sliverware components from each of the three layers to create new combinations presented as slivers. Finally, the most experienced class of programmers, sliverware component programmers can develop new sliverware components that can be used in sliver combinations. For each

component level, we discussed previously the steps this developer must take to create a new implementation that adheres to the well-defined sliverware model.

Sliverware allows users of all skill levels to use and extend the sliverware system by the unique hierarchical structure of the sliverware approach. Multiple layers of abstraction lead from the low-level implementation to useful high-level functionality. By providing a complete framework based on multiple layers of abstraction, sliverware is a flexible, extensible, and reusable model that provides collaborative services.

4 An Example Sliver: The Distributed Keyboard

Sliverware enables collaborative applications to be developed quickly and removes repetitive, burdensome communication and coordination methods. To aid understanding of sliverware we present an example application, a chat program in which a group of users shares keyboard inputs that occur within the application. The collaboration is achieved using a distributed input device, the distributed keyboard. The distributed keyboard listener replaces the traditional keyboard listener and multiplexes all the users' key-presses from within the application into a stream similar to the input from a standard keyboard.

Fig. 5(a) demonstrates how the traditional keyboard listener connects individual users to their machines. Fig. 5(b) shows the simple approach taken by the distributed keyboard listener; the individual inputs are joined into one logical distributed input device which is fed to each machine. This approach allows individuals to remain at their own computer while maintaining a consistent global

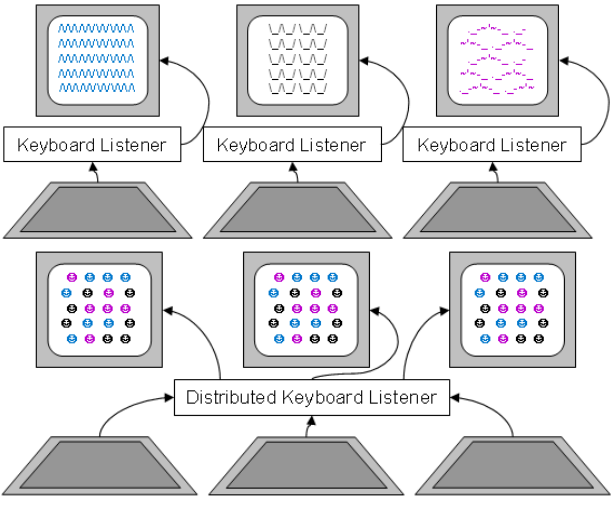


Fig. 5. The traditional keyboard listener connects a single user to a computer (top). The distributed keyboard listener connects multiple users while still allowing them to use their local machine (bottom).

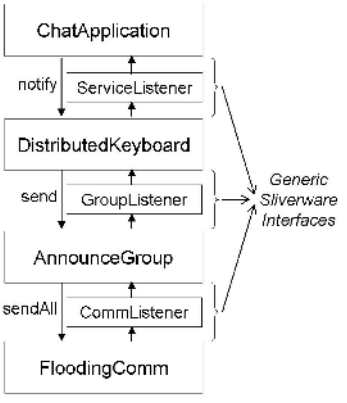


Fig. 6. The application stack for our Chat Application

view. The distributed keyboard appears to function as a standard client-server implementation, however, instead of routing all packets through a server, the sliver automatically multicasts users' input to the group. Bypassing a central server speeds the implementation, allowing for more immediate interactions. The result is a chat program that immediately broadcasts each group member's key-preses and displays the input in a text box which is consistent across the group.

Fig. 6 shows the application stack underlying the Chat Application implementation. The collaborative application programmer contributes only the uppermost block of this stack. The programmer uses the `ServiceListener` interface and the `notify` method to plug into the sliver used (in this case, a distributed keyboard implementation). The distributed keyboard demultiplexes events from other group members and delivers them to the application. It also receives the local user's events, packages them, and sends them to the group implementation (in this case, an instance of the `announce` group). Through the `GroupListener` interface, the `announce` group implementation distributes incoming events to the registered collaborative service (i.e., the distributed keyboard). The `announce` group implementation also passes the message from the local collaborative service on to the communication implementation through the `sendAll` method which delivers the key event to all other group members. The interfaces between the layers in Fig. 6 are generic sliverware interfaces included in the sliverware framework. Regardless of the type of component employed at each layer, it is a given that the same interfaces are used. These generic interfaces enable sliverware components to be exchanged for each other without impacting other components in the sliver.

Fig. 7 shows a screen shot the sliverware chat application; the implementation can be found on our group webpage at `http://mpc.ece.utexas.edu/research/sliverware.html`. Each group member appears in the chat window; as the user types, their key-presses are displayed in real-time on all connected users' screens. This particular application chooses to display distributed key events

from different users based on the events' relative times of arrival. Other application uses of the distributed keyboard may also use cursor position information to allow users to edit a single, shared document.

Fig. 7. The distributed keyboard listener in action

5 Related Work

Since the rise of CSCW, there has been a great deal of research devoted to enabling collaboration; however, current collaborative approaches are largely based on asynchronous communication such as that provided by the Microsoft Suite's Track Changes function [4]. To collaborate on a document, a group member can turn on Track Changes, then edit the document, save it, and distribute it to the rest of the group. This edit, save, send cycle leads to an asynchronous communication that effectively locks the document while someone else edits. Another common approach, simply emailing an individually edited document to another team member, also provides only asynchronous collaboration based on information exchange. Sliverware provides more synchronous communication that allows all connected group members to make and share changes concurrently.

Similar to sliverware, many new products offer synchronous editing. For example, Writely [18] provides an online collaborative word processor with the use of AJAX. There are a raft of similar solutions as part of the WebOS revolution [2] including SynchroEdit [17] and JotSpot Live [10]. While these systems are a great step forward in real-time cooperation, they have a reliance on an external server which may not always be accessible or secure. Also, software functionality can only be provided by the product team; users must trust the content provider to protect the data and add necessary features in a timely manner. Sliverware can be used to provide similar functionality for applications on the Internet or the desktop with greater control of the application which allows for greater security and extensibility.

The aforementioned applications are written with a specific collaborative goal in mind, so while they may be useful, they are only useful for the tasks they can already perform. Sliverware is an extensible framework and has broad goals for enabling collaboration through diverse means. Sliverware shares goals with existing CSCW frameworks such as DISCIPLE and BSCW which allow users to

simultaneously change documents. DISCIPLE [14] is a framework for sharing JavaBeans applications in real-time through the use of CORBA to replicate objects. BSCW provides information sharing across the world wide web with a web server that is extended using CGI scripts. DISCIPLE and BSCW provide limited usability due to the reliance on web programs; while the ideas may be extended to existing applications, the frameworks themselves cannot. Sliverware's collaborative abstractions work in desktop publishing applications as well as web applications. Sliverware focuses on a broader abstraction to enable collaboration on a larger scale.

While all the related work in collaboration is useful, the offerings do not explicitly define a simple framework for adding synchronous collaboration to existing systems across domains. Sliverware enables collaboration in a manner that is comprehensible and easily extended. Collaborative application developers have complete control of their product without spending valuable time on low level programming.

6 Conclusions

Sliverware is designed to enable collaboration in applications through an extensible, lightweight framework that is easy to understand and easy to use. Sliverware provides the same benefits as traditional middleware, but unlike traditional middleware which provides a horizontal layer that pre-invents the wheel, sliverware focuses on thin vertical pieces of the complete application stack—network communication, group membership, and a collaborative service. Additionally, while traditional middleware provides a great deal of functionality regardless of the applications' needs, sliverware provides more focused pieces of functionality that can be optimized to ensure that the system remains as lightweight as possible.

Sliverware's extensible framework enables programmers at all levels to contribute to and use the system. Sliverware provides a framework to quickly enable collaboration in programs by furnishing a set of lightweight middleware modules; application-developers can build a system by assembling sliverware and writing minimal amounts of code to utilize the collaborative service through the API—the developer is not bogged down with low-level details and can instead focus on high-level programming.

Acknowledgments

The authors would like to thank the Center for Excellence in Distributed Global Environments for providing research facilities and the collaborative environment. This research was funded, in part, by the NSF, Grant # CNS-0620245. The views and conclusions herein are those of the authors and do not necessarily reflect the views of the sponsoring agencies.

References

1. G. Abowd, C. Atkeson, A. Feinstein, C. Hmelo, R. Kooper, S. Long, N. Sawhney, and M. Tani. Teaching and learning as multimedia authoring: The classroom 2000 project. In *ACM Multimedia*, pages 187–198, 1996.
2. S. Adler. WebOS: Say goodbye to desktop applications. *netWorker*, 9(4):18–26, 2005.
3. Aim. http://www.aim.com/, 2006.
4. B. Barrios. Tutorial microsoft office word 2003: Collaboration. http://getit.rutgers.edu/tutorials/word_collaboration/media/collaborative.pdf, 2002.
5. K. Beck and C. Andres. *Extreme Programming Explained : Embrace Change (2nd Edition)*. Addison-Wesley Professional, 2004.
6. R. Bentley, T. Horstmann, J. Trevor, and K. Sikkel. Supporting collaborative information sharing with the world wide web: The BSCW shared workspace system. *4th International World Wide Web Conference*, pages 63–74, 1995.
7. E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software (Addison-Wesley Professional Computing Series)*. Addison-Wesley Professional, 1995.
8. J. Grudin. Groupware and social dynamics: Eight challenges for developers. *Communications of the ACM*, 37(1):92–105, 1994.
9. Q. Huang, C. Julien, and G.-C. Roman. Relying on safe distance to achieve strong partitionable group membership in ad hoc networks. *IEEE Transactions on Mobile Compututing*, 3(2):192–205, 2004.
10. Jotspot live. http://www.jotlive.com/, 2006.
11. J. Kuri and S. K. Kasera. Reliable multicast in multi-access wireless LANs. *Wireless Networks*, 7(4):359–369, July 2001.
12. J. Lin and S. Paul. RMTP: A reliable multicast transport protocol. In *INFOCOM*, pages 1414–1424, 1996.
13. T. W. Malone and K. Crowston. The interdisciplinary study of coordination. *ACM Compututing Survey*, 26(1):87–119, 1994.
14. I. Marsic. DISCIPLE: A framework for multimodal collaboration in heterogeneous environments. *ACM Computing Survey*, 31(2es):4, 1999.
15. C. E. Perkins and E. M. Royer. Ad-hoc on-demand distance vector routing. In *WMCSA '99: Proceedings of the Second IEEE Workshop on Mobile Computer Systems and Applications*, page 90, Washington, DC, USA, 1999. IEEE Computer Society.
16. E. M. Royer and C. E. Perkins. Multicast operation of the ad-hoc on-demand distance vector routing protocol. In *MobiCom '99: Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, pages 207–218, New York, NY, USA, 1999. ACM Press.
17. Synchroedit. http://www.synchroedit.com/, 2006.
18. Writely. http://www.writely.com/, 2006.

A Framework for Building Collaboration Tools by Leveraging Industrial Components

Du Li, Yi Yang, James Creel, and Blake Dworaczyk

Center for the Study of Digital Libraries (CSDL)
Texas A&M University
lidu@csdl.tamu.edu

Abstract. Groupware applications allow a distributed group of human users to work apart together over a computer network. They are difficult to develop due to the needs to suit a range of collaboration tasks that are often with diverse and evolutionary requirements. To address this problem, we propose a new framework in which shared data components conforming to a well-defined interface can be dynamically plugged in for flexible sharing, and a simple transformation tool is provided such that the myriad of industrial collaboration-transparent components can be transformed into shared components. The validity of our framework is evaluated by building a suite of typical collaboration tools such as group editors. Under our framework, most components in the Java Development Kit (JDK) can be transformed automatically for prototyping collaboration tools. With minimal manual work, those tools can be adapted to achieve advanced flexibility, e.g., data and control components can be bound dynamically to switch control protocols.

1 Introduction

The support of distributed collaboration has been penetrating modern computer systems in the past decade. This is evidenced by the increasing popularity of standalone groupware applications, such as instant messengers, multiplayer games, and electronic meeting systems, and groupware features integrated in group productivity tools, such as many recent programming environments and office products. In general, groupware needs to win a critical mass of users to be a success [5]. This often requires that they be sensitive to the needs of a variety of user groups and collaboration tasks. However, the needs of users and tasks often differ and evolve over time [18].

One way to address this challenge is to provide customizable groupware tools which allow users to set parameters that match their particular preferences. However, this approach often results in bloated systems that are difficult to maintain and evolve, while the level of flexibility is limited to what are foreseen and parameterized at design time. If groupware tools are developed *ad hoc* as separated applications, reusability and extensibility will be limited.

An alternative approach is to provide a reusable groupware infrastructure for users to easily prototype desired groupware tools as the needs emerge. Conceptually, this would make the practices of groupware development more systematic,

R. Meersman, Z. Tari et al. (Eds.): OTM 2006, LNCS 4275, pp. 605–624, 2006.

resulting in lower engineering and re-engineering costs, well-architected systems, and a consistent look and feel across different tools.

Recent works such as [7,17,22] represent a trend to integrate the practices of groupware development and the practices of component-based development (CBD) in software engineering [20]. While it is well understood that groupware infrastructures must provide suitable programming abstractions [16], open issues include how to model components in groupware and how to fill the gap between groupware components and current industrial components that do not observe custom groupware component models.

We propose a novel framework called EXEC (an Evolvable and eXtensible Environment for Collaboration) that addresses these two issues. First, it includes a groupware component model, which decomposes a range of groupware applications into shared data components and control components. Based on this model, data and control components can be developed as independent libraries, and the infrastructure provides services to dynamically compose them at run time. Second, it includes a transformation tool which is able to transform existing collaboration-transparent components (e.g., JavaBeans) such that they conform to our custom component model and can be plugged into our platform for sharing. This way our framework can leverage the myriad of industrial components for fast-prototyping flexible groupware applications.

In the next section, we review related works to motivate the proposed framework. Then we elaborate the proposed framework in Sections 3 and 4. Next, Section 5 evaluates the framework by concrete design examples. Section 6 summarizes contributions and future directions.

2 Related Work

There are generally two approaches to developing groupware: The first is *collaboration awareness*, in which the system is designed to support multiuser interaction, possibly aided by an infrastructure (toolkit) with reusable services [10]. The second approach is *collaboration transparency*, in which a (reusable) collaboration-aware infrastructure is provided such that an existing single-user application is shared for multiuser interaction without modifying its source code [12]. Recent *component-based approaches* combine the advantages of both [17].

Collaboration Awareness. There are a plethora of specialized groupware applications, e.g., [19]. Over the years, many of the collaboration services such as those for group communication, consistency control and session management are identified as common to a variety of groupware applications. Reusable services are provided in groupware toolkits such as DistEdit [10], Suite [3], and GroupKit [16] such that new groupware applications can be built more easily.

Toolkits generally provide custom programming abstractions with embedded consistency control protocols, and assume that groupware applications are developed based on these abstractions. For example, when used for building a group editor, Suite and GroupKit both require that the shared data be manually constructed in their custom data structures (e.g., sequences [3] and dictionaries

[16]). In general, they do not address how to reuse existing single-user programs. DistEdit [11] explores how to manually convert existing single-user applications (editors) into collaboration tools. However, it assumes that the program is well-structured so that it is easy to modify the source code to call the toolkit APIs. For programs that are not well-structured, the work is admittedly tedious.

Collaboration Transparency. Any single-user application can be turned into a multiuser application automatically by generic application sharing infrastructures such as Microsoft NetMeeting. Due to the lack of application semantics in the infrastructure, however, all collaborators must see exactly the same view. The infrastructure only allows one user (floor holder) to modify the content at a time by intercepting the window events. Flexibility is generally traded off for reuse of single-user applications in entirety.

Flexible JAMM [1] can dynamically replace components in (Java Swing based) single-user applications with custom multiuser versions. It is a major step towards achieving more flexible sharing in collaboration transparency systems. However, while it allows for flexible concurrent work on those custom-built multiuser components, the whole application is still subject to floor control. Moreover, Flexible JAMM does not address how to build the custom multiuser components that allows for flexible sharing. Several recent systems achieve flexible sharing at the application level. ICT [13,14] and CoWord [21] explored how to convert familiar single-user editors into group editors without modifying source code. However, they both require considerable manual implementation of adaptation code for detecting state changes and implementing synchronization. In those approaches, flexibility is achieved at the loss of automation.

Component-Based Frameworks. The increasing acceptance of component-based development provides new opportunities. Component-based architectures such as JavaBeans and .Net define standard ways for introspecting the internals of components and composing components to construct new applications. Accordingly, a number of component-based groupware frameworks have been proposed recently, such as [7,17,22]. The main differences lie in how groupware components are modeled and how existing components are leveraged.

The JView architecture [7] supports the dynamic composition of groupware components that implement a well-defined interface. It does not automatically leverage or transform components not based on the custom interface. In addition, their groupware components as a unit of composition do not distinguish between data and collaboration control services. For example, in its group editing component, the locking protocol is hardwired. Consequently, it is not possible to dynamically apply alternative consistency protocols (e.g., operational transformation [19]) on the shared document without a redesign of components.

CAISE [2] is a recent extensible platform aiming to support the entire collaborative software engineering process. However, from the perspective of groupware frameworks, it is essentially not different from early groupware toolkits in the sense that it requires that new collaboration tools be coded to call APIs such that a number of common collaboration services can be accessed with reduced

costs. In the APIs, typically functions such as consistency protocols are hard-wired and the resulting collaboration tools suffer from rigid control. It does not automatically transform existing single-user programs as does this work.

The work of [17] is a recent component-based framework that supports flexible composition of both application and infrastructure components. It extends the naming conventions in JavaBeans such that component properties can be defined in a more general manner. Their work does not address automatic program transformation. The essential difference is how state change events are intercepted before they take effect on shared data. They either use user- or timer-triggered diffing, which is admittedly not efficient, or require the developer to manually add notification code to the original component programs. By comparison, shared components in our framework proactively notify the infrastructure of state changes, which is more efficient. The notification code is automatically injected into the subclasses automatically derived from bytecode of the original components, which means that we do not need to modify source code of components that are to be shared. At current stage, often a few lines of code is required to implement accumulative (or indexed) properties in some components for achieving fine-grained control. This could be refined in future work, e.g., by exploring heuristics such that arrays, vectors, and hash tables can be transformed automatically to implement insert and delete methods.

Our recent work of EFG [22] separates shared data and control components and dynamically composes them at run time. This way the same editing component can be reused with different control protocols for different tasks. However, EFG requires that the editing component be coded following its custom shared component interfaces. It does not address how to transform existing components that fail to observe the custom interfaces. Nonetheless, EFG provides the basic mechanism for implementing dynamic protocols, on which EXEC is based.

3 A Model of Groupware Components

Figure 1 shows our abstract groupware model: A groupware application consists of a multiuser graphical user interface which visualizes shared data for each user and allows the user to interact with the shared data objects; a repository of shared data components that model the application states; and a set of collaboration control services that implement functions such as consistency on the shared data. The data and control components conform to well-defined interfaces such as JavaBeans such that they can be introspected and composed. The infrastructure provides services such as event interception and execution, events broadcasting and notification, data replication and persistence, and session management. It also implements mechanisms for dynamic composition of data and control components as well as user interfaces for achieving so.

In this section, we will model shared data components, present an embodiment in JavaBeans, and then give a brief summary of the runtime system. In the next two sections, we will show how to transform components that do not observe our custom component interfaces.

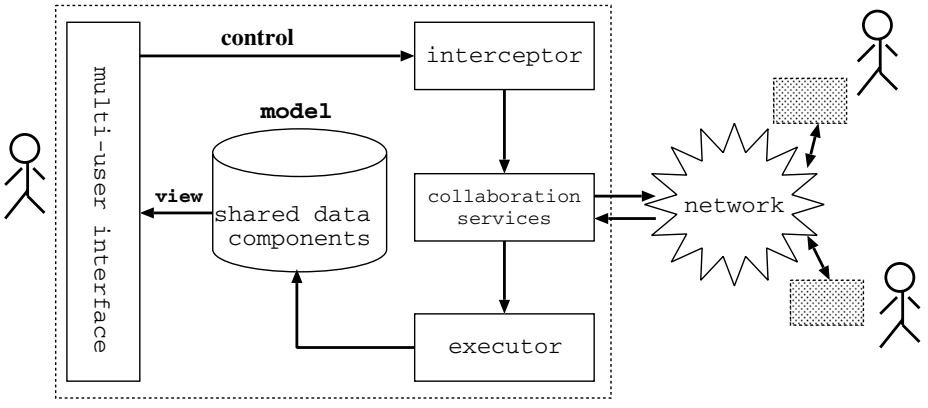


Fig. 1. The groupware component model in EXEC

3.1 Modeling Shared Data

Shared data components are expected to have the following attributes:

First, as in any distributed environment, the infrastructure must be able to uniquely identify different instances of the same shared data component, because property changes must be eventually applied on all replicas to maintain consistency across cooperating sites.

Second, a shared component must provide a well-known way for the infrastructure to intercept its shared property changes before they actually take effect. This happen-before interception is necessary to allow for both optimistic (aggressive) and pessimistic (conservative) control.

Third, a shared component must define a well-known way for the infrastructure to apply desired property changes.

Fourth, a shared container component that has composite structures must provide a standard way for the infrastructure to introspect and access its subordinate components.

Fifth, to implement flexible and efficient control, a shared component should provide a well-known way for the infrastructure to intercept and apply shared property changes both atomically (to replace an old value with a new one) and accumulatively (to model, e.g., the increase and decrease of text), if it contains shared properties that are incrementally changeable.

3.2 Java Embodiment

The above requirements can be easily fulfilled in any modern programming language that supports components. Here we presents an embodiment in Java, which includes two interfaces, *ISharedComponent* and *ISharedContainer*, the latter extending the former. The reason for defining them as interfaces instead of classes is that Java allows only single inheritance in sub-classing. Using interfaces allows more flexibility when the user wants to convert existing components

```
public interface ISharedComponent {
    //(1) to return the global unique id of shared component
    public String getOid();

    //(2) to modify shared properties
    public void insert(String propertyName,
            int offset, Object value);
    public void delete(String propertyName, int offset);
    public void update(String propertyName,
            Object oldValue, Object newValue);

    //(3) to hook collaboration services
    public void addSharedPropertyChangeListener(
            ISharedPropertyChangeListener p);
    public void removeSharedPropertyChangeListener(
            ISharedPropertyChangeListener p);
    public void fireSharedPropertyChange(
            ISharedPropertyChangeEvent e);
}
```

Fig. 2. Shared component interface

into shared components. We include a default implementation called *DefaultSharedComponent*, which extends the JDK *JComponent* class and implements the *ISharedComponent* interface. It provides a starting point for building fresh new shared components.

Figure 2 shows a Java specification of the shared component interface. The methods defined in this interface fall into three groups. The first consists of a method, *getOid*(), that returns the global unique id of the shared component. By this method, each shared component instance identifies itself globally. The second group of methods is used for the runtime system to cause property changes (insert, delete, and update) on this component. They indirectly define the shared properties. The third group includes three methods: the first two allow the runtime system to register (deregister) itself to (from) the shared component for intercepting property changes, respectively, while the third method is for happen-before notification of property changes. Definitions of *ISharedPropertyChangeEvent* and *ISharedPropertyChangeListener* resembles the standard interfaces of events and listeners in Java.

The *ISharedPropertyChangeEvent* class wraps up the event type (insert, delete, or update) and their corresponding parameters into one object. Whenever a method (insert, delete, or update) of a shared component is invoked, this shared component invokes its *fireSharedPropertyChange* method to fire a corresponding *ISharedPropertyChangeEvent*. This method iterates all registered *ISharedPropertyChangeListener* instances and then invokes the well-known notification method defined by the *ISharedPropertyChangeListener* interface. The runtime is itself a *ISharedPropertyChangeListener* instance which is registered

```
public interface ISharedContainer extends ISharedComponent {
    public void insertChildren(int offset, ISharedComponent child);
    public void deleteChildren(int x);
    public ISharedComponent getChildren(int x);
}
```

Fig. 3. Shared container interface

to be a listener of all shared component instances. Whenever a *SharedPropertyChangeEvent* is fired, the runtime system will be notified before the property change takes effect on the shared component. Then the runtime delivers the event to corresponding collaboration services.

We distinguish two types of shared properties: Atomic properties are defined by the *update* method, and accumulative properties are defined by the *insert* and *delete* methods. Atomic and accumulative properties directly map to the atom and indexed properties, respectively, in JavaBeans and .Net. Hence our shared property model retains the expressive power of the original host component model. While all properties can be treated as atomic properties, by distinguishing some of them as accumulative properties we can achieve more fine-grained control and better system performance.

Figure 3 specifies a shared container interface for modeling data objects that have composite structures. A shared container is also a shared component. Hence it extends the *ISharedComponent* interface. Additionally, it defines three methods for retrieving, inserting and deleting subordinate components. By this definition, collaboration services such as consistency can be applied on more efficiently based on the knowledge of the logical structure of the shared component. For example, when the component hierarchy is known, locking can be applied on specific components as well as branches of the tree structure. Note that the pair of methods, *insertChildren* and *deleteChildren*, effectively define a shared accumulative property called "Children". They directly correspond to the *insert* and *delete* methods in the *ISharedComponent* interface. The other methods in the *ISharedComponent* interface are inherited in *ISharedContainer*.

3.3 Runtime System of EXEC

As shown in Figure 1, the EXEC runtime system implements the proposed component model as follows. The *interceptor* registers itself as a shared property change listener to all shared data components as they are plugged in. Whenever the application triggers a shared property change, the interceptor is notified first. Then the property change is pushed into related collaboration service components (e.g., consistency protocol) for processing. The processed property change is then delivered to the *executor* to be applied on the shared component. Because property changes are intercepted before taking effect, we can implement both pessimistic and optimistic concurrency control.

The shared property change event is a tuple of (*OID*, *PropertyName*, *PropertyChangeType*, *Parameters*). Based on *OID*, the executor locates the shared

component instance. Based on the other three elements in the event tuple, the executor decides the execution method (insert, delete, or update) signature for changing the property. Finally, the executor invokes the method on the target property dynamically.

As long as a Java component conforms to the shared component model, it can be used for composing collaboration tools in our framework. Due to the clean separation between data and control, the runtime is table-driven: The association of data and control components is stored in a table. As a result, different control protocols can be dynamically associated with different data objects in the same workspace, and the same object can be dynamically associated with different protocols over time. Note that "objects" here can be individual properties, components as well as containers.

Notably the mechanisms required for implementing the proposed groupware component model, e.g., introspection and dynamic method invocation, are widely available in modern industry component technologies such as JavaBeans and .Net. Hence although the model has been prototyped only in Java, the same results can be achieved on other platforms as well.

4 Transforming Industrial Components

There are a number of client-side component technologies, such as COM, ActiveX, .Net and JavaBeans. Their vendors as well as third parties provide an ever-growing base of reusable components for developing applications. Unfortunately, as analyzed in Section 2, traditional approaches to leveraging those components for building groupware cannot achieve the desired level of automation and flexibility. In this section we describe a simple transformation tool for transforming existing JavaBeans components such that they conform to the shared data component interface defined in Section 3. As a result, the myriad of existing and emerging components can be reused for developing groupware applications with little effort.

The transformation tool converts components that follow the standard JavaBeans naming conventions into shared data components as defined above. It is worth noting that the tool does not need the source code of the input components. Instead, it generates a subclass of an input component directly from its byte code. The subclass implements the *ISharedComponent* interface.

The transformation of a component generally takes four steps: First, the user is prompted to specify the name of the source component and the name, package, and output directory of the target component. Second, properties of the source component are introspected and presented in a dialog. By default, all properties are shared and atomic. From the dialog, the user can deselect properties not to be shared and toggle the types of some properties to accumulative. Third, the target component source code is output to the specified directory by incorporating provided template code. Finally, if necessary, the user can manually adapt the source code, e.g., by adding new shared properties that are not defined in the

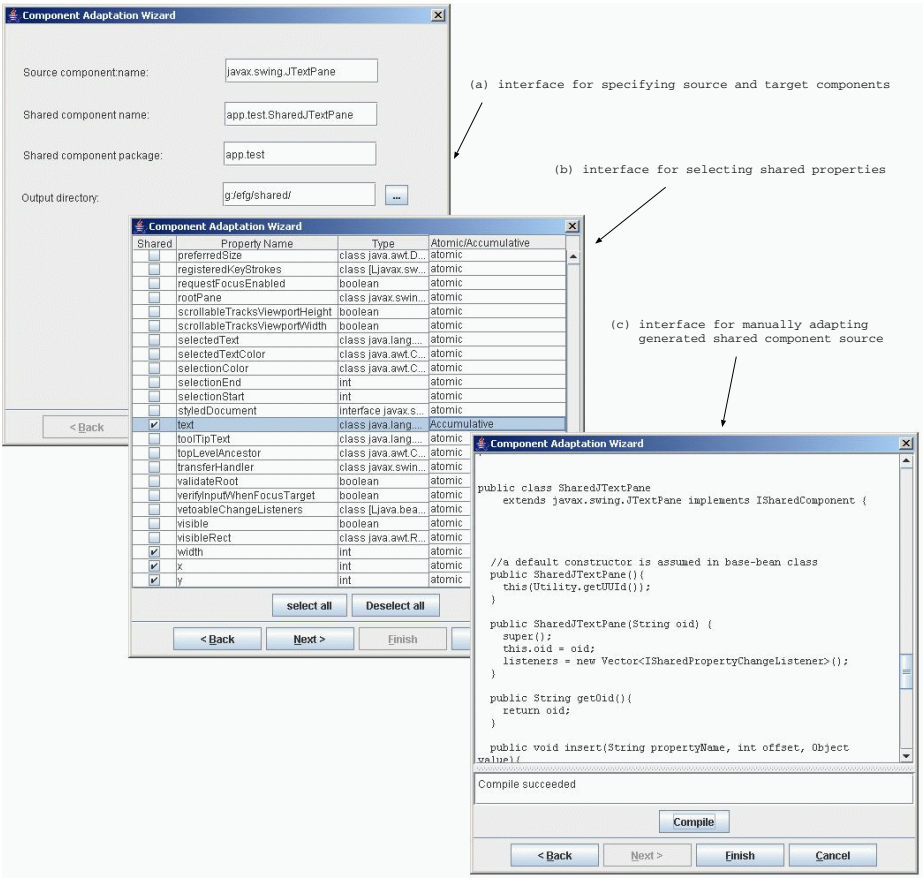


Fig. 4. The component transformation interfaces

original source component. The user interfaces in the first, second, and fourth steps are shown in Figure 4.

Code to implement the shared component interface is actually very simple. Because Java does not allow for multiple inheritance, we provide the implementation in templates. Corresponding to the above *ISharedComponent* and *ISharedContainer* interfaces, we provide four default implementation templates: *SharedComponent*, *SharedContainer*, and two *BeanInfo* classes which describe the shared properties of corresponding components.

As an example, Figure 5 shows the template source code of the shared component class. It contains special markups in the form of @*markup*@, each of which is replaced with actual values when the target component source code is generated. Most markups are configured in the beginning of the transformation process, including the package name, shared component name, and base

```
@packagename@

import framework.*;
import java.util.Vector;
import @BaseComponentName@;

public class @SharedComponentName@
    extends @BaseComponentName@ implements ISharedComponent {

  @VirtualProperties;@

  public @SharedComponentName@(){
    this(Utility.getUUId());
  }

  public @SharedComponentName@(String oid) {
    super();
    this.oid = oid;
    listeners = new Vector<ISharedPropertyChangeListener>();
  }

  public String getOid(){
    return oid;
  }

  public void insert(String propertyName, int offset, Object value){
    fireSharedPropertyChange(new SharedPropertyChangeEvent(
        SharedPropertyChangeEvent.INSERT, oid, propertyName, offset, null, value));
  }

  public void delete(String propertyName, int offset){
    fireSharedPropertyChange(new SharedPropertyChangeEvent(
        SharedPropertyChangeEvent.DELETE, oid, propertyName, offset, null, null));
  }

  public void update(String propertyName, Object oldValue, Object newValue){
    fireSharedPropertyChange(new SharedPropertyChangeEvent(
        SharedPropertyChangeEvent.UPDATE, oid, propertyName, 0, oldValue, newValue));
  }

  public void addSharedPropertyChangeListener(ISharedPropertyChangeListener l){
    if(!listeners.contains(l)) listeners.add(l);
  }

  public void removeSharedPropertyChangeListener(ISharedPropertyChangeListener l){
    if(listeners.contains(l)) listeners.remove(l);
  }

  public void fireSharedPropertyChange(SharedPropertyChangeEvent e){
    for(ISharedPropertyChangeListener l: listeners){
      l.sharedPropertyChange(e);
    }
  }

  protected Vector<ISharedPropertyChangeListener> listeners;
  protected String oid;

}
```

Fig. 5. Template source code of the shared component implementation

component name. The generated shared component class inherits the original component class and implements the *ISharedComponent* interface.

In Figure 5, the @*VirtualProperties*@ markup is replaced by code for implementing heuristic virtual properties such as *boundbox* in shared graphical user interface components, which will be explained in the next section. The main methods for modifying shared properties and hooking collaboration services implement those defined in Figure 2. The *listeners* vector is used to store registered *IShareProperty ChangeListener* instances. Variable *oid* is the id of the shared component instance. Both *oid* and *listeners* are initialized when the shared component is instantiated. The default constructor invokes *getUUID()*, which is a utility function for generating the global unique id.

5 Experience and Framework Evaluation

We have done extensive experiments to study the extent to which our framework can be used for prototyping collaboration tools. In this section, we present what we learned in the experiments. In particular, we discuss what can be automated and what cannot, how some typical tools are built, and the level of flexibility that is achieved in our work.

5.1 Automatic and Manual Adaptation

Using our transformation tool, we automatically transformed all Swing components (derived from the *JComponent* class) and all AWT components (derived from the *Component* class) that come with JDK 1.5. The experiments used default settings, e.g., all original properties are treated as shared properties and all shared properties are atomic. All the generated shared component source passed Java compilation without any exception.

However, we note several problems with automated transformation that entail manual intervention by the developer. These problems are not necessarily limitations of the transformation tool or the proposed approach. Some of them may disappear as new conventions are established in JavaBeans, while some others must involve context-sensitive decisions by human users. Ultimately, there is always a limit with automation; only the users (developers) know what they want and how to implement certain features more intelligently in specific collaboration tools. We document our experience as follows.

First, automatic transformation typically results in an overwhelmingly **large number of shared properties**. For example, a converted *JButton* component contains as many as 87 shared properties (due to that number of original properties). AWT components have relatively fewer properties than Swing components. Nevertheless, in most practical situations, not all the properties of a component need to be shared. Delivering a lot of unnecessary property changes may degrade performance and overwhelm the users. Current component models such as JavaBeans have no standard convention for describing the semantics of properties. It is impossible at this stage to automatically decide shared properties. In fact, even the same property of a component may be shared in one application while not shared in another. Only the developer knows whether a property should be shared in a specific application. Our transformation tool allows the developer to determine shared properties via a simple spreadsheet-like user interface.

Secondly, manual adaptation is necessary when the developer wants to add **additional shared properties** that do not exist in the original component. Sometimes even though the properties do exist, the developer may still want to create virtual properties to simplify control. For example, each Swing component has properties *X*, *Y*, *Width*, and *Height*, which describe the relative coordinates of the component in its container. Instead of using these four properties directly, we may create one virtual property called *Boundbox* that logically congregates them, the value of which can be modified by direct manipulation. As this kind

of adaptation often happens on user interface components, we provide the virtual property and its implementation in separate templates for the developer to choose during the transformation process. However, in general manual coding is required to create new properties.

Thirdly, a few JDK components cannot be transformed automatically to implement accumulative properties for **fine-grained sharing**. Take the "text" property of the *JTextPane* component as an example. By default, it is adapted as an atomic property implemented by an *update* method. The transformation tool can automatically create *insert* and *delete* method signatures to implement it as an accumulative property. However, it turned out that this simple-minded automation failed to work properly due to some hidden issues, which will be explained in Section 5.2. Fortunately, this kind of manual adaptation is only required for a few JDK components and the amount of work is minor.

Additionally, while our framework aims for supporting fast-prototyping of collaboration tools, there are problems that cannot be solved in a once-and-for-all manner. As shown in Figure 1, our framework includes graphical user interfaces to visualize shared components and for users to change their properties. However, different properties might need customized ways for users to change their values intuitively. For example, a color property needs a special color selector, while a string-based property needs a text editor. To mitigate this problem, we provide a spreadsheet-like shared property editor (adopted from Sun BeanBuilder), as shown in Figure 6 bottom left corner, which supports a few basic types of properties, such as numeric, string and color. Nonetheless, it is possible that there are shared properties of non-basic types that are not supported by BeanBuilder, and the property editor, mostly designed for developers, may not be intuitive for end users. Per-application adaptation is generally necessary to achieve **refined user interface design**.

5.2 Building and Adapting Components

We surveyed about 40 commercial and research systems and identified several common groupware tools, such as group text editors, group sketch, group calendar, group browser, and group todo-list. We did experiments to evaluate how easy it was to prototype these typical groupware tools under our framework. For each tool, the general strategy is to first find (or develop) a corresponding single-user component and then transform it into a shared component. The experiments are described as follows.

Group Text Editor. A group text editor allows users to edit a shared document together. Our objective is to adapt some existing component into a shared component following our shared component model. There are three built-in text components in JDK: JTextField, JTextArea, and JTextPane, among which JTextPane is the most sophisticated component supporting both plain text document and styled text document such as HTML and RTF. We chose JTextPane.

The component transformation tool can directly convert the JTextPane component into a shared component. Original properties, e.g. *x*, *y*, *width*, *height*,

background, and even the *text* content itself, can be translated into shared properties automatically. Then we plug the resulting component into EXEC platform for sharing. However, the preliminary experiment revealed some usability problems: it turned out that the automated implementation of the shared text property did not work properly. We explain the problems and their solutions.

First, the automatically generated code failed to achieve happen-before interception. Due to the well-known model-view-controller (MVC) design, the *JTextPane* component uses the *StyledDocument* component as its model. Text changes first happen in the model, which in turn notifies JTextPane to retrieve the new value for its text property. Hence the value of the text property in JTextPane is always a happen-after state if extra care is not taken. Fortunately, the *StyledDocument* class provides a method *setDocumentFilter*, through which we can hook a custom document listener to implement happen-before interception of the accumulative text property. The custom code simply fires a shared property ("text") change and then exits. This way JTextPane is able to intercept atomic as well as accumulative changes before they take effect on the model.

Second, the above-adapted JTextPane component still suffered from a subtle side effect on the user interface. Specifically, after a change is caused by the consistency control component, the caret is not moved accordingly, which causes subsequent changes by the local user to happen at wrong positions. This is, again, due to its MVC design: The view is implemented by the *JTextPane* class itself, while the model is implemented by the *StyledDocument* class. When property changes work around the *JTextPane* methods, the caret position maintained in *JTextPane* is thrown out of sync. Therefore, in the automatically generated *insert* and *delete* methods, we need to add a few lines of code for adjusting the caret position every time after an incremental property change is applied.

The deeper reason is that many JDK components separate their model and GUI delegates on purpose in order to reuse both. For example, the same JTextPane component can use different document models, e.g., RTF, HTML, and plain text. However, while achieving reuse, this separation makes it difficult for GUI delegates to wrap up the model events as their own events for, otherwise, they would be bound with one model and can not be reused with others.

Fortunately, the kind of indepth plumbing work is only required on a few (text editing related) JDK components in order to achieve fine-grained sharing of incremental state changes. The manual adaptation is done via well-documented APIs and does not require analysis of source code. To implement the above explained manual adaptation of the JTextPane component, it only takes about 100 lines of code (including moderate comments) in total.

Group Sketch. A group sketch tool allows multiple users to draw graphic shapes such as scribbles, lines, and circles on the canvas of a shared workspace. In our approach, the problem is reduced to finding or developing a corresponding single-user component. Since there is no single-user sketch in JDK, we need to build one from scratch, which takes two steps.

First, a base shape component is implemented with common properties such as *lineStyle*, *lineWidth*, *lineColor*, *filledColor*, and *bounds*. The *bounds* property

defines a minimum bounding box which covers the shape. By directly manipulating the bounding box of a shape, its position parameters such as location, width, and height are changed automatically.

Next, we build shape components, including lines, circles and rectangles, by inheriting the base shape component. The difference between them is very minor, mainly in that they have different paint functions for drawing the corresponding shapes. The base component takes around 100 lines of code to implement the setter and getter methods for the common properties. Each inherited shape simply overloads the *paint* function, which is about 10 lines of code.

The single-user shape components are transformed into shared components automatically. This experiment demonstrates that, when needed single-user components are not available, we first build them and then transform them into shared components. With support by the EXEC runtime system, flexible sharing can be achieved without the burden of implementing collaboration functions.

Group Browser. A group browser allow multiple users to browse web pages together. There is no single-user web browser component in JDK. Components such as JEditorPane and JTextPane support HTML documents browsing but they do not have address bars and history mechanisms as in web browsers. We chose to extend JTextPane into a single-user web browser such that it allows users to type in new URLs and navigate back and forth.

The extension only adds an address bar (*JTextField*), two buttons (*JButton*) for backward and forward navigation, and three properties: *bound*, *currentURL* and *history*. Whenever a user types in a new URL in the address bar or clicks on a web link or the backward/forward button, the *currentURL* property is set to the corresponding URL. Building this single-user browser takes around 100 lines of Java code. Note that the adaptation in this case is very simple because there is no need to implement the aforementioned fine-grained sharing on the "text" property as in a group text editor.

Group Calendar. A group calendar allows users to browse, add and remove tasks together. This is convenient for collaborators to detect schedule conflicts. Almost all meeting software and enterprise applications have some form of group calendar. There is no single-user calendar component in JDK. We implemented a single-user calendar component as follows: First, we found an open-source component (about 350 lines of code) on the Internet that was able to display a calendar and allowed the user to pick dates. Next, we extended the source code by 250 lines, which implemented task management functions such as displaying, adding, removing tasks associated with a specific date. The adaptation simply declares the *events* property of the self-built single-user component as an accumulative property, which requires the developer to manually add a few lines of code to implement the inserting and deleting of *events*.

Group Todo-List. A group todo-list allows multiple users to browse and maintain a list of tasks together. The tasks might or might not be associated with due dates. Unlike a calendar which organizes activities by dates, a todo-list organizes activities by themselves.

There is no single-user todo-list in JDK. So we developed a simple one which is but a list with three columns: a checkbox to indicate whether a task has been completed or not, a description of the task, and the due date of the task. This single-user todo component takes about 300 lines of code, which is very straight-forward. Similarly to the adaptation of the *events* property in the calendar component, we share the *tasks* property of todo-list component as an accumulative property. The developer needs to manually add a few lines of code to implement the function signatures for inserting and deleting *tasks*.

5.3 An Integrated Evaluation Environment

Now we use an integrated example to demonstrate how the above adapted JDK components and the five groupware tools can be used for supporting flexible collaboration. The graphical user interface of our experimental system, EXEC, is shown in Figure 6. It is intended to be a collaboration platform for supporting our daily collaborative research and education activities, focusing on exploring the following two research ideas: First, its functionality can be continuously extended, by transforming single-user components into new collaboration tools (as in the presented work) and by sharing familiar single-user applications without modifying source code (as in [13,14]). Secondly, shared data components can be flexibly controlled by a variety of collaboration protocols, including those for session management, awareness control, access control, consistency maintenance, and processes and procedures (as in [22]). Protocols control data at various granules (property, component, and workspace) and can be switched dynamically at run time to cater for evolutionary collaboration needs. EXEC provides a unified testbed for a variety of projects in our group.

In Figure 6, panels on the left side are for control and the one on the right side is the current workspace. The users can create workspaces, which are but container components that contain other components. Workspaces are organized in a tree structure, which is displayed on the upper left panel. By selecting different workspaces in the hierarchy, the user implicitly switch between different collaboration sessions, similar to [4]. A presence awareness panel on the middle left lists users who are in the current workspace or in the whole system. The infrastructure provides persistency and communication services, which are implemented via a storage and notification server. All workspaces and user interaction events are logged and persisted on the storage server automatically. Events are delivered to collaborating sites by the notification server.

The vertical toolbar lists available groupware components and tools. When a user clicks on a toolbar icon, a new instance of the corresponding shared component will be created and inserted into the current shared workspace. We have implemented a library of shared components by transforming JDK, third-party and self-built collaboration-transparent components. The screenshot displays a few components transformed from JDK and the five tools discussed in Section 5.2: group text editor, group sketch, group calendar, group todo, and group browser. The workspace is also a sketch tool itself, on which the user(s) can draw graphics in individual or in group.

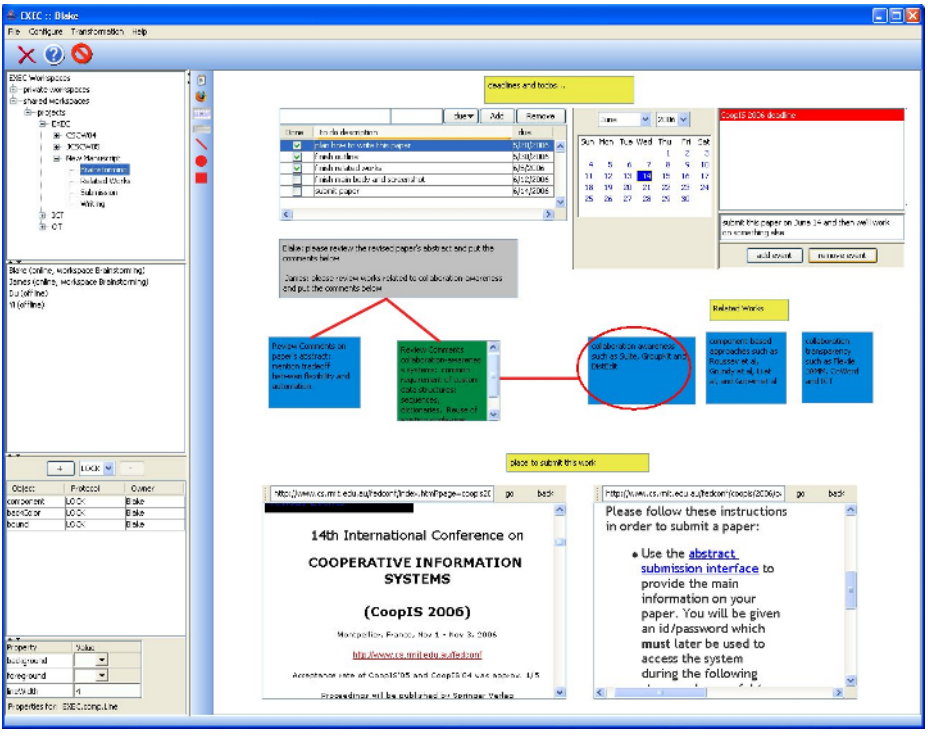


Fig. 6. A screenshot of EXEC showing the brainstorming workspace of this paper

As shown on the bottom left panel, a spreadsheet-like user interface is provided for users to apply and switch object-level protocols. An object here means a property, a component, or a container. Different properties of the same component can be controlled by different protocols. If a protocol is applied on a component, updates on all properties of that component will be controlled by the same protocol. A protocol applied to a workspace (container) will control all components in that workspace (container).

As shown in Figure 1, whenever an update event is intercepted on an object, it is routed to the protocol associated with the object. Depending on the level of optimism implemented, the protocol either first applies the event on the local object and then communicates with collaborating sites to resolve conflicts (optimistic), or communicates with collaborating sites first to make sure that the proposed update is safe and then applies it locally (pessimistic). Intermediate levels of optimism are possible [22].

The screenshot in Figure 6 shows the following example scenario. The EXEC environment is used for projects management and papers writing in our group. In the writing of this paper, the process is organized into several phases: brainstorming, related work review, group writing, and submission. In the brainstorming phase, objects in the workspace are controlled by optimistic protocols such that coauthors are allowed to modify anything at any time [9]. In the literature survey

Table 1. Efforts to prototype typical groupware tools

groupware tools	single-user component source	lines of manual code
group text editor	JTextPane	120
group sketch	self-built	100+(10/shape)
group browser	extended from JTextPane	100
group calendar	extended from third-party source	250
group todo-list	self-built	300

and the group writing phases, the work is divided among coauthors and the subworkspaces are locked by members who are responsible for them, respectively. In the submission phase, the whole workspace is locked and members take turns to smooth out the draft until it is submitted. The brainstorming workspace is shown in Figure 6, in which the collaboration tools are used for coordinating the writing of this paper: The group calendar and todo list are used for describing deadlines and expectations; The browsers show possible places to submit this work; The texts and graphics show a raw division of tasks among the coauthors with visual annotations. The control panel shows a component on which a locking protocol is applied for concurrency control: because the component is locked in entirety, all properties are locked.

5.4 Discussions of Work and Benefits

In general it takes little work to transform existing single-user components into shared components. The key issue is to find the right single-user components. However, even if a needed component is not available, *with collaboration functions offloaded*, building a single-user component is generally far less complicated than implementing its multi-user counterpart.

The experimental results are summarized in Table 1 These five tools together only took one person (the second author) two weeks, which, of course, does not include the time to develop the runtime infrastructure and the transformation tool. It is worth noting that this work aims to serve the purposes of fast prototyping. It would certainly take more lines of code if more complex functionality and polished user interfaces were required. However, this extra work does not undermine the validity of our framework, which transforms collaboration-transparent components for prototyping flexible collaboration tools.

Under the proposed framework, most single-user components that conform to industry standards such as JavaBeans can be automatically transformed into shared components. With no extra coding effort except some simple configuration, e.g., to provide output component names or to select shared properties, much flexibility can be achieved with the generated component. Due to the happen-before interception of property change events, both optimistic and pessimistic control can be applied on shared components and their properties.

With some minor manual coding work, e.g., to add some additional shared properties and to adapt some accumulative properties, much more flexibility and efficiency can be achieved. For example, after JTextPane is tweaked to support

fine-grained insert and delete notifications, consistency protocols alternative to locking and serialization, such as operational transformation (OT) [19], can be applied on the "text" property. With OT, multiple users are allowed to edit any part of a shared accumulative property simultaneously in a nonblocking manner and local response time is not sensitive to networking latencies.

Moreover, the infrastructure allows for the following extra flexibility that is not found previously to the best of our knowledge [22]. First, different shared properties of the same shared component can be controlled by different protocols. For example, the *bounds* property of a group editor component can be locked to fix its position and size in the workspace, while the *text* property can be controlled by OT to allow for concurrent editing of content. Secondly, control protocols can be applied at the property level as well as any congregate of properties. For example, at the component level, it means the same protocol is applied on all shared properties of that component, and at the workspace level, it means a protocol applies on all components in that workspace. Thirdly, different consistency protocols can be applied on the same object (property, component, workspace, etc.) dynamically over time. For example, in a brainstorming meeting, a group editor may be first controlled by OT to encourage free contribution and then be applied a turn-taking protocol to smooth out the meeting minutes.

6 Conclusions

Our work reduces the problem of developing collaboration tools to developing and reusing collaboration-transparent components in which collaboration functions are offloaded. A groupware component model is proposed which distinguishes components of shared data and collaboration services. With the support of a runtime infrastructure, these components can be independently developed and dynamically composed for prototyping flexible collaboration tools. Moreover, a simple transformation tool is provided such that the myriad of collaboration-transparent components can be transformed into shared data components. As evidenced by our experiments, all JDK components can be transformed for building collaboration tools. Even automatic transformation of components can achieve flexible sharing, e.g., dynamic object-level consistency control. Usually only minor manual adaptation on the transformed components is required to achieve fine-grained control and refined user interface effects.

We plan to extend this work into a full-fledged collaboration platform (EXEC) by leveraging single-user components as far as possible. In particular, we will investigate how user interfaces for manipulating groupware components can integrate seamlessly into the platform, as in [6], and how workspace awareness information can be automatically collected and presented in groupware components, complementary to [8]. Next, we will investigate how existing methods in software engineering, such as [15], can be extended to help users find needed components for developing groupware applications.

Acknowledgment

This work was supported in part by the National Science Foundation under CAREER award IIS-0133871.

References

1. J. B. Begole, M. B. Rosson, and C. A. Shaffer. Flexible collaboration transparency: supporting worker independence in replicated application-sharing systems. *ACM Transactions on Computer-Human Interaction*, 6(2):95–132, June 1999.
2. C. Cook and N. Churcher. Constructing real-time collaborative software engineering tools using CAISE, an architecture for supporting tool development. In *Proceedings of the 29th Australasian Computer Science Conference*, Jan. 2006.
3. P. Dewan and R. Choudhary. A high-level and flexible framework for implementing multiuser user interfaces. *ACM Transaction on Information Systems*, 10(4):345–380, 1992.
4. S. Greenberg and M. Roseman. Using a room metaphor to ease transitions in groupware. In M. Ackerman, V. Pipek, and V. Wulf, editors, *Beyond Knowledge Management: Sharing Expertise*, pages 203–256. MIT Press, Cambridge, MA, 2003.
5. J. Grudin. Groupware and social dynamics: Eight challenges for developers. *Communications of the ACM*, 37(1):92–105, 1994.
6. J. Grundy and J. Hosking. Developing adaptable user interfaces for component-based systems. *Interacting with Computers*, 14(3):175–194, Mar. 2002.
7. J. Grundy and J. Hosking. Engineering plug-in software components to support collaborative work. *Software – Practice and Experience*, 32(10):983–1013, Aug. 2002.
8. J. Hill and C. Gutwin. The MAUI toolkit: Groupware widgets for group awareness. *Computer Supported Cooperative Work*, 13(5):539–571, 12 2004.
9. C. M. Hymes and G. M. Olson. Unblocking brainstorming through the use of simple group editor. In *ACM CSCW Conference*, pages 99–106, Nov. 1992.
10. M. J. Knister and A. Prakash. DistEdit: A distributed toolkit for supporting multiple group editors. In *ACM CSCW Conference*, pages 343–355, Oct. 1990.
11. M. J. Knister and A. Prakash. Issues in the design of a toolkit for supporting multiple group editors. *Computing Systems*, 6(2):135–166, 1993.
12. J. C. Lauwers and K. A. Lantz. Collaboration awareness in support of collaboration transparency: Requirements for the next generation of shared window systems. In *Proceedings of ACM CHI'90 Conference*, pages 303–311. 1990.
13. D. Li and R. Li. Transparent sharing and interoperation of heterogeneous single-user applications. In *ACM CSCW Conference*, pages 246–255, Nov. 2002.
14. D. Li and J. Lu. A lightweight approach to transparent sharing of familiar single-user editors. In *ACM CSCW Conference*, Nov. 2006. To appear.
15. B. Morel and P. Alexander. Automating component adaptation for reuse. In *Proc. of the 18th IEEE International Conf. on Automated Software Engineering*, pages 142 – 151, Oct 2003.
16. M. Roseman and S. Greenberg. Building real-time groupware with GroupKit, a groupware toolkit. *ACM Transactions on Computer-Human Interaction*, 3(1):66–106, Mar. 1996.
17. V. Roussev, P. Dewan, and V. Jain. Composable collaboration infrastructures based on programming patterns. In *ACM CSCW Conference*, pages 117–126, 2000.

18. L. Suchman. *Plans and Situated Actions.* Cambridge University Press, 1987.
19. C. Sun and C. Ellis. Operational transformation in real-time group editors: issues, algorithms, and achievements. In *ACM CSCW Conference*, pages 59–68, Dec. 1998.
20. C. Szyperski. Component technology – what, where, and how. In *Proceedings of International Conference on Software Engineering*, pages 684–695, Portland, Oregon, May 2002.
21. S. Xia, D. Sun, C. Sun, D. Chen, and H. Shen. Leveraging single-user applications for multi-user collaboration: the CoWord approach. In *ACM CSCW Conference*, pages 437–446, Nov. 2004.
22. Y. Yang and D. Li. Separating data and control: Support for adaptable consistency protocols in collaborative systems. In *ACM CSCW Conference*, pages 11–20, Nov. 2004.

Evaluation of a Conceptual Model-Based Method for Discovery of Dependency Links

Darijus Strasunskas and Sari Hakkarainen

Dept. of Computer and Information Science,
Norwegian University of Science and Technology, NO-7491 Trondheim, Norway
{dstrasun, sari}@idi.ntnu.no

Abstract. In practice dependency management often suffers from labor intensity and complexity in creating and maintaining the dependency relations. Our method targets projects, where developers are geographically distributed and a wide range of tools is used. A conceptual domain model is used to inter-relate the development objects and to automate dependency link discovery. The proposed method is based on association of development objects with concepts from domain model. These associations are used to compute dependency among development objects, and are stepwise refined to direct dependency links.

A preliminary empirical evaluation of the method is conducted. The method is evaluated both on performance and psychological variables. The evaluation has been performed in laboratory settings using two real cases. The results, although preliminary, provide positive evidence about the ability of our method to automate discovery of dependency relations, the analysis indicates that the method is perceived to be easy to use and useful by its potential users.

Keywords: Conceptual model-centric development, geographically distributed development, cooperative systems development, dependency management.

1 Introduction

Traditional systems development has been centered on small projects and co-located teams using a limited toolset. Today, the organisations meet a large number of stakeholders, broad geographical distribution alongside a wide range of tools [24]. The output from a systems development process ranges from requirements specification and architecture to design and code. Typically, all these parts should be integral and related. Thus, the end product of IS development is not a homogeneous specification, but rather a collection of loosely correlated *product fragments* (e.g., requirements specification, design, code, test scenarios, and documentation).

Distributed development projects have special settings and needs. One problem here is the management of product fragments diversity; discovery and maintenance of dependency links among the heterogeneous product fragments. However, dependency management often suffers from an extensive effort and complexity of creating and maintaining the traces [4]. To solve the problem, we have proposed to use conceptual model not only to guide the design of a system, but also to actually access and manage the information produced during the IS development. We propose to use a

R. Meersman, Z. Tari et al. (Eds.): OTM 2006, LNCS 4275, pp. 625 – 642, 2006.

conceptual model throughout whole IS development life-cycle [20], i.e. we advocate for conceptual model-centric IS development [12]. In our method, associations of product fragments using concepts from a domain model constitute the base to calculate a semantic distance between the product fragments [20, 22]. The computed semantic distance is further used to assess change impact and establish dependency links between the product fragments.

Utility of the method has been evaluated with respect to the state-of-the-art in [20, 22]. There it was argued that the method is capable to solve the problem of dependency management in a geographically distributed development. While, the utility is demonstrated in application of the method in a prototype, called CO_2SY (COOperative SYstem), implementation [20]. The contribution of this paper is an empirical evaluation of the method and its implementation by presenting and discussing the results from a laboratory experiment that investigates whether the proposed method achieves its objectives. The goal of the preliminary experiment is to evaluate the method and implemented prototype, receive hints for further method improvement. In order to achieve the goal, we analyse its effectiveness and likelihood of adoption in practice from the point of view of possible users. The paper addresses broad evaluation research questions as follows.

RQ1. Is the method potentially effective? That is, does the method facilitate dependency establishment among product fragments and helps to explore relatedness of product fragments and discover "hidden" dependencies?
RQ2. Is the method apt to be adopted in practice? That is, what are users' perceptions about method usefulness and ease of use?

The method is evaluated by the means of the experiment and a post-task questionnaire. The analysis performed using the Method Evaluation Model (MEM) [11]. As our method is user-centric, we have chosen to test users' perceptions regarding ease of use and intention to use. Usefulness of the proposed method is tested analysing observed efficiency and perceived usefulness. In addition, we want to collect the responses for further possible improvement of the method and CO_2SY.

The paper is structured as follows. In Section 2 our method is presented. In Section 3 the evaluation method and design of the experiment are presented. Then in Section 4 the results are discussed. In Section 5 the cause and threats to validity are analysed, as well our method is compared to related work. Finally, in Section 6 conclusions are drawn and future work is laid down.

2 Conceptual Model-Centric Cooperative IS Development

Success in system development depends on effective human communication [18], where early understanding and modeling of problem domain is a key to managing large scale systems and projects [18]. Furthermore, conceptual modeling is "*the first step and one of the most important steps for application engineering*" [2, p. 297]. Here, we propose to use conceptual models not only to guide the design of a system, but also to actually access and manage the information produced during IS development, in particular for computation of dependency between product fragments. Further, associations of development objects with a concept from a domain model are

used to communicate the meaning of development objects between stakeholders. This requires the stakeholders to reach a certain level of shared interpretation of the domain referred throughout the development [20].

2.1 Overall Method

The goals, the means to achieve them and the functional steps of the method are summarized in Figure 1. The modeling environment [20] is used for collaborative problem modeling and to reach a common conceptualization of a domain. The resulting conceptual domain model is used for product fragment management, and change impact assessment. Since IS development activities are iterative and product fragments are changed many times, we are able to accumulate statistics about fragments dependency. A log of confirmed change impacts serves as a means to refine and establish *direct* dependency links between the product fragments, finally achieving product lifecycle traceability.

More specifically, the method is comprised from four functional steps [20, 22] as follows.

Step1. Developing a conceptual domain specific model. Each developer produces a model fragment based on his/her view. The views are aligned semi-automatically resulting in a common domain model and aligned terminology. The developers still maintain their personal view and preferred terminology as advocated in [6, 20].

Step2. Fragments association with concepts. Each developer uploads a product fragment developed by him/her. While uploading to the repository the developed product fragment is associated with one or more concepts from an earlier defined domain model. An association process here can be treated as a classification of deliverables, i.e. by related the product fragments to the structure of problem domain. Developers can choose confidence level when associating with concepts, e.g., to specify the strength of association.

Step3. Dependency discovery. Thus, dependency relations (relatedness) are based on the semantics of the product fragments. Given that all developed fragments are linked through the conceptual model, there exists a set of domain concepts $\{C_1, C_2, \ldots, C_n\}$

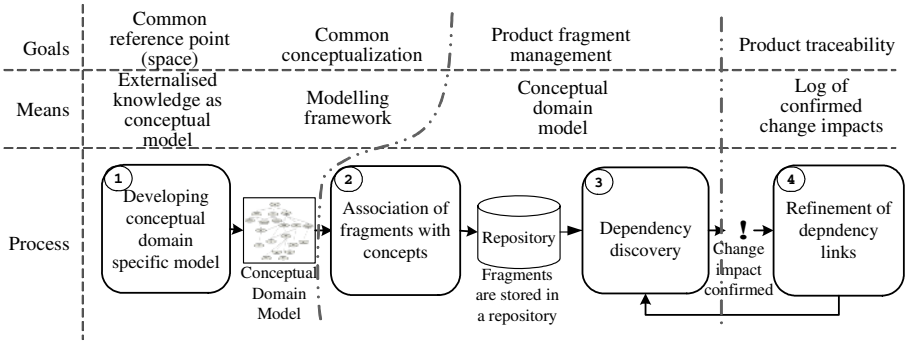


Fig. 1. Overall method: goals, means and process

and a set of product fragments {F_1, F_2, …, F_m}. The fragments are related as follows. First, if product fragment F_i is associated to domain concept C_i and product fragment F_j is associated to C_i, then transitively F_i also relates to F_j. Second, given the related domain concepts C_i and C_j, and product fragment F_i associated to concept C_i and product fragment F_j associated to C_j, then dependency to a certain degree exists between F_i and F_j.

Step4. Refinement of dependency links. Since IS development activities are iterative, the product fragments are changed many times. Statistics on confirmed change impacts enable to refine and establish *direct* dependency links between the product fragments, finally achieving product lifecycle traceability.

Step1 is elaborated in details in [20]. The subsequent steps are more relevant to the experiment reported here; therefore a more detail account for the remaining steps is given in section 2.2. The method has been applied in the CO_2SY prototype [20] implemented in Python with a repository support implemented using PostgreSQL ORDBMS.

2.2 Conceptual Model-Based Dependency Discovery

In this subsection we elaborate on the method part concerning change impact prediction and dependency discovery. Here a common conceptualization of the domain is used to compute relatedness among the product fragments.

Dependency management in practice often suffers from an extensive effort and complexity of creating and maintaining traces [4]. Therefore, given earlier discussed special settings of distributed development, our method is based on a "fuzzy" association of product fragments with concepts in a particular domain model. We tackle the challenge of describing heterogeneous information by semantically enriching the product fragments, i.e., providing means to explicate their meaning through associations with concepts. The facts that concepts are interrelated with each other in the conceptual model and all fragments are linked to domain concepts enable us to compute semantic distance between different fragments. Further, a conceptual model is used to interoperate across different representation formats used in distributed development throughout whole lifecycle.

Each time when a new revision of the product fragment is uploaded into a repository, change impact is computed based on associations of the product fragments with the conceptual model (recall step3 in section 2.1). Developers can specify a confidence level for every association, e.g., specifying the strength of association. For usability sake we use categorical ranges for confidence levels instead of numerical. Three confidence levels are defined. They are as follows. *High* confidence level of association corresponds to numerical value of 0.2; *medium* is represented by 0.5 when computing overall semantic relatedness and *low* confidence level is equal to 0.8.

A semantic distance algorithm [20] (see simplified version in Algorithm 1) is a base for the dependency (change impact) computation. Here we have adapted an approach [23] which is based on computing cosine similarity [1] using concept feature vectors. The feature vectors are constructed from extension of the concepts, i.e. provided natural language descriptions of the concepts or accessed from the WordNet database. Semantic distance (SD) between concepts is computed as follows.

$$SD_{(a,b)} = 1 - sim(C^a, C^b) = 1 - \frac{\overrightarrow{C^a} \times \overrightarrow{C^b}}{|C^a| \times |C^b|} = 1 - \frac{\sum_{i=1}^{n} (C_i^a \times C_i^b)}{\sqrt{\sum_{i=1}^{n} (C_i^a)^2} \times \sqrt{\sum_{i=1}^{n} (C_i^b)^2}} \quad (1)$$

Where, C^a and C^b are feature vectors for concepts a and b; n is the dimension of the feature vectors, $|C^a|$ and $|C^b|$ are lengths of the two vectors. Result of $sim(C^a, C^b)$ is deducted from 1 in order to convert a semantic similarity value to represent a semantic distance. Our method is based on the semantic distance between concepts and product fragments, where smaller value shows concepts (product fragments) being semantically closer, i.e. contrarily to the semantic similarity value, where higher values show concepts being more similar.

Algorithm 1 presents an overall computation of relatedness between the product fragments. First, a semantic distance is calculated between every pair of concepts based on Eq. 1. Last, a path length is computed between an altered product fragment and other associated product fragments. Here, we adopted Dijkstra's shortest path algorithm [3]. Final values are normalized to fall into range [0…1].

Algorithm 1. Semantic distance computation between product fragments

Variables	MF – model fragment; C – a set of concepts in a model fragment MF; TR_c – textual resource for concept c (concept extension), where $c \in C$; PF – a set of product fragments associated with model fragment. PFC – a set of changed product fragments, such as $PFC \subseteq PF$.

Function	∀ a (a ∈ C):
	∀ b (b∈ C):
	If ∃ R(a, b):
	If $TR_a \neq 0 \wedge TR_b \neq 0$:
	$W_{(a,b)}$=sim(a,b) //i.e. Eq.1
	
	∀ pfc (pfc∈ PFC):
	∀ pf (pf∈ PF):
	Return Shortest_path(pfc, pf) //i.e. [3, 22]

The product fragments are altered many times during IS development life-cycle. Each change generates a list of possible impacts, which are either confirmed or rejected. That allows us to accumulate reliable statistics regarding dependency for each pair of the product fragments. The accumulated statistics are used to refine and establish direct dependency (more precise) links between the product fragments, and achieve product traceability at a certain stage of IS development.

3 Evaluation Model and Experimental Settings

Empirical study provides a means to evaluate the efficacy (i.e. both the efficiency and the effectiveness), while feasibility and acceptance of the method are determined by measuring users' perceptions [15]. Therefore, we adopt the Method Evaluation Model

(MEM) [11], a model for evaluating design research methods. The MEM (see Figure 2) incorporates both aspects we are interested to measure: observed (actual) efficacy and likelihood of adoption in practice.

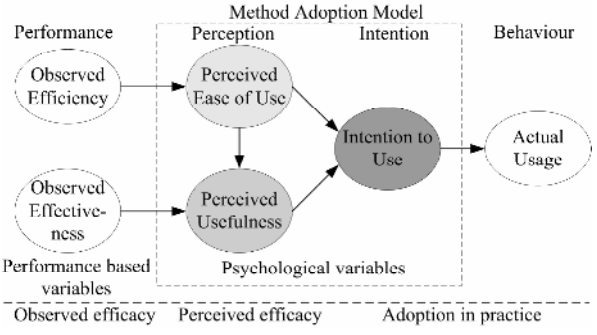


Fig. 2. The Method Evaluation Model (adapted from [11])

The constructs of the MEM are defined as follows. *Observed (Actual) Efficiency* is the degree to which the method reduces the effort required to apply it. *Observed (Actual) Effectiveness* is the degree to which the method improves the quality of the result. *Actual Usage* is the degree to which the method is used in practice. *Perceived Ease of Use* is the degree to which a person believes that using the method would be effortless. *Perceived Usefulness* is the degree to which a person believes that the method would be useful. *Intention to Use* is the degree to which a person intends to use the method.

Observed Efficacy measures whether the method actually improves a task, while Perceived Efficacy represents perceptions of both the method's efficiency and effectiveness. Adoption in practice is determined by perceptions, which in turn are determined by performance [11]. Psychological variables are central constructs and constitute the Method Adoption Model in MEM.

3.1 Experimental Settings

The experiment is designed to evaluate the effects and the usability of the proposed method for dependency assessment in distributed development. The experiment design is summarised in Figure 3, where experimental treatment, experimental tasks, materials and collected data are illustrated. As our method is user centric, we organise the experiment in a way that best records users' perceptions. Despite of user centred measurements being subjective, conclusions can be drawn by measuring consistency and reliability of the answers. In addition, performance measures and observations support the users' feedback. In the following subsections the experiment design is further elaborated as in Figure 3. We start from discussion about participants and experimental treatment, then the experimental materials and used cases are discussed and exemplified. But prior to that, we decompose relevant evaluation research questions (recall section 1) to the hypothesis underlying the experiment.

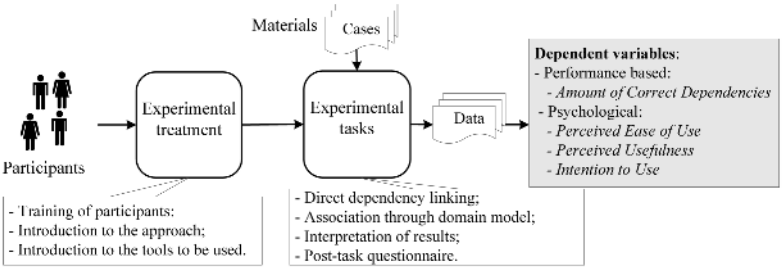


Fig. 3. Design of the experiment

Hypotheses. A priori, based on the MEM constructs the assumed effects of using the domain model to manage relatedness of product fragments and their dependency are hypothesised as follows.

H1: The method is effective, i.e. domain model facilitates dependency establishment among product fragments and helps to explore relatedness of product fragments and discover "hidden" dependencies;

H2: The method is perceived as easy to use, i.e. "fuzzy" linking is easier than direct dependency linking;

H3: The method is perceived as useful; and consequently,

H4: There is an indication of intention to use the method.

Participants selection and experimental treatment. Since the proposed method is intended to be used by a variety of stakeholders, we've selected test subjects with different backgrounds, though all of them from computer science area (information management, databases, information systems, and knowledge management). All 6 test subjects are from Dept. of Computer and Information Science at NTNU. 5 out of 6 subjects have experience of work in IT industry, 3 have participated in big software projects, while one has experience from a big geographically distributed project.

The subjects participated in a 45 minutes long training session. First, the method to be tested together with the overall idea was presented. Then, an evaluation task and the procedure was presented, and discussed at a common meeting. Finally, short tutorials on each of the tools used in the experiment were given.

Couple of subjects was familiar with the concepts of the method and have seen the CO_2SY prototype, but had not been using it themselves. Other subjects had not seen the tool before. The test subjects have extensive computer experience (5 of them have 11-25 years experience, and one 5-10 years experience). Though, none of them is an expert user of the tested tools (see next subsection), in fact only one had experience of using traceability matrix before. None of the users had addressed the problem of dependency management prior to the experiment, with one exception that has earlier been working with dependencies and traceability links.

Experimental material. The instrumentation used in the experiment included experimental materials, tools (CO2SY, Telelogic® Doors™, and traceability matrix implemented in MS Excel™). Further, a log for performance measurement and survey techniques (a questionnaire and "think aloud" protocol) were used.

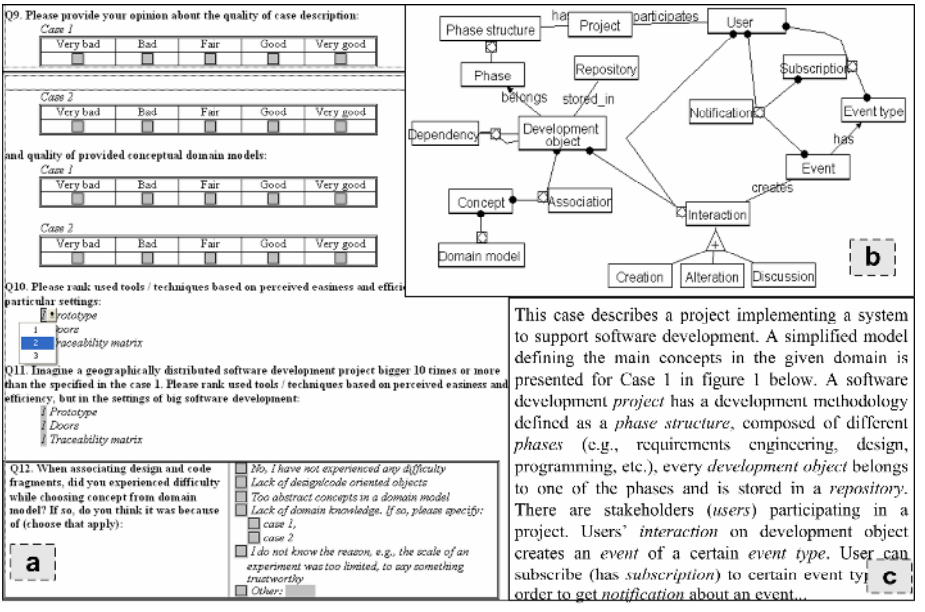


Fig. 4. Example of (a) questionnaire, (b) domain model and (c) case description provided to users

The experimental materials [20, 21] consisted of two cases and their descriptions in natural language, domain models in RML [17], and some diverse product fragments. Case 1 was taken from a MSc project [5], where figures 4b[1] and 4c illustrate a domain model and description of the case 1. The MSc project was similar to the method proposed here, i.e. dealing with dependency and traceability. Case 1 consisted of 4 different product fragment types, i.e., requirement statements (natural language), design fragments (UML™ sequence diagrams), code fragments (C#) and user manual in a form of screenshots (see Figure 5). Case 2 was based on development materials of the MEIS (Model Evaluation Information System) system that is used for an introductory course on information systems [10]. The MEIS system is used for exercise delivery, peer-to-peer review and for the evaluation of both, the exercise and the review. Case 2 had two types of product fragments, i.e. requirements statements (natural language) and code (php).

The conceptual domain models that the researcher designed for the projects had 18 concepts 21 relationships in case 1 and 18 concepts 26 relations in case 2. Case 1 had 83 product fragments, and case 2 had 23 fragments. For both cases, the task assigned to the subjects was to study the materials and afterwards, specify dependency relationships for a set of selected development objects.

Experimental task. The task for test subjects was to do the both, to establish the direct dependency links among fragments and to associate the fragments with concepts from the provided domain model. After dependency between product fragments has been

[1] Here figures 4 and 5 are used for the explanatory purposes.

computed, they needed to choose three random product fragments and investigate three top ranked relatedness values (not items) provided by CO2SY. They were asked to treat the list as indication that product fragments are dependent. They were further asked to investigate the list and to classify the fragments returned by CO2SY as totally wrong (having nothing to do with each other), partially correct (for those that seems to be dependent, but needed more detail investigation), or totally correct (the fragments are dependent).

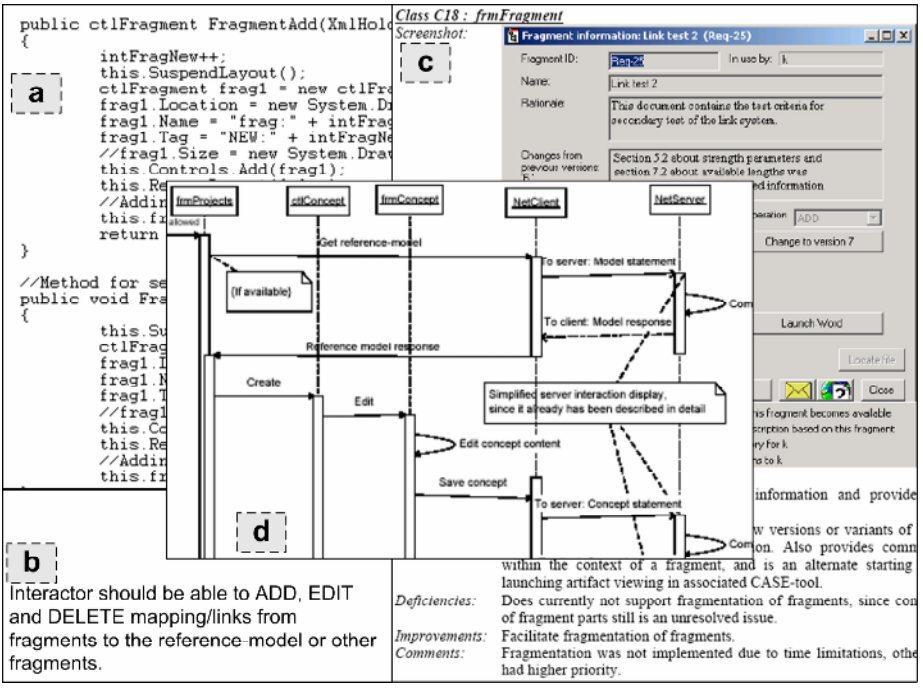


Fig. 5. Illustration of product fragments type in case 1: (a) C# code fragment, (b) requirement statement, (c) user manual (screenshot and description) and (d) sequence diagram

Comparative tools. In addition to the CO_2SY prototype, two tools were used, namely, Telelogic Doors™ and a traceability matrix implemented in MS Excel™. Both consider direct linking of related product fragments, by means of different interfaces. In order not to be biased by graphical user interface, the test subjects were put in two groups. Group A first established direct dependency links using Doors (with case 1 materials) and traceability matrix (with case 2 materials), then associated product fragments using the CO_2SY prototype. While group B first assigned associations using CO_2SY (with materials of both cases), and then established direct links using the comparative tools.

Survey. At the end of the evaluation, the test subjects answered a questionnaire [20, 21] (see Figure 4a) in order to obtain general feedback on the system as well as to discuss its drawbacks. To gain a deeper insight into subjects' views, "think aloud"

protocol was used, where subjects were asked to think aloud while performing the tasks. Most of the questions were closed. The questionnaire included 21 closed questions, 3 closed questions with unordered responses, and 3 open questions. A five-point Likert scale was used to measure 13 out of 21 closed questions, i.e. subjects were asked to express agreement or disagreement of a five-point scale. Each degree of agreement is given a numerical value from one to five. Thus a total numerical value can be calculated from all these responses.

3.2 Dependent Variables

We distinguish two types of dependent variables (recall Figure 3): performance based and psychological variables. Evaluation of the observed efficacy requires measuring the efforts needed to use the method and quality of output. We have chosen to measure one performance based variable and three psychological variables as follows.

Amount of correct dependencies is measured as comparison of results from direct dependency linking and associating through domain model for each test subject. This construct was analysed using a log and one question (i.e. practical task, Q6 in [21]) from the questionnaire.

Perceived Ease of Use (PEU) is measured using four questions. *Perceived Usefulness* (PU) is measured using two questions. *Intention to Use* (IU) is measured using three questions.

The order of the questions in the questionnaire was randomized to avoid monotonous responses. To avoid a possible ceiling effect, there was no time limit for the experiment restricting the subjects.

4 Results of Experiment

In average, the subjects used 3,5 hours to perform the task. Approximately 3/5 of the time was used to study the experimental materials. There are three reasons why measurement of time is not used to calculate efficiency. Namely, 1) the subjects were not studying the material equally before and during the actual linking; 2) some subjects found "short cuts" in some interfaces; and 3) some encountered the lack of "undo" command in CO_2SY more than others. An interaction with the tools is otherwise assumed to take similar time if disregarding the routines forced by the interface. Analysis of the log, showed that in average a fragment had 3,8 (in case 1) and 2,5 (in case 2) directly linked fragments. Similarly, a fragment had been associated with 3,3 (in case 1) and 3,1 (in case 2) concepts in average (see Table 7). These parameters are similar, i.e. the time used is comparable.

Observed effectiveness, i.e. amount of correct dependencies. Recall that the subjects were asked to classify the correctness of the output. The results for our method are displayed in Table 1. Where all partially correct are new dependency pairs for the users, i.e. additional links discovered by the CO_2SY prototype. Correctly identified dependency pairs are grouped exclusively as mutual (identified by the test subject as dependent already when establishing direct dependency links) or additional

knowledge (new dependency links, proposed by CO_2SY and identified being correct by the test subject), see columns 4 and 5 in the table.

Table 1. Analysis of dependency discovery performance

	Subject ID	Totally wrong	Partial additional	Totally correct			Total inspected	Total additional	% of correct	% of possbily correct (incl. Partial)	% of wrong
				mutual	additional	%					
	(1)	(2)	(3)	(4)	(5)	(6) =(5)/((4)+(5))	(7) =(2)+(3)+(4)+(5)	(8) =(3)+(5)	(9) =((5)+(4))/(7)	(10) =((8)+(4))/(7)	(11) =(2)/(7)
Case 1	*1*	0	3	5	1	17 %	9	4	67 %	100 %	0 %
	2	4	3	4	0	0 %	11	3	36 %	64 %	36 %
	3	5	8	12	3	20 %	28	11	54 %	82 %	18 %
	4	2	0	13	2	13 %	17	2	88 %	88 %	12 %
	5	6	1	4	3	43 %	14	4	50 %	57 %	43 %
	6	6	1	5	0	0 %	12	1	42 %	50 %	50 %
	Total	**23**	**16**	**43**	**9**	**17 %**	**91**	**25**	**57 %**	**75 %**	**25 %**
Case 2	*1*	2	3	4	0	0 %	9	3	44 %	78 %	22 %
	2	3	2	3	0	0 %	8	2	38 %	63 %	38 %
	3	1	3	7	0	0 %	11	3	64 %	91 %	9 %
	4	5	3	4	1	20 %	13	4	38 %	62 %	38 %
	5	1	2	6	2	25 %	11	4	73 %	91 %	9 %
	6	9	1	5	0	0 %	15	1	33 %	40 %	60 %
	Total	**21**	**14**	**29**	**3**	**9 %**	**67**	**17**	**48 %**	**69 %**	**31 %**

In case 1 CO_2SY helped to discover 9 additional correct links (column 5), and 16 possibly correct (column 3). Overall, 17% (column 6) of correct dependency links computed by CO_2SY was identified as a new knowledge for the test subjects, i.e. the dependencies overlooked when making direct dependency links. While the amount of total additional (both, totally correct and partially correct) equals to 25 (column 8). In case 2, the results are slightly worse, i.e. only 3 dependency pairs were considered being correct additions to the set already identified by the users. That makes 9% being new knowledge to the test subjects. Total additional dependency links identified by the CO_2SY prototype sum up to 17 (column 8). Effectiveness of the method is proved by increased recall compared to manually linked product fragments. Therefore, hypothesis $\mathbf{H_1}$ is confirmed. Difference in results between cases is explained by the size of the cases, i.e. 89 and 23 product fragments (in case 1 and case 2, respectively). Since the bigger amount of product fragments makes it possible to oversee some dependencies. However, section 5 presents more detailed analysis of the difference between the results with case 1 and case 2.

The Perceived Ease of Use (PEU). Hypothesis $\mathbf{H_2}$ can be statistically tested by verifying whether the scores that subjects have given to the questions related to the PEU constructs of MEM are significantly better than the middle score, i.e. the score 3 on the Likert scale for the questions. The score 3 means, that a subject's perception is neutral, i.e. the method was perceived neither easy nor difficult to use. If subject's rating is higher than the middle score, then he/she perceives an advantage of the method. Therefore, the null hypothesis for the hypothesis $\mathbf{H_2}$ is formulated as follows.

$\mathbf{H_{2N}}$: *The perception of the method being ease of use is neutral.*

Table 2 shows an average score for each of the subjects, calculated from the responses to the PEU relevant questions. The One-Sample Kolmogorov-Smirnov test (with Normal theoretical distribution) was applied to the answers related to the PEU

constructs. The distributions were normal, i.e. *p* values were high (lowest was 0,33 for Q13 Ease of Using CO_2SY) indicating that distribution is quite normal.

Consequently, one-tailed t-test was used to check for the difference in mean of PEU construct and the middle score value 3. To evaluate the significance of the observed difference, we applied a statistical test with a significance level of 5%. Table 3 provides descriptive statistics for the PEU construct. The results in Table 4 allow for the rejection of the null hypothesis H_{2N}, meaning that we empirically corroborated that participants perceived the tool and method to be easy to use.

Table 2. Mean scores assigned by users for PEU

Subject ID	1	2	3	4	5	6
PEU	4,00	3,20	3,40	4,20	3,60	4,40

Table 3. Descriptive statistics for PEU

N	Min	Max	Mean	Std. Deviation	Std. Error Mean
6	3,200	4,400	3,800	0,473	0,193

Table 4. One sample t-test for difference in mean for PEU

t	1-tailed p	Mean difference	95% Confidence Interval of the difference	
4,140	0,005	0,800	0,303(lower)	1,297(upper)

To measure reliability, Cronbach's alpha was computed for the construct PEU. Alpha value was 0,86 (usually values over 0,7 are expected in order for construct to be reliable). Results of total item statistics show that all items are consistent, i.e. Cronbach's alpha is still above 0,8 if any of the items deleted [20]. So, we conclude that the items used to measure Perceived Ease of Use are reliable and valid measures for this perception based construct.

The Perceived Usefulness (PU) we have measured using two items from the questionnaire [21]. Answers to one question were identical, i.e. all test subjects answered that the prototype helped to discover *some* new correct dependency links. While answering to another question about the accurateness of the results provided by the prototype, two test subjects have chosen "neutral", i.e. middle value in a scale from *1=Total disaster* to *5=Very accurate*. Others have chosen value 4.

In order to validate the users' consistency in answering, we have compared answers to that question (i.e. Q18) with observed (actual) effectiveness, i.e. the answers presented in Table 1. Data in Table 5 are taken from Table 1, except the last column. Table 5 explicitly shows that the same subjects had least percentage of the additional correct dependency links discovered by CO_2SY that scored "neutral" in Q18. So, we can treat their answers as honest and consistent.

In summary, the PU construct shows that the usefulness of the method and prototype system has positive perceptions among users. In section 5 we return to the analysis why 1/3 of users ranked accuracy of CO_2SY results as "neutral" (average).

Table 5. Consistency in responses about PU

Subject ID	1	2	3	*4*	5	*6*
# of total inspected	18	19	39	30	25	27
# of total additional	7	5	14	6	8	2
% of additional	38,9%	26,3%	35,9%	**20,0%**	32,0%	**7,4%**
Q18	4	4	4	**3**	4	**3**

Remark. Values for the last three lines are calculated from the data in Table 1.

Intention to Use (IU) has been measured by three items in the questionnaire. The subjects were asked in two questions (i.e. Q10 and Q11, see Figure 4a) and [20, 21]) to rank the tools in preferable to use order, i.e. placing to 1st, 2nd or 3rd place. Because of specificity of the response format, we have used non-parametric test. Namely, Kendall coefficient of concordance W [16] has been used to measure agreement among users' ranking. Table 6 displays the ranks given by users and average of the ranks ($\overline{R_i}$). Kendall coefficient of concordance $W_{Q10} = 0{,}58$ for Q10 and $W_{Q11} = 0{,}86$ for Q11.

Table 6. Responses to Q10/11 and mean of ranks

Question	**Q10**							**Q11**						
Subject ID	**1**	**2**	**3**	**4**	**5**	**6**	$\overline{R_i}$	**1**	**2**	**3**	**4**	**5**	**6**	$\overline{R_i}$
CO_2SY	1	1	1	2	1	1	*1,17*	1	1	1	1	1	1	*1,00*
Doors	3	2	2	1	2	3	*2,17*	3	2	2	2	2	2	*2,17*
Trace.matrix	2	3	3	3	3	2	*2,67*	2	3	3	3	3	3	*2,83*

The subjects needed to choose the preferable tool in third question measuring IU (i.e. Q24). There all test subjects selected the prototype, though subject #2 in addition selected traceability matrix, and subject #4 – Telelogic Doors. The latter has provided justification that "Doors is more suitable for moderate-sized project", "CO_2SY would be great for large scale / distributed development". Based on the above presented data analysis, we can claim that there is an intention to use the tool (method), i.e. H_4 is confirmed. Next section takes a closer look to the results discussed in this section.

5 Analysis and Discussion

Here we further analyse the possible reasons behind the results. Furthermore, we discuss the results in comparison with other methods and finally, we elucidate on threats for validity of the conducted experiment.

5.1 Cause Analysis

In order to investigate the difference between the results with case 1 and case 2 respectively, we have analysed average amount of concepts associated with a product fragment (see Table 7). Generally, it can be observed that the bigger concept cluster the bigger result set per fragment is produced, i.e. the result set will contain more

false positive. No significant correlations have been found between an average concept cluster size (Table 7) and the result set (recall Table 1).

However, there is a notable difference in a cluster size between group A and group B. The difference exists in both cases. This difference in an average amount of concepts used to describe semantics of a product fragment is an outcome of different usage sequence of the experimental tools. Group B has started performing task with the CO_2SY prototype, and later proceeding to direct linking using the comparative tools. While group A did it other way around. Therefore, group A needed to investigate the product fragments more thoroughly when performing the direct dependency linking task, i.e. they had more clear perception of the semantics (content) of the product fragments when associating them with the concepts. Consequently, they have used smaller concepts cluster to define semantics of product fragments.

Next, we have analysed the responses on the quality of case 1 and case 2. We assume that high quality of the domain model and the case description will facilitate association of product fragments with concepts, while lower quality of the product fragments makes the direct dependency linking more difficult. The results of quality assessment are summarised in Table 8. Values in Table 8 are displayed as count of answers, e.g. 3 subjects responded that quality of fragments in case 1 was fair. The weighted total quality, WT, is calculated as in eq.2, as follows.

$$WT = \sum_{i=1}^{5} (w_i \times |V_i|) \cdot \tag{2}$$

VC is an exhaustive set of value categories, i.e. VC = {very bad; bad; fair; good; very good}, V_i is a set of all occurrences of a response type from VC, v∈ VC and v ∈ V; W is a set of weights, w∈ W, where W is a set of weights, i.e. W= {-2; -1; 0; 1; 2}.

Table 7. Mean of concepts cluster size associated per fragment

Subject ID	Case 1					Case 2		
	Mean				Overall mean (± st.dev.)	Mean		Overall mean (± st.dev.)
	code	require-ments	user manual	design		code	require-ments	
1	2,3	3,0	2,0	2,3	2,4 (±0,8)	1,2	2,5	1,9 (±1,0)
2	3,0	4,5	1,5	4,0	3,3 (±1,4)	3,2	2,8	3,0 (±2,4)
3	2,1	2,2	1,9	3,0	2,3 (±1,2)	3,4	4,8	4,1 (±2,1)
4	4,5	5,5	4,5	4,0	4,6 (±0,7)	4,2	1,3	2,7 (±2,5)
5	4,5	4,5	4,0	5,0	4,5 (±1,1)	4,4	4,3	4,3 (±1,0)
6	4,0	4,5	1,5	2,0	3,0 (±1,5)	2,0	3,0	2,5 (±1,4)
Overall					**3,3 (±1,4)**			**3,1 (±2,0)**
group A					*2,3 (±1,2)*			*2,7 (±1,0)*
group B					*4,1 (±1,2)*			*3,4 (±2,1)*

The quality of case 2 description and domain model was perceived much higher than of case 1 (in addition, half of subjects identified lack of domain knowledge in case 1), while the variation of the individual fragments' quality is not so big. However, that does not explain the differences of results in Table 1. Small variation in perceived quality of product fragments suggests, that the direct linking should have been easier in case 1, meaning *less additional correct links* identified by CO_2SY.

However, one test subject noted that fragments were more related to the structure of the problem in case 1, whereas in case 2 fragments seemed to be related to the structure of program (software). That sounds reasonable and explains the results in Table 1, as case 1 was based on the MSc project.

Further, in order to analyse the variance of the subjects' perceptions regarding PU construct, we decided to take a look at users' pattern using confidence level when associating with concepts. It is reasonable, since high confidence level of association gives a numerical value of 0,2 (0,5 is for medium and 0,8 is for low confidence level) - these values are denoting semantic relatedness of fragments, i.e. lower value shows that fragments are closer in their semantics [22]. Table 9 shows that, actually subject #4 used only high confidence level (100%) and subject #6 used over 90% of high confidence level for associations. This was the reason to generate a lot of false positive dependency assessments and is considered to be the reason for different answers for the construct of Perceived Usefulness.

Table 8. Quality of cases

Quality of		Very bad	Bad	Fair	Good	Very good	Weighted total	Median
Case 1	description	-	1	4	1	-	**0**	3,0
	domain model	-	-	4	2	-	**2**	3,0
	fragments	-	1	3	1	1	**2**	3,0
Case 2	description	-	-	2	4	-	**4**	4,0
	domain model	-	-	1	5	-	**5**	4,0
	fragments	-	2	2	2	-	**0**	3,0

Table 9. Percentage of confidence levels used for associations and overall performance

Subject ID	1	2	3	4	5	6
High	68%	80%	67%	**100%**	64%	**91%**
Medium	29%	15%	28%	0%	33%	9%
Low	3%	5%	5%	0%	3%	0%
# of total inspected	18	19	39	30	25	27
# of total additional	7	5	14	6	8	2
% of additional	38,9%	26,3%	35,9%	**20,0%**	32,0%	**7,4%**

Remark. The last three lines are from Table 5.

5.2 Related Work

Most approaches for traceability concern process traceability by recording rationale for change, etc (cf. [14]). Our focus here is product traceability, i.e. relating various product fragments throughout whole IS development life-cycle. The special settings (i.e. distributed development) for which our method is developed and span of the whole development life-cycle by our method [20, 22], makes it difficult to compare with the evaluation of other methods. Other approaches mainly deal with a limited set of development life-cycle phase (i.e. limited set of product fragments types), e.g., from requirements to architecture [13]. Some approaches are based on a particular tool (e.g., IBM® Rational Rose™ [9]) or specific notation family (e.g., UML [8]).

Furthermore, there are approaches tackling the problem of dependency discovery adopting Information Retrieval techniques (e.g., [1, 7, 19]). Unfortunately, they support only natural language based fragments.

However, *precision* and *recall* are two commonly used metrics to evaluate the utility of traceability techniques (e.g., [7, 19]). *Recall* is the percentage of all true links retrieved, and *precision* equals the percentage of true links in the answer set. However, in evaluations of other traceability approaches, the set of true links is usually based on relations identified by users (cf. [19]), i.e. the complete set of correct (true positive) dependency links has not been known.

The precision level for case 1 is 75% (see column 10 in Table 1) including the partially correct. While precision of totally correct (column 9) is 57%[2]. The corresponding values for case 2 are 69% and 48%. The experiment does not lend itself to calculate recall, however. Firstly, the complete set of correct (true positive) dependency links was not known. We have decided not to use relations identified by users since our experiment has shown that users can easily overlook some of true dependency links, simply either because of the amount of product fragments they need to investigate, or not being experts in a particular domain or product fragment notation. Secondly, as our method is user-centric, it was more important to observe users' perception of effectiveness. However, the earlier discussed amount of additional dependency links (correct and partial correct, see column 8 in Table 1), indicates that our method is effective, i.e. CO_2SY has discovered more dependency links than processing the fragments "manually". Furthermore, without any enforced threshold (at the cost of precision) on the dependency (semantic distance) computation, our method can actually provide 100% recall, since associated product fragments and concepts in domain model comprise a network of indirectly linked product fragments.

5.3 Threats to Validity

The following possible threats to the validity of this experiment have been identified.

- The case study is executed at the university. However, the experiment examined real cases and 5 out of 6 test subjects had industrial experience.
- Fair answers vs. colleagues answers. However, the analysis of the perceived usefulness construct shows that answers are consistent and likely to be fair.
- Users provided subjective evaluations. The individuals interpret the experimental materials and tasks according to their experience. Experience seemed to be similar for most of individuals.
- Subjective choice of comparative tools. Telelogic Doors is one of the leading tools in the area. Traceability matrix was chosen to use as one of the traditional techniques. Availability of IR techniques-based tools (e.g., [1, 7, 19]) is limited, since most of them are academic prototypes. Even if we would have such a tool, its applicability would be limited because of variety of product fragments types (e.g. binary files).
- Fatigue effect. On average 3,5 hours were spent to complete the tasks and fill the questionnaire. Therefore, this effect is not considered relevant.

[2] Precision ratio of 50% means that developer has to examine about one false positive per true link.

6 Conclusions and Future Work

A method for dependency management using a conceptual domain model has been briefly presented here. The objective of this paper has been to discuss and present an empirical evaluation of the method and its implementation. In the experiment described here, we have focused on testing its possible effectiveness and practical applicability. The results, although preliminary, provide positive evidence about the ability of our method to automate discovery of dependency relations. The subjects' perceptions seem to confirm the performance-based results. The test subjects have perceived our method as easy to use and useful, and they have expressed intention to use the method.

We have corroborated that the test subjects produced consistent answers, though the number of participants was limited in this experiment. Despite the fact that the scope was limited, results of the experiment give indications on the method applicability and feasibility. Yet, the results should be interpreted only as preliminary, due to limited scope and amount of data, as well as artificial, different than intended, use of the method and tool. In intended settings of usage, developers first collaborates in problem modeling and later on associating their own product fragments (he/she has developed) with the concepts from the domain model. Therefore, a larger scale experiment that would imitate the intended use of the method is necessary in order to reconfirm the results obtained here. Furthermore, a certain future need is to investigate model quality issues in more detail, i.e. how detail and intricate the model has to be to encourage more selective association and thereby detect really useful hidden relationships among fragments.

Moreover, the experiment has shown the necessity to improve the user interface of the current version of CO_2SY. Telelogic Doors outperformed CO_2SY in visualisation of dependency links. Therefore, we consider enhancing manipulation of results by providing different views and filters.

References

1. Cerbah, F., and Euzenat, J. Traceability between models and texts through terminology. *Data and Knowledge Engineering* 38(1), Elsevier Science Publishers (2001) 31-43.
2. Chen, P.P., Thalheim, B., and Wong, L.Y. Future directions of conceptual modeling. In Chen, P.P. *et al*. (Eds.), LNCS 1565, Springer-Verlag (1999) 287-301.
3. Dijkstra, E.W. A note on two problems in connexion with graphs. *Numerische Mathematik* 1 (1959) 269–271.
4. Egyed, A., and Grunbacher, P. Supporting software understanding with automated requirements traceability. *JSEKE* 15(5) (2005) 783-810.
5. Erichsen, K.O. *Enabled traceability in distributed system development*. Master thesis, IDI, NTNU, Trondheim, Norway (2003).
6. Halpin, T. *Information modeling and relational databases, from conceptual analysis to logical design*. Morgan Kaufman, San Mateo, California, USA (2001) 792 p.
7. Hayes, J.H., Dekhtyar, A., and Osborne, J. Improving requirements tracing via information retrieval. In *Proc. of Intl. Conf. on Requirements Engineering* (RE'2003) 138-147.

8. von Knethen, A. Change-oriented requirements traceability: Support for evolution of embedded systems. In *Proc. of 18th Intl. Conf. on Software Maintenance* (ICSM 2002), Montreal, Canada, IEEE Computer Society (2002) 482-485.
9. Letelier, P. A framework for requirements traceability in UML based projects. In *Proc. of the 1st Intl. Workshop on Traceability*, Edinburgh, UK (2002) 32–41.
10. Matulevicius, R. *et al*. MEIS system requirements specification. Technical report, IDI, NTNU, Norway (2004).
11. Moody, D.L. *Dealing with complexity: A practical method for representing large entity relationship models*. PhD thesis, University of Melbourne, Australia (2001) 354 p.
12. Olive, A. Conceptual schema-centric development: A grand challenge for information systems research. In *Proc. of CAiSE 2005*. LNCS 3520, Springer-Verlag (2005) 1-15.
13. Pohl, K., Brandenburg, M., and Gulich, A. Integrating requirement and architecture information: A scenario and meta-model based approach. In *Proc. of the 7th Intl. Workshop on Requirements Engineering: Foundation for Software Quality* (REFSQ'01) (2001).
14. Ramesh, B., and Jarke, M. Toward reference models for requirements traceability. *IEEE Transactions on Software Engineering* 27(1) (2001) 58–93
15. Riemenschneider, C.K., Hardgrave, B.C., and Davis, F.D. Explaining software developer acceptance of methodologies: A comparison of five theoretical models. *IEEE Transactions on Software Engineering* 28(12) (2002) 1135-1145.
16. Siegel, S., and Castellan, N.J. *Nonparametric statistics for the behavioural sciences*. McGraw-Hill, Inc. 2nd edition (1988).
17. Solvberg, A. Data and what they refer to. In *Conceptual modeling: Current issues and future trends*. LNCS 1565. Springer-Verlag (1999) 211-226.
18. Solvberg, A., and Kung, D.C. *Information systems engineering – An introduction*, Springer (1993).
19. Spanoudakis, G., Zisman, A., Perez-Minana, E. and Krause, P. Rule-based generation of requirements traceability relations. *Journal of Systems and Software* 72(2) (2004) 105-127.
20. Strasunskas, D. *Domain model-centric distributed development. An approach to semantics-based change impact management*. PhD thesis, NTNU, Norway (2006) 311 p.
21. Strasunskas, D. *Evaluation of domain model-based change impact management. The experimental materials.* Technical Report, IDI, NTNU (2005). Available at http://www.idi.ntnu.no/~dstrasun/evaluation/
22. Strasunskas, D., and Hakkarainen, S. Process of product fragments management in distributed development. In *Proc. of the CoopIS'2003*, LNCS 2888, Springer-Verlag (2003) 218-234.
23. Su, X., and Gulla, J.A. An information retrieval approach to ontology mapping. *Data & Knowledge Engineering* 58(1) (2006) 47-69.
24. VA Software. *Leveraging open source processes and techniques in the enterprise*. White Paper. VA Software, November (2004).

Advanced Recommendation Models for Mobile Tourist Information

Annika Hinze and Saijai Junmanee

University of Waikato, New Zealand
a.hinze@cs.waikato.ac.nz, saijai.j@egat.co.th

Abstract. Personalized recommendations in a mobile tourist information system suffer from a number of limitations. Most pronounced is the amount of initial user information needed to build a user model. In this paper, we adopt and extend the basic concepts of recommendation paradigms by exploiting a user's personal information (e.g., preferences, travel histories) to replace the missing information. The designed algorithms are embedded as recommendation services in our TIP prototype. We report on the results of our analysis regarding effectiveness and performance of the recommendation algorithms. We show how a number of limiting factors were successfully eliminated by our new recommender strategies.

1 Introduction

A number of mobile or location-based tourist information systems have been designed [5,12,20] and used (e.g., [19]). To the best of our knowledge, none of these systems address the problem of how to give recommendations to mobile users that go beyond simple predefined lists of interesting sites near by.

Recommendations have been used successfully in E-commerce. Recently, the number of tourism portals on the Internet that provide information filtered by users' preferences has increased [17]. However, the algorithms used there cannot simply be re-applied in a mobile, location-based and personalized context. We support Zipf's [23] observation that a successful tourist application requires good integration of personalization and context-awareness. Inherently, tourists frequently change their location during their travel, leading to a fluid change of context. Furthermore, tourists' information needs are location-dependent by nature, often indicating the user's current position as a parameter for the recommendation system. The systems additionally need to consider contextual information about the sights [1].

Related work in tourist guides as well as in recommender systems has been extensively examined in our previous study [9]. We have shown that none of the other tourist information systems provide advanced personalized and context-aware recommendations. In the same study, we introduced an initial recommendation service to our mobile personalized tourist guide TIP. Our earlier study confirmed the high potential for employing recommendation services in mobile

R. Meersman, Z. Tari et al. (Eds.): OTM 2006, LNCS 4275, pp. 643–660, 2006.

tourist information systems. Following that study, we identify the following limitations and challenges for the existing recommendation models:

New user problem: Recommendations depend on information about the user, such as the users' explicit interests or their feedback scores given for sights on previous visits. When a new user enters the system, no or little information may be available. The lack of sufficient initial information prevents recommendations.

Sparse feedback scores or cold start problem: Identifying a group of users who have similar interests as a given user depends on the feedback given by these users. If the number of users is small relative to the number of sights, especially at the beginning of the system usage, the feedback becomes sparse. Similar users may not be identified. Cold-starts are a significant problem as the initially given recommendations may be of poor quality, which negatively influences the user's confidence in the system.

Users with specific preferences: Also known as the *Gray Sheep* problem [22], it refers to a user whose preferences are unusual compared to the other users in the system. Typically, feedback given by this user does not consistently agree with any groups of users in the system. These individual users will rarely receive accurate recommendations that match their interests.

Over-specialization: The user may be restricted to see only items that are similar to those they have already seen (and given feedback about) since the system may only recommend items scoring highly against the user's specified preferences. Discovering of preferences that are not explicitly known to the user is often not possible.

Transparency and user control: Understanding the relationship between their input to the system (feedback) and system's output (recommendations) allows the user to interact effectively with the system. Transparency allows users to meaningfully revise their input in order to improve recommendations, rather than being blinded by the given result [21]. Lacking a feeling of control, users may be unwilling to follow the system's recommendations.

User satisfaction: The user's satisfaction with the recommendations reflects the quality of the implemented recommendation models and algorithms and the quality of the available data. However, user satisfaction is difficult to evaluate [6]. One simple way is to record the user interaction with the system. If the user follows a given recommendation, we may assume they are satisfied with the recommendation.

Scalability: Providing recommendations requires expensive computation that may grow non-linearly with the number of users and items in the database. Therefore, to successfully employ recommendations on a mobile device, sophisticated data structures and advanced, scalable architectures are required to provide recommendations with acceptable delay time responding to the users' requests [16].

Consequently, the project reported in this paper addresses the identified limitations. Our goal was to introduce collaboration between TIP's information service and the recommendation service. We propose recommendation methods that can

retain the existing advantages and balance the current drawbacks of providing recommendation in a mobile tourist information system. This paper introduces extended recommendation models and reports about their implementation. We discuss our findings from the analysis of the extended models in respect to the identified limitations. We believe the new models will shield the users from information overload as well as provide the users with satisfactory personalized recommendations in a mobile tourist information environment.

The remainder of the paper is organized as follows: Section 2 recaps foundations of recommender models and the TIP system. Section 3 introduces the concepts our advanced recommendation models; details of which are given in Section 4. In Section 5, we report about the results of the evaluation of the implemented models. Section 6 summarizes the results of the project.

2 Background

This section gives background information about recommendation methods and the TIP system.

2.1 Recommendation Paradigms

This section introduces the three basic recommendation principles. In the general formwork of recommendation models, tourist sights are considered as *items*.

Content-based Recommendation uses information about items and a user's preferences. Preferences are stored in *user profiles* and may be defined either explicitly by the user or implicitly by extracting user information from past activities, e.g., their purchase history. Items that are similar to the items the user bought in the past, or items that adhere to the user's profile are recommended. This approach is independent of other users' feedback; after profile definition, recommendations can be given immediately. Also users with a unique interest can be catered for. However, users may be restricted to recommended items similar to what they liked before [13]. This approach has been used in E-commerce systems for music [4] and books [15].

Collaborative Filtering provides recommendations based on the feedback of like-minded users about items in their past activities. Feedback can be given explicitly as a rating score, or can be implicitly derived, e.g., from purchase records [18]. The more feedback ratings are available the better the prediction. Based on the similarities between ratings provided by the given user and other users, a group of *neighbor* users is defined that have similar preferences. Items that neighbors liked are then recommended to the given user. The system must be initialized with a large amount of data to generate effective recommendations [2]. This approch has been used in amazon.com.

Knowledge-based Recommendation uses knowledge about users and items to reason which products meet the users requirements. These systems offer a dialog interaction that walks the user through a discrimination tree of item features. A sequence of questions is designed to eliminate some items from

consideration. This system neither needs explicit user feedback scores nor user preference definitions. However, a complex knowledge engineering algorithm is required. Only static suggestions can be gained [2,5]. This approach has been used for FindMe [3].

Each of the three recommendation paradigms has advantages and drawbacks that need to be balanced to provide effective recommendations in TIP.

2.2 TIP Background

The Tourist Information Provider (TIP) is a mobile tourist information system. The system is a combination of an event notification service and a location-based service (for details see [10]). The main focus of the system is the delivery of information about sights based on user location, interest, travel route and sight-related information to hand-held devices such as PDA. We refer to this feature as TIP's *information service*. In TIP 2.0, the initial *recommendation service* had been extended by the authors to provide three basic recommendation methods [9]: (1) sights near by the current user location, (2) similar to sights that the user liked in the past, and (3) sights enjoyed by similar users.

The set of data that is used for the information delivery in TIP may be re-used for the recommendation service. We classified the data into six parameters:

(R1) User profile: A user's profile specifies information of interest about the user. The system learns the user's preferences from the given information and provides information based on the acquired knowledge about the user. A user may have different profiles depending on context. Here, we assume that at each point i tie only one profile is active.

(R2) Context of a user: A user's context may specify current location, time, means of travel, or other background information. It is used to personalize the information and its delivery.

(R3) Context of a sight: The sight context contains information about groups or types of sights for recommendation, which have certain features in common e.g., churches. The context of sights also covers their location, operating hours and weather conditions. Sight groups may be used to find similar sights within a group.

(R4) User travel history: The user's travel history includes sights, times and locations the user previously visited. The system should not recommend sights the user has already visited (or indicate them as such).

(R5) User Feedback: User feedback are ratings or scores given to the sights on past visits. We distinguish two forms:

- **a)** *Feedback of this user:* The user receives recommendations based on the similarity of sights to other sights this user gave positive feedback about.
- **b)** *Feedback of similar users:* Sights which other similar users liked may be recommended to the user.

As a result, the user gains wider information based not only on their preferences but also their similar preferences with other users.

We propose to combine the six parameters with the three recommendation paradigms. We believe that our approaches will achieve better balance between advantages and drawbacks of the three paradigms.

3 Concepts of Advanced Recommendation Models

In this section, we describe general concepts of the proposed recommendation methods. We distinguish five major approaches as shown in Table 1. They illustrate our concept of combining parameters and existing recommendation paradigms. We now describe the motivation and concept of each approach.

Table 1. Proposed Recommendation Approaches

A.Pure Approaches	
A1.Content-based	based on a particular user's profile (R1) and their feedback (R5a). Recommends sights that are similar to what the user liked in the past.
A2.Collaborative Filtering	based on previous feedback of this and other users (R5a and R5b). Recommends sights that are highly rated by these similar users.
A3.Knowledge-based	based on sight context; recommends sights that are semantically-related to sights this user has visited in the past (R2 and R4).
A4.Must-see Sights	preset sights that are the points of interest in a particular area, e.g., Sky Tower in Auckland. These points of interests can be defined based on the feedback of a large set of users (R5a and R5b).
A5.Nearby Sights	based on user context, sight context and user history (R2, R3 and R4). User context may be the current location and means of their travel. Recommends sights that can be reached conveniently and the user has not seen before.
A6.User Profile	sights that match this user's profile (R1).
B.Compound Approaches	
B1.Nearby Sights and User Profile	extends A5 by further filtering its results according to the user's profile
B2.Revise Profile	based on user profile (R1) revised according to their feedback given to the system (R5a).
B3.Extend Profile	based on user profile that is extended by information about other users (R5a and R5b). Information in profiles of similar users is added to this user's profile (R1).
C.Extended Content-based Approaches	
C1.Implicit Feedback	based on content-based methods but without explicit feedback. Feedback is created from the information in the user profile (R1) and the information in the user history (R4).

Table 1. (*continued*)

C2.Content-boosted Recommendation	combination of the content-based method and the collaborative filtering. The data set for collaborative filtering is extended by simulating missing user feedback based on the feedback of similar users (as proposed in [13]).
C3.Context-aware Feedback	based on content-based methods where the user gives feedback according to their context, e.g., the user prefer going to restaurant X when it is raining or the user likes going to cafe Y on a sunny day because it is near the beach.
C4.Implicit Context-aware Feedback	based on this user's feedback (R5a) that are recorded according to sight context (R3) and user history (R4). User feedback is created from the information in the user history (R4) and the sight context (R3).
C5.User Information and Feedback	considers user profile (R1), user context (R2), sight context (R3), user history (R4) and their feedback (R5a). User context may or may not be considered.
D.Extended Collaborative Filtering Approaches	
D1.User Profile	Assumes that the users like items that match their user profile (R1). Therefore if no feedback (both R5a and R5b) is available from the number of users, the feedback is simulated by creating positive synthetic feedback data based on the user's profile. This synthetic feedback is then used as input for collaborative filtering.
D2.User History	Similar to E1. Synthetic feedback is created based on the information in users' histories (R2).
D3.User Profile and User History	A combination of E1 and E2. Synthetic feedback is created for a group of similar users based on information from their user profiles (R1) as well as their user histories (R4).
E.Extended Knowledge-based Approach	
E1.Supplementary Sight Context	Updates sight context according to the feedback of the user (R5a and R5b). Recommendations are given based on the information stored about the sights, e.g., the semantic groups they belong to. Feedback from user given about the sights may create new groups.

A - Pure Approaches. In addition to the three pure recommendation models introduced before (A1–A3), we use models with one main parameter each (A4–A6): the points of interest in a particular area, everything near by, or sights that match the preferences defined in their profile.

B - Compound Approaches. The models combine user information such as the user's current location and profile to create filtering criteria. Nearby sights which match a user's preferences are recommended to the user (B1). Users are often reluctant to define or adjust profiles explicitly. We therefore

take feedback as well as information from other users to revise and/or extend the user's profile (B2/B3). These methods address the new user problem, the cold start problem, the gray sheep problem as well as over specialization.

C - Pure and Extended Content-based Approaches. The pure content-based recommendation approach relies heavily on similarity between the items' content and the users' preferences. The user is restricted to recommendations similar those already rated (over specialization). When a new user starts to use the system, due to missing feedback no recommendations can initially be given. The extended content-based models therefore aim at enlarging available information about the users' preferences. More user feedback is generated implicitly from information in user profile and user travel history. We also combine content-based and collaborative filtering. Similar users are determined based on their profile and their travel history. As a result, the user need not explicitly provide much feedback on the sights they have visited, but it is still not restricted to receiving recommendations only on sights similar to those highly rated in the past. The extended approaches avoid the limitation discussed before as well as reduce user effort in giving feedback scores.

D - Pure and Extended Collaborative Filtering Approaches. The approach focusses on the similarity of the users. For each user a set of nearest neighbor by calculating correlations between a user and other users. Predicted feedback is calculated for all sights based on the neighbors' feedback. High scoring sights are recommended to the user. The pure method suffers from the cold start problem as well as the gray sheep problem. The system might be unable to detect a neighborhood due to sparse feedback, or lack of similarity, respectively. To overcome these problems, user feedback is taken from other sources, such as user profiles and their travel history. We assume that a user would give high feedback scores to sights that match their user profile. We also believe that sights that the user has visited several times are likely to be their favorites. Therefore, we give high feedback scores to sights the user has visited at least twice in their travel history. The three extended collaborative filtering models remedy sparse user feedback scores from information in the user profiles, their travel history and a combination of their profile and travel history.

E - Pure and Extended Knowledge-based Approaches. The pure knowledge-based recommendation model typically use the user's behavior as a representation, commonly using machine-learning techniques to discover useful patterns [14]. TIP 1.0 employed a simple knowledge-based recommendation based on the user profile. The extended model uses supplementary sight context with the existing semantic groups. New semantic groups may be created based on the feedback score given by the user.

All models introduced here have been implemented in TIP. The next section will give selected details about the implemented design.

4 Design of Recommendation Algorithms

In this section, we explain the design of the algorithms generated for the recommendation models introduced in the previous section. We first define important terms and then introduce details of the extended filter algorithms.

4.1 Terms and Definitions

Users: A user u_x is a person who registers with TIP. Let m be the number of registered users; the set of all users is referred to as U. The *active user* $u_a \in U$ is a particular user who is currently asking for recommendations.

Sights: A sight s_x is an item a user might be interested in. The number of sights stored in the TIP system is denoted n; the set of all sights is S.

A sight group g_x is a set of sights which have some features in common. A given sight can belong to more than one sight group, or to a hierarchy of sight groups. The set of all sight groups is G, with k being the number of all sight groups. Sight groups may be predefined (as in our case) or determined using machine learning approaches.

Sight group profiles: A sight group profile p is a set of sight groups a user u_x has chosen as their sight groups of interests. A sight group profile p is defined as $p : U \rightarrow \{0,1\}^k$ and a sight group profile of a particular user u_x is $p_{u_x} = \{\alpha(u_x, g_1), \alpha(u_x, g_2), \ldots, \alpha(u_x, g_k)\}$ where $\alpha(u_x, g_i)$ is the (Boolean) interest of user u_x in the particular sight group g_i.

Travel history: A travel history $H(u)$ is defined as a temporally ordered list of sights visited by a user u in the past: $H(u) = \{(s_j, t_j) | s_j \in S \wedge u \in U \wedge t_j \in T \wedge u\, visited\, s_j\}$, where T is the set of all time stamps.

Feedback: A feedback f is a numeric value which represents a user's opinion about a sight. It is a rating score $f : U \times S \rightarrow [1, 10] \subseteq \mathbb{N}$ with

$$f(u,s) = \begin{cases} 1 \ldots 4 & : \text{user } u \text{ has negative impression of sight } s \\ 5 & : \text{user } u \text{ has indifferent impression of sight } s \\ 6 \ldots 10 & : \text{user } u \text{ has a good to excellent impression of sight } s \\ no - value & : \text{user } u \text{ has not yet given an impression of sight } s \end{cases}$$

If a user changes their opinion, the new value is kept and the existing one discarded. We distinguish two types of feedback: the given feedback and the predicted feedback. The *given feedback* can be either provided explicitly (f_g) or inferred from available information of this user (f'_g to f''''_g). The number of ($'$) in the implicit given feedback indicates the type of user information used, e.g., f''_g uses information from the user sight groups profile. The *predicted feedback* f_p is a numeric value calculated by the system.

Similarity and Neighborhood. A similarity factor ζ identifies similarity of preferences between the active user u_a and another user. A similarity factor ζ is a numeric value which is calculated by the applied calculation algorithm $\zeta : U \times U \rightarrow [-1, 1] \subseteq \mathbb{R}$. The similarity factor between the active user u_a and user u, $\zeta(u_a, u)$, can be calculated based on historical information such as their feedback scores given to sights in their past visits, their travel histories or their interests in sight groups that have been defined in their sight group profiles. A neighborhood N of an active user u_a are other users who have historically shown preferences similar to the active user. Typically, a specific numeric value is set as criterion c for a valid similarity factor (we use $c = 0.7$). Therefore, the neighborhood N of the active user u_a is $N(u_a) = \{u | u \in U \wedge u \neq u_a \wedge \zeta(u_a, u) > c\}$ where $\zeta(u_a, u)$ describes the similarity factor between the active user u_a and user u.

Recommendation. A recommendation R is a set of sights that the system predicts an active user u_a would like to visit. Recommendations given to the active user may vary depending on the recommendation model selected by the active user.

4.2 Design of Algorithms Using Collaborative Filtering

The idea behind these algorithms (A2, D1, D2 and D3) is that it may be useful to consult the behavior of other users who share the same or related interests. To compute the recommendations for an active user u_a, we use neighborhood-based methods [7] that can be separated into three steps: (1) feedback scores collection, (2) neighborhood formation, and (3) recommendation generation. For our advanced models, we change the first step, the feedback score collection.

New Feedback Score Collection. The problem space can be described as a matrix of users versus sights. In this case, we formulate a $m \times n$ *user-sight* matrix F where each entry $F(i, j) = f_g(u_i, s_j)$ is the feedback of the user u_i for the sights s_j. This matrix is usually sparse because of numerous *no-feedback* values. To remedy the sparsity of the initial user-sight matrix, we insert implicit feedback scores to replace the *no-feedback* values. We propose these implicit feedback scores to be generated using the following techniques:

- Default Feedback: We change the user-sight matrix F in the pure collaborative filtering approach (A2) by placing neutral default feedback scores instead of *no-feedback* values:

$$f'_g(u, s) = \begin{cases} f_g(u, s) & : \text{if user } u \text{ rated sight } s \\ 5 & : otherwise \end{cases}$$

- Using information in the user profile (D1):

$$f''_g(u, s) = \begin{cases} f_g(u, s) & : \text{if user } u \text{ rated sight } s \\ 10 & : \text{if sight } s \text{ belongs to sight groups user } u \\ & \quad \text{has defined in their sight group profile} \\ 5 & : otherwise \end{cases}$$

– Using information from the user travel history (D2):

$$f_g'''(u,s) = \begin{cases} f_g(u,s) : \text{if user } u \text{ rated sight } s \\ 10 \quad\quad : \text{if user } u \text{ has visited sight } s \text{ at least twice} \\ 5 \quad\quad : otherwise \end{cases}$$

– Using a combination from user travel history and user profile (D3):

$$f_g''''(u,s) = \begin{cases} f_g(u,s) : \text{if user } u \text{ rated sight } s \\ 10 \quad\quad : \text{if sight } s \text{ belongs to sight groups user } u \\ \quad\quad\quad\quad\;\; \text{has defined in their sight group profile} \\ \quad\quad\quad\quad\;\; or \text{ if user } u \text{ has visited sight } s \text{ at least twice} \\ 5 \quad\quad : otherwise \end{cases}$$

In our four algorithms using collaborative filtering, we address the *sparsity of the feedback scores* matrix by using implicit feedback to fill in the missing values. The feedback scores are used to find similarities between the active user and other users. The next two steps of neighborhood formation, and recommendation generation remain the same as in the original collaborative filtering approach: A weighted combination of these neighbors given feedback scores is used to calculate the *predicted feedback* for sights for the active user. Sights with high predicted feedback scores (≥ 7) are then recommended.

4.3 Design of Algorithms Using Content-Based Paradigm

In content-based recommendations (A1, C1, C2, C3, C4, and C5), the user first expresses preferences for a set of items (i.e., sight groups). The system then retrieves from a catalogue the items that share common features with the items the user is interested in. For pure content-based recommendations (A1), we select sight groups with sights that the active user visited. We calculate the average value $\bar{\gamma}(u_a, g)$ of the feedback scores given to sights in each of these groups. Sights in sight groups that have high $\bar{\gamma}(u_a, g)$ are given as recommendations. The algorithm tends to over-specialization and suffers from the new user problem. We extend the method by applying knowledge of the *neighborhood-based methods* from the collaborative filtering paradigm: We focus on similarities between available user information. The basic method has the following three steps: (1) user information collection, (2) neighborhood formation, and (3) recommendation generation; we change the first and the second step, the user information collection and the neighborhood formation.

New User Information Collection. Note that we do not collect the feedback scores as in the collaborative filtering algorithms. Rather, we form a *user-user information* matrix V which can be either a $m \times k$ *user-sight group* matrix or a $m \times n$ *user-sight* matrix. We assign a score to each of the entries using one of the following techniques:

– User-sight group profile matrix: Each entry $V(i,j)$ represents the interest $v'(u_i, g_j)$ of user u_i in sight group g_j with

$$v'(u,g) = \begin{cases} 10 : \text{if the user } u \text{ defined their interest in} \\ \qquad \text{the sight group } g \\ 5 \quad : \textit{otherwise} \end{cases}$$

– User-user travel history matrix: Each entry $V(i,j)$ represents the interest $v''(u_i, s_j)$ of the user u_i in the sight s_j based on a number of their visits:

$$v''(u,s) = \begin{cases} 10 : \text{if the user } i \text{ has visited the sight } j \\ \qquad \text{at least twice} \\ 5 \quad : \textit{otherwise} \end{cases}$$

– User-sight group profile and travel history matrix: Each entry $V(i,j)$ in matrix V represents the interest $v'''(u_i, s_j)$ of the user u_i in the sight s_j based on two conditions: The sight is in the sight groups the user is interested in or the user has visited the sights at least twice in the past.

$$v'''(u,s) = \begin{cases} 10 : \text{if sight } s \text{ belongs to sight groups user } u \\ \qquad \text{has defined in their sight group profile} \\ \qquad \textit{or} \text{ if the user has visited sight } j \text{ at least twice} \\ 5 \quad : \textit{otherwise} \end{cases}$$

New Neighborhood Formation and Recommendation Generation. We apply principles from collaborative filtering but use different measures depending on the user information gathered in the user-user information collection step: 'Similarity on sight group of interest' and 'Similarity on sights of interest'. We also divide the recommendations generation into two categories: recommendations generated from other users who have the similar sight groups of interest and recommendations generated from other users with similar sights of interest.

The extended content-based recommendation algorithms address the *over-specialization* and the *new user* problems found in the pure approach. We employ knowledge of collaborative filtering to form a group of users who have similar interests in sight groups and/or travel history. We then recommend to the active user a set of sights liked by these users.

4.4 Design of Algorithms Using Knowledge-Based Paradigm

This method extracts information about the user's behavior from their past activities. This knowledge is used to initialize the system with a set of implicit preferences. The system then creates a dialogue to obtain the user's feedback to the system's recommendations. This feedback is then used to improve the model of the user's preferences (used in A3 and E1). For method A3, we assume that the user who has seen several sights in the group is interested in seeing more of the same. The user history is used for this process. The extended knowledge-based recommendation (E1) applies the principles of the content-based algorithm.

4.5 Design Summary

We described details of the recommendation algorithms for the models A2, D1, D2, and D3 (using collaborative filtering), A1, C1, and C5 (using content-based techniques), and A3 and E1 (knowledge-based recommendation). For each of the above recommendation models, we transformed the outline given in the previous section into the calculation formulas. An overview of all formulas and the details of their implementation in TIP can be found in [11].

5 Evaluation and Analysis

We evaluated three aspects: *effectiveness* and *performance* of the implemented recommendation algorithms as well as the *presentation and interactions* of the recommendation models. The effectiveness evaluation determined the influence of our new methods on the problems of new users, cold starts, gray sheep and over-specialization. Evaluation of interface and interaction addressed the issues of transparency and user control and the user satisfaction. The performance evaluation determined the influences on scalability. In particular, we concentrate on rising number of users and sights.

5.1 Qualitative Evaluation

The effectiveness of the system was evaluated in two extensive exemplary scenario studies. The details of this study cannot be discussed here; interested readers may turn to [11] for details. Figures 1(a) to 1(c) show screen shots of scenario studies. The results of the qualitative study are as follows:

1. The *new user* and *cold-start problem* due to a shortage of the user's feedback scores for the collaborative filtering algorithms are solved. In addition, users that used the system but did not give feedback can also be supported.
2. The *Gray sheep problem* was diminished since the users did not have to give feedback. However, we see this only as a short term remedy. Neither do we follow suggestions to weaken the representation of the unusual opinion of the *gray sheep* in favor of the more usual opinions of members of the neighborhood. Further research is necessary.
3. The main problem of *over-specialization* was remedied with our extended content-based recommendation; users now receive wider recommendations than before. We believe that this type of recommendation helps users to discover new interests and sights.

We also performed a first-cut study of *transparency and user control* and *user satisfaction* issues using the measures of *acceptability* and *tractability*. The results are promising (for details see [11]) and lead the way to further long terms studies.

(a) content-bd. (C1) (b) collabor. (D1) (c) knowld-bd. (E1)

Fig. 1. Screen shots of scenario studies

5.2 Quantitative Evaluation

We examined the performance of the algorithms as *response time*. We also verified the *coverage* as a measure for the number of items for which an algorithm can provide predictions. Here we set our focus on the algorithms using extended *Collaborative Filtering*. The test setting consists of off-line synthesized users and sights information (cf. [8] for a discussion of the use of synthetic data sets to identify the promising recommendation algorithms for future study).

The recommendation service in the TIP system and its evaluation are based on two assumptions: The system will be utilized in a particular area, e.g., downtown Hamilton (location-based pre-caching can be used to restrict the accessed sight data); and there are many more users than items in the tourist information system domain. Accordingly, a typical test data set used in our experiment consists of 1,000 users and 100 sights. We divide the test data sets into a training and a test portion with $p = 0.8$ (80%/20% distribution). Each of the sights is rated by at least one of the users. We use 10 sight groups with randomly assigned sights. Each user specifies 5 random sight groups in their profiles. The number of sights stored in the user travel history is is 20% higher than the number of user feedback scores. The sparsity level (cf [18]) of the data matrix is set to 90%. All experiments are run on a Windows based PC with AMD Athlon XP 2700+ processor with 2.16 GHz and 512 MB RAM.

Complexity Discussion. Vozalis et al. [22] claimed that some existing recommendation algorithms offer theoretically promising mathematical techniques in order to generate their results but nonetheless require complex calculations.

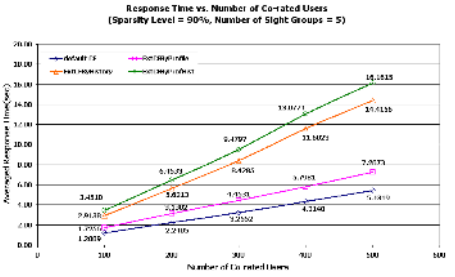


Fig. 2. Response time vs co-rated users

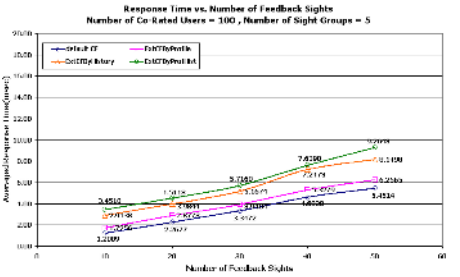


Fig. 3. Response time vs feedback sights

We consider both complexity and response time. Our complexity analysis concentrates on the *user-user* similarity calculation step as the performance bottleneck. We present a worst case estimate and give an approximation for real-world conditions.

The traditional computation complexity of the *user-user* similarities is $O(m^2n)$, where m is the number of users and n the number of sights. The similarity between each pair of users needs to be computed for the subset of their co-rated sights. For our extended collaborative filtering algorithms we further need to take into account the number of sight groups k and the number of sights h in a user's history. The complexity when using the *user sight group profile* is $O(m^2nk)$ as we need to verify whether a sight with *no-feedback* score belongs to any of the sight groups stored in the user profile. When using the *user travel history*, the complexity is $O(m^2nh)$ since we need to compare a no-feedback sight with the sights stored in the user travel history.

In general, the number of sight groups k is rather static and much smaller than the number of sights in the database, ($k \ll n$). Although the number of sights visited by the user h will increase, it still holds $h \ll n$. Consequently, k and h have less influence on the overall complexity $O(m^2n)$. Our experiments are designed to test this theoretical analysis.

Experiments Regarding Response Time. We focus on the influence of the number of users and sights. We designed the typical usage setup described above. We use the following experimental variables:

- m' is the number of users with whom the active user has at least one co-rated sight, with $m' \ll m$;
- n' is the number of feedback given to sights by the active users, with $n' \ll n$;
- k' is the number of sight groups defined in the user profile of the active user and other users, with $k' \leq k$;
- h' is the number of sight visited by the active user, with $h' \geq n'$ since the user may not give their feedback to every sight they have visited.

We refer to these four variables as *co-rated users*, *feedback sights*, *sight groups* and *visited sights*, respectively.

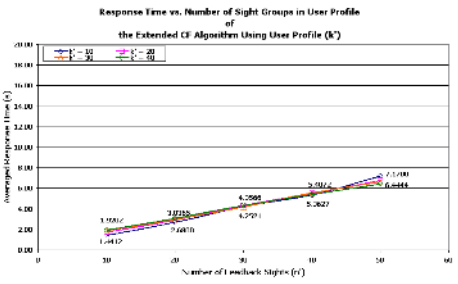


Fig. 4. Extend CF with user profile

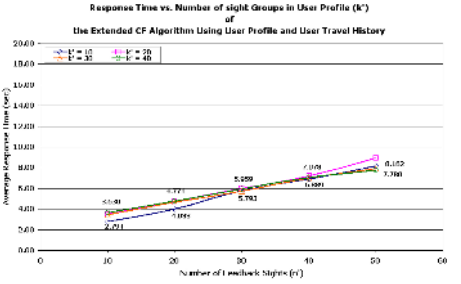


Fig. 5. With user profile and travel history

We performed three experiment series; we now discuss the results of each in turn. Firstly, we evaluated the influence of an increasing number of co-rated users (m'). The results can be seen in Fig 2. The average response time of these four algorithms (A2, D1, D2, and D3) is linear in the increasing number of co-rated users. We observe a separation when using the travel history: These algorithms require more accesses to the database to retrieve the information about the visited sights and the number of visits of each user in the *user-sight* matrix F. Secondly, we evaluated the influence of increased number of feedback sights (n'). Fig. 3 shows that the response time grows linearly with the number of co-rated sights. Thirdly, we evaluated the influence of the number of sight groups (k') defined in the active user's profile and the feedback sights (n'). Here, we only focus on algorithms that use profile information. The results are shown in Figs 4 and 5: increasing number of sight groups k' has less impact than the increasing number of feedback sights n'.

We conclude that the results of experiments one and two confirm our theoretical analysis. Using default values or profile information to replace no-feedback values is less costly than evaluation of user histories. Results of experiment three show that an increase in the number of feedback sights strongly influences the performance. Further extensive experiments on each of these four variables are required. The results of this pilot study (using synthetic test data) will guide further extensive analysis with real world data.

Experiments Regarding Coverage. We examine the algorithms ability to provide predictions over the number of un-rated sights of the active user. As the design of the extended collaborative filtering primarily aims to solve the problem of the missing feedback scores, we use a sparse user-sight matrix ($\approx$ 93%). The experimental data contains 3000 users and 100 sights. The number of sight groups per user profile is randomly allocated (between 1 and 10). All other settings are as in the previous tests. We randomly select 4 test data sets with user groups of 1000 to 2500. We observe the coverage (Fig. 6) and the percentage of the users who get recommendations (Fig. 7).

The coverage is lowest for the algorithm exploiting user profiles ($\approx$ 0.58 on average) and highest for using default values ($\approx$ 0.88 on average). The other

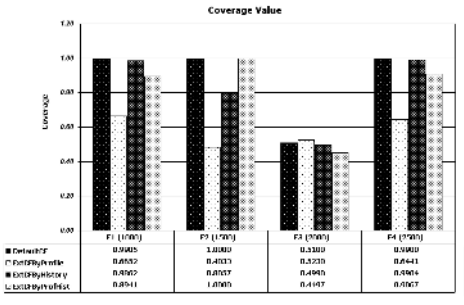


Fig. 6. Coverage values

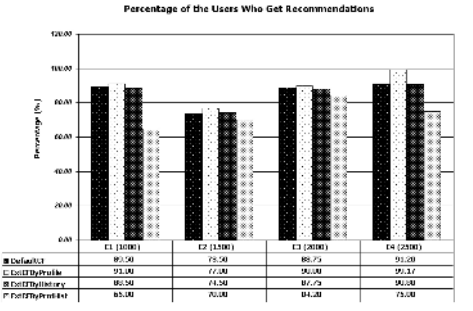


Fig. 7. Users with recommendations

two algorithms have similar coverage values. We also see that the user number is not as influential as the quality of their feedback. For example, the third test data (2,000 users) has always the lowest coverage due to its low fraction of rated sights. Consequently, some users have many neighbours and sights but only a few predictions. The quality of the predictions is expected to be high. A large number of users (65% − 99%) will receive recommendations (see Figure 7)The algorithm that uses user profiles shows the highest number in every data set (91% − 99%); numbers are lowest for the combination of user profile and travel history (65% − 75%).

Summary of evaluation. We presented a multi-faceted evaluation of our improved recommendation service. In summarizing our results, we draw four key findings from our evaluation:

1. Our theoretical analysis concluded that the algorithms have a complexity of $O(m^2n)$ for m users and n sights. The results of the experiment regarding the system's response time confirm our theoretical analysis.
2. The results of the experiments show that using the default value to replace the missing feedback scores has the best performance. It provides highest coverage and serves a large number of users.
3. The extended algorithms with user profiles serves most users but fails to provide good coverage. The complexity i similar to using default values.
4. Algorithms using the travel history suffer from high computational complexity due to extensive data access. Incremental indexing methods and view update strategies may provide solutions.

6 Conclusion

The goal of our project was to develop a recommendation component for TIP without the limitations of typical algorithms. Earlier in this paper, we identified seven critical limitations to existing designs. To eliminate these shortcomings, we proposed a collaboration between TIP's information delivery service and

the recommendation service to re-use user-related information. Based on six parameters, we introduced 18 models in 5 groups. We implemented 15 algorithms to facilitate a systematic initial comparison. The most promising four algorithms were selected for a closer, multi-faceted evaluation.

Our evaluation analyzed the new recommendation component in regard to the identified limitations. We measured the recommendation quality as well as the system performance. Quality was evaluated in scenario-based tests. We found that the problems of new users, cold-start and over-specialization were remedied. The gray sheep problem was lessened but we see the need for future research. The quantitative evaluation focussed on system performance and scalability. We discussed the computational complexity ($O(m^2n)$) and evaluated response time and coverage of the four recommendation algorithms using collaborative filtering paradigm. The simple default-value algorithm is the most effective one and reaches a high number of users. Using information from profiles and history also shows good performance results. The next step is now an extensive user study to determine the quality of the recommendations and the user satisfaction. We are currently focussing on improving performance and user acceptance by including trust measures, which will significantly reduce the number of user users that need to be evaluated.

To summarize, our study has addressed all limitations we initially identified. The lack of user information used for building a user model has been remedied. The current system has been analyzed in multi-faceted studies based on artificial data. The promising results will direct further studies involving users and real-world data. Future research will include extensive real-world evaluation as well as trust-based communication to further improve transparency, user control, and scalability. Further research is needed to explore the handling of multiple context-dependent user profiles.

References

1. I. Amendola, F. Cena, L. Console, A. Crevola, C. Gena, A. Goy, S. Modeo, M. Perrero, I. Torre, and A. Toso. UbiquiTO: A multi-device adaptive guide. In *Mobile HCI: Mobile Human-Computer Interaction*, Glasgow, UK, September 2004.
2. R. Burke. Knowledge-based and collaborative-filtering recommender systems. In *Proceedings of the Workshop on AI and Electronic Commerce. AAAI 99*, Orlando, Florida, 1999.
3. R. Burke, K. Hammond, and B. Yong. The findme approach to assisted browsing. *IEEE Expert: Intelligent Systems and Their Applications*, 12(4):32–40, 1997.
4. H. Chen and A. Chen. A music recommendation system based on music data grouping and user interest. In *Proceedings of the tenth international conference on Information and knowledge management*, Atlanta, Georgia, USA, October 2001.
5. K. Cheverst, K. Mitchell, and N. Davies. The role of adaptive hypermedia in a context-aware tourist guide. *Communication of the ACM*, 45(5):47–51, 1997.
6. C. Hayes, P. Massa, P. Avesani, and P. Cunningham. An on-line evaluation framework for recommender systems. In *Workshop on Personalization and Recommendation in E-Commerce*, Malaga, Spain, May 2002.

7. J. Herlocker, J. Konstan, A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative filtering. In *Proceedings of the SIGIR'99*, Berkley, California, USA., August 1999.
8. J. Herlocker, J. Konstan, L. Terveen, and J. Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems*, 22(1):5–53, January 2004.
9. A. Hinze and S. Junmanee. Travel recommendations in a mobile tourist information system. In *Proceedings of Information Systems and its Application ISTA'2005*, Palmerston North, New Zealand, May 2005.
10. A. Hinze and A. Voisard. Location and time-based information delivery in tourism. In *Proceedings of Advances in Spatial and Temporal Databases, 8th International Symposium*, Santorini Island, Greece, July 2003.
11. S. Junmanee and A. Hinze. Design and implementation of an advanced recommendation component in the tourist information system tip. Technical Report X/2006, University of Waikato, Computer Science Department, Hamilton, New Zealand, June 2006. based on Master's Thesis.
12. P. Klante, J. Krosche, and S. Boll. Accessights - a multimodal location-aware mobile tourist information system. In *Proc. of the 9th Int. Conf. on Computers Helping People with Special Needs ICCHP 2004*, Paris, France, July 2004.
13. P. Melvile, R. Mooney, and R. Nagarajan. Content-boosted filtering for improved recommendations. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence (AAAI-2002)*, Edmonton, Canada, July 2002.
14. S. E. Middleton, N. Shadbolt, and D. D. Roure. Ontological user profiling in recommender systems. *ACM Transactions on Information Systems*, 22(1):54–88, January 2004.
15. R. Mooney and L. Roy. Content-based book recommending using learning for text categorization. In *Proceedings of the fifth ACM conference on Digital libraries*, San Antonio, Texas, USA, June 2000.
16. M. Papagelis and D. Plexousakis. Qualitative analysis of user-based and item-based prediction algorithms for recommendation agents. In *Proc. of the Workshop for Cooperative Information Agents VIII*, Erfurt, Germany, September 2004.
17. C. Paris. Information delivery for tourism. *IEEE Intelligent System*, 17(6):61–63, November/December 2002.
18. B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of 10th International World Wide Web Conference ,WWW10*, Hong Kong, May 2001.
19. Scottish citylink online bus ticket booking. available at `http://www.citylink.co.uk/howtobuy.htm` Accessed on 19/4/2005 4:42 p.m.
20. T. Simcock, S. Hillenbrand, and B. Thomas. Developing a location based tourist guide application. In *Proceedings of the Australasian information security workshop conference CRPTIS'21*, ACSW frontiers, Australia, 2003.
21. R. Sinha and K. Swearingen. The role of transparency in recommender systems. In *Proceedings of Conference on Human Factors in Computing Systems*, pages 830–831, London, UK, April 2002.
22. E. Vozalis and K. Margaritis. Analysis of recommender system's algorithm. In *Sixth Hellenic-European Conference on Computer Mathematics and its Applications(HERCMA)*, Athens, Greece, September 2003.
23. A. Zipf. Adaptive context-aware mobility support for tourists. *IEEE Intelligent System*, 17(6):57–59, November/December 2002.

Keeping Track of the Semantic Web: Personalized Event Notification

Annika Hinze and Reuben Evans

University of Waikato, New Zealand
{hinze, rjee1}@cs.waikato.ac.nz

Abstract. The semantic web will not be a static collection of formats, data and meta-data but highly dynamic in each aspect. This paper proposes a personalized event notification system for semantic web documents (*ENS-SW*). The system can intelligently detect and filter changes in semantic web documents by exploiting the semantic structure of those documents. In our prototype, we combine the functionalities of user profiles and distributed authoring systems. Typically, both approaches would lack the ability to handle semantic web documents.

This paper introduces the design and implementation of our event notification system for semantic web documents that handles the XML representation of RDF. We analyzed our prototype regarding accuracy and efficiency in change detection. Our system supports sophisticated change detection including partial deletion, awareness for document restructuring, and approximate filter matches.

1 Introduction

In this project, we address the problem of alerting users of changes in semantic web documents. Some work on change detection in the semantic web (SW) has already been done; most of the projects focus on ontologies (see, Qin and Atluri, 2004). Here, we focus on changes in documents containing data or metadata.

A system for detecting, filtering and notifying about events is called an event notification system (ENS). We identified two possible approaches to our problem: either to extend a SW system with ENS functionality or to add SW support to an existing ENS system. We focused on the latter approach, extending a proven event notification system, and combining it with a well-accepted distributed authoring system to handle semantic web documents. Typically, both types of systems lack the ability to handle semantic web documents. In this paper, we introduce concept, implementation, and evaluation of the proposed system *ENS-SW*.

The details and challenges of our project are discussed after an introduction to background knowledge about Semantic Web technologies. Section 2 provides a description of the document formats RDF and RDFS, which are used in semantic web models. We identified several challenges for detecting changes or updates in these models. In Section 3, the detailed focus of our project is defined. In Section 4, we discuss the conceptual design of our system. Section 5 describes the prototype

R. Meersman, Z. Tari et al. (Eds.): OTM 2006, LNCS 4275, pp. 661 – 678, 2006.

implemented. Section 6 presents the evaluation to test the performance of the system under significant load. Section 7 outlines our conclusions from this work and identifies areas for future work.

2 Background and Project Focus

This section describes the context of the study and introduces the main concepts.

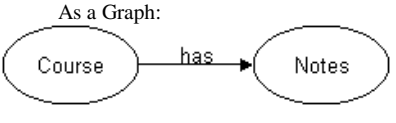
2.1 Brief Introduction to Semantic Web and RDF/S

The semantic web is an extension of the current Internet (Berners-Lee et al., 2001). Information contained in semantic web documents is enhanced by semantic annotations. Agents and services will give access to these data (metadata and knowledge).

Three complementary components, the Resource Description Framework (RDF), RDF Schema (RDFS) and Ontologies are the implementation level methods of representing metadata and its related knowledge representation within the semantic web.

RDF and RDF Schema

In order to have unique semantic annotations for representing items in the semantic web, it is not sufficient, or desirable, to have a single definition for all the semantic concepts needed. The Resource Description Framework (RDF) is a "general-purpose language for representing information in the Web" (Beckett, 2004). RDF has a triple structure that combines three resources: subject, property, and object. Several RDF triples can be combined to form RDF networks; a network can be defined in one or several documents. These RDF networks can be represented in a number of ways, for example by XML, Graph or in N3 notation. The example in Figure 1 shows a simple RDF relationship of a course to its lecture notes expressed in each of the three formats.



As a Graph:

Using N3 notation:

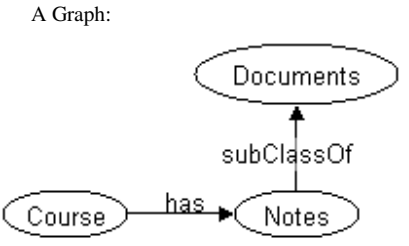
:Course a rdf:Class.
:Notes a rdf:Class.
:has a rdf:property.
:Course :has :Notes.

As XML:

```
<rdf:RDF>
  <rdf:Property rdf:about="#has"/>
  <rdf:Description rdf:about="#Course">
    <:has rdf:resource="#Notes"/>
  </rdf:Description>
  <rdf:Description rdf:about="#Notes"/>
</rdf:RDF>
```

Fig. 1. The three major ways of representing RDF structures

RDF provides the structure for the metadata, but is not sufficient to concisely describe complex semantic relationships. RDF Schema (RDFS) deals with knowledge representation in RDF. RDFS (Brickley and Guha, 2004) adds to RDF the ability to specify classes and more restrictive relationships between the resources described in the document. For example, we could use RDFS to extend our example from



As XML:

```
<rdf:RDF>
   <rdf:Property rdf:about="#has" />
   <rdfs:Class rdf:about="#Course" >
      <:has rdf:resource="#Notes" >
   </rdfs:Class>
   <rdfs:Class rdf:about="#Notes">
      <rdfs:subClassOf="#Documents">
   </rdfs:Class>
   <rdfs:Class rdf:about="#Docu-
ments"/>
</rdf:RDF>
```

Fig. 2. Extension of the example in Figure 1 to use RDFS

Figure 1, so that we can record not only that lecture notes relate to a course but that lecture notes are documents. An example of an RDF network using RDF Schema is shown in Figure 2 above. By converting our existing type descriptions to RDFS classes we turn our example into a schema, which, in turn, can be used to define instances of type Course and Notes.

Ontologies

One of the key requirements of the semantic web is the common definition of terms and concepts among agents. An ontology bridges the gap (Heflin, 2004) by describing both what an identifier means and how that identifier relates to other identifiers. An ontology can further provide rules for reasoning relationships between resources that are not explicitly linked by RDF triples but which are implied by those triples. RDF Schema defines basic ontological modelling primitives on top of RDF, e.g., the concept of a subclass, domain and range for properties. Other semantic web languages with richer modelling primitives, such as disjoints and rules, can be constructed by extending RDF Schema. Examples are DAML+OIL (Connolly et al., 2001) and OWL (Patel-Schneider et al., 2004). Here, we focus on documents using RDF and RDF Schema. Our system could be easily adapted for similar SW languages.

2.2 Problem Description

We will use the example of a semantic network for teaching-related documents, to illustrate the problem addressed in this paper. Figure 3 shows an RDF/RDFS network representing the data instances in the lower part of the schema (RDF) and the conceptual relationships between lecturers, courses and lecture notes in the upper part of the schema (RDFS in dashed lines). The network given here can be seen as a sub-section of the network that could describe all the courses and lecturers at the university.

A user may search the XML representation of this network (e.g. by querying the XML document shown in Figure 4) to retrieve the data. This is satisfactory as long as the semantic data and the underlying schema remain unchanged. Difficulties would arise if another user changes the information (e.g., about a lecturer of a particular course) or changes the structure (e.g., lecture notes are no longer tied to a course in general but to a specific semester). The user may want to know that their retrieved

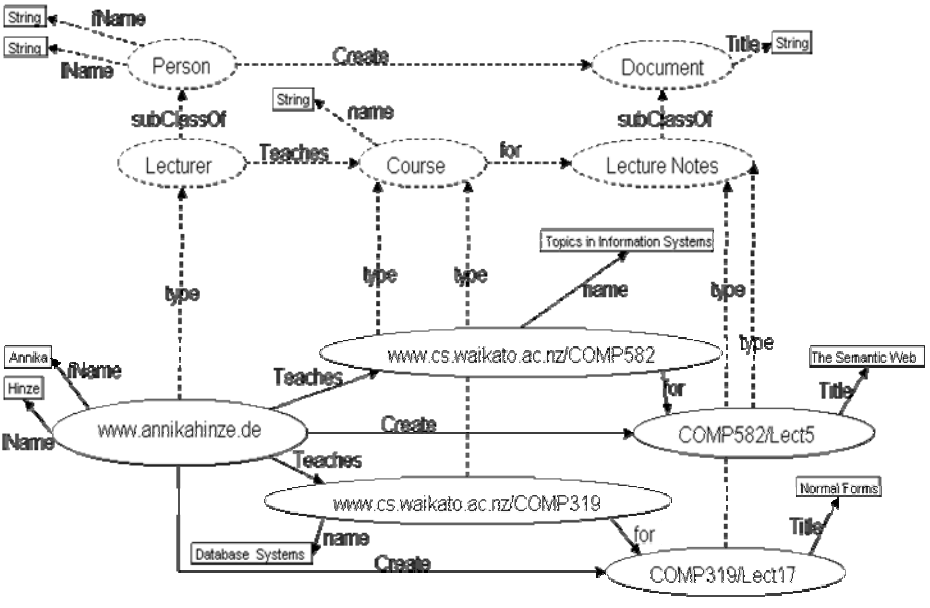


Fig. 3. A diagram of an RDF/RDFS network

```
<rdf:RDF xmlns=http://isdb.waikato.ac.nz/nonExistant/SchemaDoc#
        xmlns:rdfs=http://www.w3.org/2000/01/rdf-schema#
        xmlns:rdf=http://www.w3.org/1999/02/22-rdf-syntax-ns#>
        <rdfs:Class rdf:about="#Person" />
        <rdfs:Class rdf:about="#Lecturer">
                <rdfs:subclassof rdf:resource="#Person" />
        </rdfs:Class>
        <rdfs:Class rdf:about="#Course" />
        <rdfs:Class rdf:about="#Document" />
        <rdfs:Class rdf:about="#LectureNotes">
                <rdfs:subclassof rdf:resource="#Document" />
        </rdfs:Class>
        <rdf:Property rdf:about="#fName" />
        <rdf:Property rdf:about="#lName" />
        <rdf:Property rdf:about="#name" />
        <Lecturer rdf:about="#Annika">
                <fName>Annika</fName>
                <lName>Hinze</lName>
        </Lecturer>
</rdf:RDF>
```

Fig. 4. A portion of Figure 3 expressed in RDF's XML representation

result may have become invalid. Consequently, search is an insufficient mechanism to deal with changes in the data or data structure.

This problem becomes more pronounced if the schema is used as a global standard by universities around the world to categorise their lecture material. Then, the task of updating the network (with all implications) becomes complex. In addition, external users could use the data as a component in their semantic web documents. From this example scenario, the following questions arise:

1. How will user find out when the data they are using has changed?
2. How does the owner of a network know who needs to be notified about a change in their network?
3. How does a user find out when a part of a network to which they refer in their network is changed?
4. How does a user identify what needs to be changed in their network in response to a notification they have received about changes in a network to which they refer?

There are complex issues behind each of these questions. In this paper, we address the first two questions as a starting point of the larger problem outlined above. Our aim is to detect changes in semantic data and process such changes so as to notify the users or agents that have expressed an interest these changes. Consequently, the goal is to design and implement an event notification system for semantic web documents that exploits information about the semantic structure of the documents for filtering.

2.3 Principles of Event Notification

Users can define their interests in certain events, such as a change in a document, in profiles. An event notification system matches observed events to profiles in a process called 'filtering'. In the context of the semantic web, events are changes to the RDF files (e.g., new, changed, deleted). Typically, events have to be sent to the Event Notification System (ENS) by the producer, see Figure 5. They are filtered individually against an index of user profiles. Profiles are similar to continuous queries; they are created by users of the ENS to specify their area of interest. An ENS indexes profile queries, not documents. When the system receives a message about an event, it filters the message against the stored profiles and where a match is found, it notifies the owner of that profile about the event.

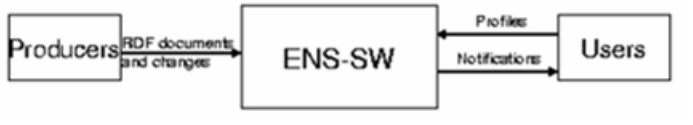


Fig. 5. User interaction with an event notification system

For our problem, new or changed RDF documents have to be submitted to the system to be matched against the profiles and then to be sent out to interested users. Figure 6 shows the internal components of the system: The RDF documents are processed by the observer which isolates the events within the documents and passes them to the filter. The filter compares the event messages to the stored profiles. Whenever it detects a match, it will pass the event and the profile to the notifier.

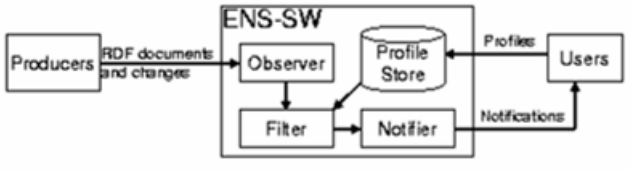


Fig. 6. Internal components of an event notification system

Proposed Solution

Typically, ENS neither support event observation in documents nor semantic web specific features. Our approach combines an event notification system with a distributed authoring system and adapts both components to adapt for application in a semantic web context. The system needs to support a profile language that can express RDF constructs. One may extend XML or RDF query languages to cover events, such as XML-QL (Deutsch *et al*. 1999) or RQL (Karvounarakis *et al*. 2002). As these components need to be tightly integrated, the system will include the profile store and notifier as well as the filter. The observer component has requirements that differ from the standard ENS. In Figure 6, we see the events coming in from the producers. However, the producers in this situation provide edited documents where very few of the triples have changed. They do not provide explicit event messages but whole documents. Accordingly, the observer component needs to be an active observer comparing new versions of semantic web documents that it receives against previous states of those documents. It will forward information about the actual changes, additions, and deletions that occur between one document version and the next. Accordingly, we chose to use a distributed authoring system that supports change detection and WebDAV (Whitehead and Goland, 1999). WebDAV is an extension of the http protocol that will allow the producers to easily supply their documents to the system.

4 Conceptual Design

Addressing research questions 1 and 2, we focus here on the design of a local system, which can later be extended as a broker component in a distributed system. Our *ENS-SW* system consists of five main components as seen in Figure 7: a repository for storing the RDF data, an observer to isolate changes in the RDF data, a store and index for profiles, a filter to identify matches between those changes and the collection of profiles, and a notifier to send out the notifications for profiles that have been matched. We now briefly describe each of the components.

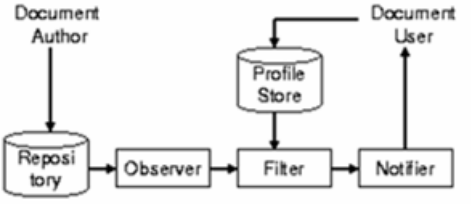


Fig. 7. Concept Architecture of ENS-SW

4.1 Repository

The repository designed either as active repository that includes some or all of the observer functionality or as passive repository that must be monitored and polled by an observer. We will use an active repository for which we identify three key requirements:

1. The repository needed means to represent change. Various forms can be envisioned, such as changes that are committed to a database in a transaction, or versioning through the use of diff files as in CVS. This will not be sufficient to detect all details of the events. Instead it will act as the trigger mechanism that alerts the event detector that some changes may have occurred. The trigger would also give an indication of where the potential changes have occurred.
2. The repository was required to represent data in the chosen format. For our design, we decided to consider the XML representations of RDF/S documents (see Figure 4). XML filter languages are sufficiently and prior approaches using RDF triple representations have shown the limitations of those languages. Consequently, the repository needs to support storage of XML data. It should also be able to indicate those documents that might have changes in XML format to the change detector. This requirement is fundamental due to our design decision to work exclusively with the XML representation of RDF.
3. The final requirement for the repository was that it should be accessible via the Internet to be included in larger semantic networks. This would also allow for easier system deployment. We identified WebDAV as the protocol that the system should support for document updating; WebDAV provides a powerful yet simple way for agents to access the repository.

4.2 Observer

The observer has complex functionality; we decided to design it as separate component. However, the observer needs to be tightly coupled with the repository. When presented with a file in which some form of change may have occurred, the observer has to (1) detect the change and (2) determine which effect the change in the XML document has on the structure of the RDF network that is described by the XML.

This problem is not trivial, as there are a number of different changes that can occur in a file that would not necessarily change the underlying RDF network. The reasons for this lie, amongst others, in the wide variety of ways in which a particular RDF network can be represented in XML. The syntax is verbose, allows considerable redundancy and does take into account the order of attributes. This means that there may be significant changes to a file without any change to the RDF network. In addition, simple syntactical changes, such as blanks, should be discarded. These constraints make it difficult to use many of the existing XML diff algorithms.

The observer component must be able to detect the following types of events:

- Modification: change to existing triples
- Insertion: creation of new triples
- Deletion: removal of existing triples

Modification events are the simplest ones; they can have only two different forms of a class substitution or literal value change. Class substitution occurs when the object of a triple is changed to some other resource within the RDF structure. Literal value changes involve the substitution of one metadata literal for another. For example, changing the name property of the resource COMP582 (see Figure 3) from ‘Topics in Information Systems’ to ‘Event-Based Systems’. Insertions and deletions both follow the structure as shown in Figure 8. For deletions, we can see the limitations of relying

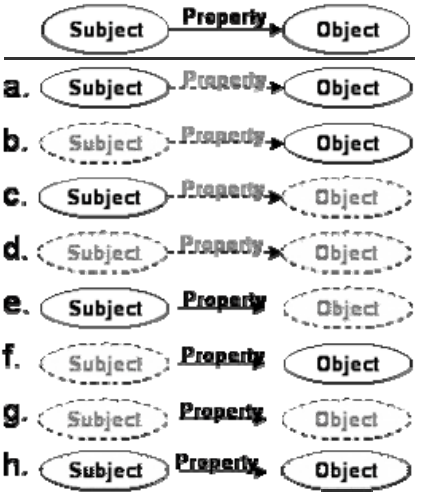


Fig. 8. The eight patterns of change

simple diff programs to obtain the event information: The available information is insufficient to detect the deletion pattern. In deletions, there are eight different event patterns. Multiple change events, each with a different pattern, could be caused by a single actual deletion within the RDF document, some of which may or may not be apparent from the event. Changes to the data may be inconsistent. Thus, deciding when to notify of a deletion involves more processing than just considering the individual resources that have been removed from the network. Consider the XML fragment in Figure 9, which defines four triples, on lines six, seven, nine, and ten. Using this as the base XML fragment, each of the eight deletion patterns is illustrated in Figure 10. Insertions follow the same patterns as deletions but in reverse. Detecting the number of new triples is not a straightforward operation but involves considerable processing of the surrounding XML. This problem could be reduced if the document editor would check that all references were updated when changing a document. However, this is not always feasible because of the large multi-file semantic networks that are used to describe and assist with real world problems.

```
1: <?xml version="1.0"?>
2: <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
3:    xmlns:dc="http://purl.org/dc/elements/1.1/"
4: xmlns="http://www.example.com/stuff#"
5: xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">
6:    <rdfs:Class rdf:about="Subject">
7:       <property rdf:resource="Object"/>
8:    </rdfs:Class>
9:    <rdfs:Class rdf:about="Object" />
10:   <rdf:Property rdf:about="http://www.example.com/stuff#property" />
11:</rdf:RDF>
```

Fig. 9. Example RDF document

4.3 Profile Store/Index

For the profile store and profile index, two critical factors exist: efficiency and ease of filtering. Both are served by the use of a tree storage structure which minimises the number of profile nodes that need to be visited by the filter for each event and simultaneously reduces the number of nodes that need to be remembered for each profile.

4.4 Filter

The primary action of the filter is the matching of event message to stored profiles. Ideally, the filter would use the changed RDF triples identified by the event

6: <rdfs:Class rdf:about="Subject"> 8: </rdfs:Class> 9: <rdfs:Class rdf:about="Object" />	Removed lines: 7 & 10 Triples removed: 2 Deletion type a.
9: <rdfs:Class rdf:about="Object" />	Removed lines: 6-8 & 10 Triples removed 3 Deletion type b.
6: <rdfs:Class rdf:about="Subject"> 8: </rdfs:Class>	Removed lines: 7,9 & 10 Triples removed: 3 Deletion type c.
	Removed lines: all Triples removed: 4 Deletion type d.
6: <rdfs:Class rdf:about="Subject"> 8: </rdfs:Class> 10: <rdf:Property rdf:about=http://www.example.com/stuff#property />	Removed lines: 7 & 9 Triples removed: 2 Deletion Type e
9: <rdfs:Class rdf:about="Object" /> 10: <rdf:Property rdf:about=http://www.example.com/stuff#property />	Removed lines: 6-8 Triples removed: 2 Deletion Type f
10: <rdf:Property rdf:about=http://www.example.com/stuff#property />	Removed lines: 6-9 Triples removed: 2 Deletion Type g
6: <rdfs:Class rdf:about="Subject"> 8: </rdfs:Class> 9: <rdfs:Class rdf:about="Object" /> 10: <rdf:Property rdf:about=http://www.example.com/stuff#property />	Removed line: 7 Triples removed: 1 Deletion Type h

Fig. 10. Deletion patterns in XML

observer/detector and map those to the registered profiles. The filter would need to handle XML namespaces and attributes. RDF Syntax encodes almost all significant features in these two forms; element tags and text nodes are rarely used. In addition, it is important that the profile language supports profiles that distinguish particular change patterns, as outlined before. Another question is the scope of the detected changes: When matching an event to a given set of profiles, should a profile that registers an interest in change in a given resource be notified on *any* change to a triple involving that resource or only when the resource *itself* is changed?

4.5 Notifier

The Notifier is the least complex component of the system. It sends notifications to users that own matched profiles. It should support a variety of means for notification to cater for different types of users: sending emails, RSS feeds, or making the notification available through a web site. Sending an email is good for occasional notifications to human users; a website or RSS feed would involve storing the notification on disk and retrieving it later when the feed was requested or the user logged into the site. Automatic messages via a given interface could cater for automatic/agent clients.

5 Implementation of *ENS-SW*

This section describes the implementation details of our XML-based ENS for the Semantic Web called *ENS-SW*. The repository and the observer are based on the authoring system SVN, which provides the necessary data storage functions and basic

change detection. Subversion (SVN) is a relative of CVS (Collins-Sussman et al., 2004). It supports distributed authoring of documents. The detailed event detection is performed by a Delta Adapter component that we developed. It receives documents from the SVN and performs an XML diff to locate the relevant changes. It analyses the XML differences in the document before and after the changes were made. For the profile store and filter components, we extended the XML-based ApproXFilter (Michel and Hinze, 2005) to process RDF data. Our system exploits knowledge of the structure of the RDF Schema to create useful notifications to interested users. The ENS-SW can be accessed using the WebDAV protocol.

The system consists of three components as shown in Figure 11: a repository (SVN), an observer that is tightly coupled to the repository (Delta Adapter), and an event filter (ApproxFilter). Whenever SVN receives a new or changed document it activates the adapter program by the use of a hook script. This program obtains a list of all changes to the repository since the latest revision. In this list, the adapter identifies the RDF files that changed in this revision and obtains both the version before and after the revision (see example in Figure 12).

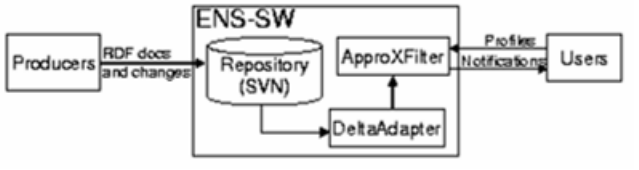


Fig. 11. Components of the *ENS-SW* system and their interactions

The Adapter accesses copies of all changed RDF files (before and after the change) and uses DeltaXML libraries (La Fontaine, 2001) for the comparison. DeltaXML performs an unordered Tree-based diff. It records for each sub tree the insertions, changes, deletions or if the node and all its children remain the unchanged. The resulting mark-up that DeltaXML in the XML file can be seen in Figure 13.

Although Delta XML identifies the changes in the XML file, it is not sufficient for identifying RDF events. Triples are interconnected; a change on one part of the network may affect other resources. In RDF, unlike in XML, a class and its properties

RDF Before Editing	RDF After Editing
<rdf:RDF xmlns:rdf="…" xmlns:rdfs="…"> <rdf:Property rdf:about="#Teaches" /> <Course rdf:about="#COMP582"/> <Lecturer rdf:about="#Annika"> <fName>Annika</fName> <lName>Hinze</lName> <Teaches rdf:resource="#COMP582"/> </Lecturer> </rdf:RDF>	<rdf:RDF xmlns:rdf="…" xmlns:rdfs="…"> <rdf:Property rdf:about="#Teaches" /> <Course rdf:about="#COMP582"/> <Lecturer rdf:about="#Annika"> <fName>Annika</fName> <lName>Hinze</lName> <Teaches rdf:resource="#COMP319"/> </Lecturer> </rdf:RDF>

Fig. 12. Portion of an RDF file in XML changes made to it

```
   <rdf:RDF xmlns:deltaxml="…" xmlns:rdf="…" del-
taxml:delta="WFmodify">
    <rdf:Property deltaxml:delta="unchanged" rdf:about="#Teaches"/>
    <Course deltaxml:delta="unchanged" rdf:about="#COMP582"/>
    <Lecturer deltaxml:delta="WFmodify" rdf:about="#Annika">
    <fName deltaxml:delta="unchanged" >Annika</fName>
    <lName deltaxml:delta="unchanged" >Hinze</lName>
        <Teaches deltaxml:delta="WFmodify" deltaxml:old-
attributes="rdf:rescouce=&quot;#COMP582&quot;" deltaxml:new-
attributes="rdf:resource=&quot;#COMP319&quot;"/>
    </Lecturer>
   </rdf:RDF>
```

Fig. 13. The results of running DeltaXML on the files in Figure 12

may be kept separate from one another. Profiles interested in changes in a property that is used with a class in this manner will be interested in events that occur on the class as well as directly on the property. To correctly identify the events, the results from DeltaXML are passed through an XML style sheet transformation (XSLT). For example, it transforms the result of the Delta XML from Figure 13 into the event message seen in Figure 14. It removes superfluous DeltaXML information, e.g., about unchanged elements.

```
<Annika>
<action>modify</action>
<Teaches>COMP582
        <action>delete</action>
</Teaches>
<Teaches>COMP319
        <action>add</action>
</Teaches>
</Annika>
```

Fig. 14. Message after XSLT transformation

The DeltaXML libraries only work with changed documents. For the addition and deletion of entire documents, the system simulates a change by creating a temporary, empty document. This enables us to treat these events in the same manner as changed documents and thus allow for filtering.

Profile	Interpretation
1) Annika [Teaches[“*”]] and Teaches[action[“add”]] 2) Course[action[“modify”]] 3) Lecturer[action[“add”]] 4) Lecturer[fName[“*”]]	1) Annika teaches another course 2) A course is changed 3) Another Lecturer is added to the network 4) Any events that affects a Lecturers First Name

Fig. 15. Some profiles that users might register on the data in Figure 12

The next step is to filter the event message using an XML filter algorithm. We employ the XML filter algorithm ApproXFilter (Michel and Hinze, 2005). We adapted ApproXFilter to match profiles regarding triple changes in the makeup the RDF document (for example profiles see Figure 15).These profiles specify a set of

elements that the user is interested in and the types of changes that will trigger a notification. Elements are separated by square brackets to enable the user creating a profile to specify the relationships that the elements must have to one another.

In our implementation supports three points of user interaction: Document authors submit documents to the SVN server either by using WebDAV in an Apache server, the dedicated SVN server, or directly to the repository. New profiles enter the filter via a connection to port 8088; the system inserts them to its profile tree. Direct insertion via the profile directory is also possible. Finally, notifications are sent via email.

6 Evaluation

In this section, we report the results of our performance evaluation. We designed two main tests: The first test evaluated the efficiency and accuracy of the Delta Adapter program. The second set of tests evaluated the performance of the filter for different numbers and types of profiles. The separate parts of the system were tested independently of each other since they are distinct programs that co-operate.

6.1 Observer (DeltaAdapter) Testing

These tests determine the efficiency of the event detection process in the observer. Since deletions and insertions use the same operations, we only test triple deletions. The variables for the observer tests are shown in Figure 16. O_T and N_T represent the size of the input document before and after changes, respectively. They influence the Delta generation as each document is created as a tree and then compared node by node. C denotes the number of triples that have changed between O_T and N_T. The more changes the more time is consumed by the tree comparisons. C_S is large when a high number of changes is large compared to the document size. Processing a change with large C_S should be very quick. Large C_I adds noise to the filtering process.

Symbol	Description
O_T	Number of triples in original document
N_T	Number of triples in changed document
C	Number of changed triples
C_S	Changes relative to size of document (C/O_T)
C_I	Number of Changes that do not represent a triple change

Fig. 16. Variables for observer tests

Observer test 1 – Triple change test
Goal: Test the performance of the observer. We use a specified number of changes in RDF documents of increasing size.
Hypothesis: Regardless of the number of changes it is expected that there will be a linear increase in time taken to process a document as the number of triples increases.

Figure 17 shows the time to process a document for increasing numbers of triples (one triple removed, $N_T = O_T - 1$, $C=1$, $C_I=0$; all triples removed, $N_T = 0$, $C= O_T$, $C_I =0$). We see the linear dependency on the number of triples in the document. This

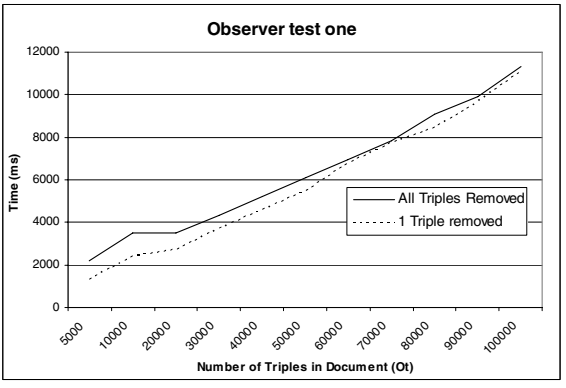


Fig. 17. Triple change test

behaviour is independent of the number of triples removed. The jump between 10000 and 20000 triples is investigated further in Figure 18, which shows the average time per triple in milliseconds. This graph clearly shows the heavy influence the size of the document has on the speed of event detection. The reason is that the system has to build DOM trees in main memory for both documents and then compare to find the minimum difference. This graph shows the influence of the two different costs associated in the change detection process: below 20,000 triples the initialisation structures used to compare the XML files is the main influence. The second factor is the cost of comparing the DOM trees; it dominates above 30000 where the fixed initialisation costs are divided among many more triples.

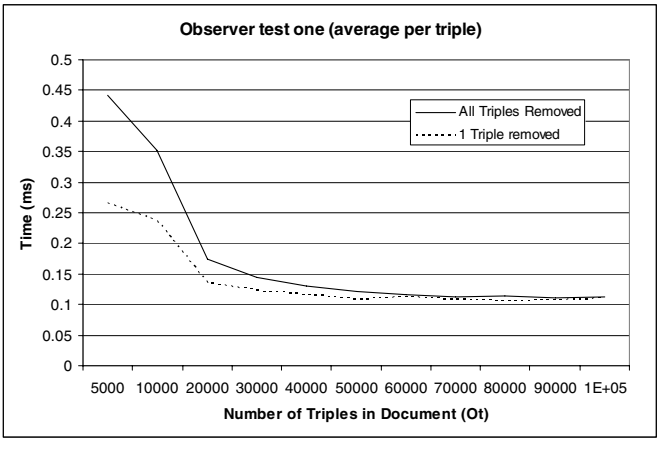


Fig. 18. Triple change test (averaged)

Observer test 2 – Non change test
Goal: Test the effect of a change that does not change the meaning of the RDF on the performance of the Delta Adapter.

Hypothesis: regardless of the number of changes, it is expected that there will be a linear increase in the time it takes to process a document with increasing numbers of non triple changes.

We observe that an increasing number of non triple changes cause an increase in the processing time. It is important to note that O_T and N_T were identical in this test

Fig. 19. Non triple change test

(Parameters: N_T = 10,000 O_T=10,000, C=0, Cs=0). All the changes involved adding extra c arriage returns or swapping the order of child nodes, neither of which makes any difference to RDF. There were no effective RDF changes; the system had to match the different portions of the DOM tree to confirm that nothing had changed in the model. Comparing Figure 19 with the respective data in Figure 17 (10,000 triples) one finds that five non structural changes to the RDF file result in the doubling of the observer time.

6.2 Filter Testing

The second series of tests are efficiency tests for the filter. Because the system performs approximate matching, the number of profiles that match an event is not a factor in the processing time. The filter does not stop unless all profiles and the whole document are tested. Non-matches create costs for each profile. The match cost also do not influence the performance. Only those profiles with match cost below a given threshold will be notified. The variables for the filter tests are described in Figure 20. E affects the filtering as each member of the triple needs to be checked against the profile tree. P are P_C expected to be major influences on the performance. P_D indicates the increases in the size of the filter tree. P_C is important since every conjunct adds between one and three extra nodes to the tree for the profile. This increases the number of nodes to be evaluated. S represents equivalent words. The size of the profile tree increases linearly with the size of the synonym net. The parameters for the tests are E = 2, P = 10K – 100K, P_D=0, S=0, P_C=0,1,2.

Symbol	Description
E	Number of Triples in Event Message
P	Number of profiles
P_D	Degree of independence in profiles
P_C	Average Number of Conjuncts in profile
S	Number of terms in the synonym net

Fig. 20. Variables for filter tests

Filter test 1 – Event parsing Test
Goal: Test the effect of increasing number of distinct profiles and number of conjuncts on the event parsing stage of the filtering process

Hypothesis: We expect than an increase in the number of distinct profiles should have a linear effect on the processing time. Increasing the number of conjuncts should not affect the runtime of the event parsing.

The results are given in Figure 21. We observe a linear influence of the number of profiles on the performance due to the increase in the size of the profile tree caused by the increase in the number of distinct profiles. The influence of the number of conjuncts is not statistically significant.

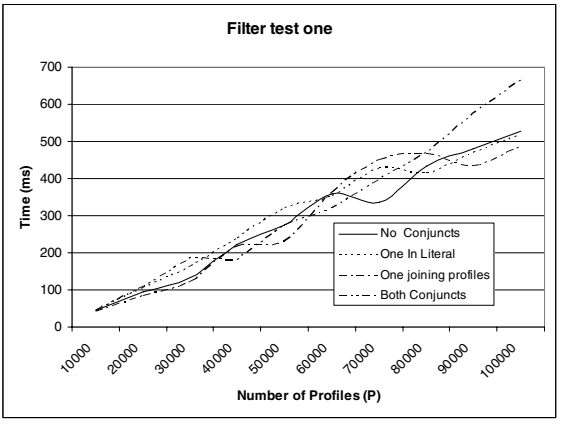


Fig. 21. Event parsing test

Filter test 2 –Profile matching test

Goal: Test the effect of increasing the number of distinct profiles and the number of conjuncts on the profile matching.

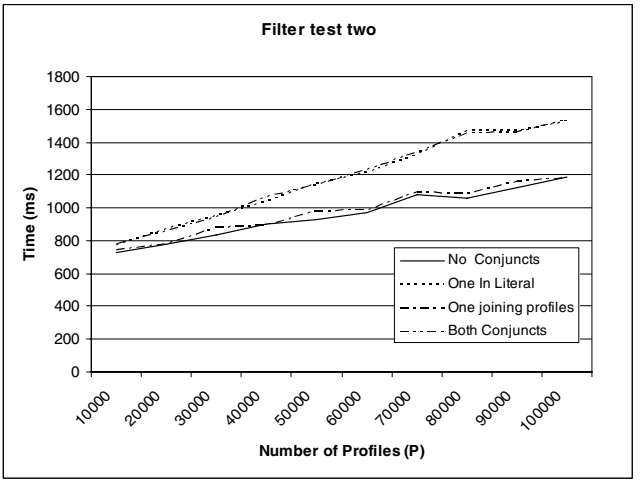


Fig. 22. Profile matching test

Hypothesis: An increase in the number of profiles has a linear effect on the runtime of the filtering process. Increasing the number of conjuncts should have a linear effect on the runtime due to the extension of the filter process.

We observe that the number of profiles has a linear influence on the performance. This relationship is not as pronounced as we expected, because of the considerable initial

overhead. The effect of adding conjuncts is interesting: when adding a conjunct to the end of the profile we achieved the expected linear increase in profile match time (see Figure 22). However, when the conjunct was joining two distinct triples, the profile matching was as quick as if there had been only one. We believe that this is caused by the profile cost calculation: Once the filter has checked a profile it evaluates the cost of that profile to see if it exceeds the cost threshold. When a profile consists of multiple independent parts connected by a conjunct, the filter can stop processing that profile. This would account for conjuncts joining two independent profiles taking the same time as single profiles with no conjuncts. Profiles incorporating both types of conjuncts behaved exactly as the first type with only one conjunct.

7 Conclusions and Future Work

We identified four research questions to be addressed to keep track of changing semantic web models. In this paper, we focussed on two closely related questions from that list: how to detect changes in the semantic web data and how to identify users who need to be notified about a change in the SW network? We presented a common solution for both challenges. Our *ENS-SW* system allows the user to register a profile indicating an interest in a portion of the RDF network stored in the repository. Whenever a change occurs on the network, it will be filtered against all the profiles stored in the ENS, and when a match is found, the corresponding users will be notified. Our evaluation has shown that the system performs as expected.

The key points of our system are as follows:

(1) Easy Deletion Detection: Detection of deletions is often ignored in ENS due to the complexity. Often, a mirror would need to be used for the data repository. Since we store the data in the form of deltas, deletion detection becomes a trivial task.
(2) Using Subversion and diff: To the best of our knowledge, versioning software or XML diffs have not been used to detect changes in XML data. Typically, only new XML files can be processed by filters.
(3) Using the ApproXFilter algorithm: The matching algorithm supports approximative filtering. A user can state how close a match has to be to count as a match. This approach removes the need for a user to register several similar.

For future research, we plan to address the challenge of distribution. The current system supports connection to any number of observers that each connect to one filter only. Support for XML namespaces and attributes is also planned to be added to the system, both on the observer side and for the filtering. Finally, two further questions identified early in the paper remain open for future research. The fourth question – that of identifying what needs to be changed in an RDF network in response to a notification – is a particularly interesting and significant problem. We aim to automatically adapt the network in response to notifications received from other filters in order to automatically synchronise distributed semantic web networks.

References

1. Beckett, D. (2004) "RDF/XML Syntax Specification, W3C Recommendation, 10 February 2004." Available at http://www.w3.org/TR/2004/REC-rdf-syntax-grammar-20040210 (23 March 2005).
2. Berners-Lee, T., Hendler, J. and Lassila, O. (2001) "The Semantic Web." Scientific American 284(5):34-43; May.
3. Berners-Lee, T., Fielding, R.T. and Masinter, L. (2005) "Uniform Resource Identifier (URI): Generic Syntax." Available at http://www.gbiv.com/protocols/uri/rfc/rfc3986.html (23 March 2005).
4. Brickley, D. and Guha, R. V. (2004) "RDF Vocabulary Description Language 1.0: RDF Schema, W3C Recommendation 10 February 2004." Available at http://www.w3.org/TR/2004/REC-rdf-schema-20040210 (23 March 2005).
5. Broekstra, J., Kampman, A., van Harmelen, F. (2001) "Sesame: An Architecture for Storing and Querying RDF Data and Schema Information." In Semantics for the WWW, edited by D. Fensel, J. Hendler, H. Lieberman and W. Wahlster, 2001. MIT Press, Boston, Massachusetts.
6. Collins-Sussman, B., Fitzpatrick, B.W. and Pilato, C. M (2004) Version Control with Subversion. O'Reilly, Cambridge, Massachusetts.
7. Connolly, D., van Harmelen, F., Horrocks, I., McGuinness, D.L. and Patel-Schneider, P.F. (2001) "DAML+OIL (March 2001) Reference Description W3C, Note 18 December 2001." Available at http://www.w3.org/TR/2001/NOTE-daml+oil-reference-20011218 (24 March 2005).
8. Deutsch A., Fernandez M., Florescu D., Levy A., Suciu D. (1999) "XML-QL: a Query Language for XML". Proc. of the Int. World Wide Web Conference (WWW), Toronto
9. Gibbins, N., Harris, S. and Shadbolt, N. (2004) "Agent-based Semantic Web Services." Web Semantics: Science, Services and Agents on the World Wide Web 1(2):141–154.
10. Heflin, J. (2004) "OWL Web Ontology Language Use Cases and Requirements, W3C Recommendation 10 February 2004." Available at http://www.w3.org/TR/2004/REC-webont-req-20040210 (24 March 2005).
11. Hinze, A. (2003) "A-mediAS: An Adaptive Event Notification System." Proc 2nd International Workshop on Distributed Event-based Systems, San Diego, USA.
12. Karvounarakis, G, Alexaki, S,Christophides V, Plexousakis, D, Scholl, M,(2002) "RQL: a declarative query language for RDF." In WWW, pages 592–603, 2002.
13. Klyne, G. and Carroll, J.J. (2004) "Resource Description Framework (RDF): Concepts and Abstract Syntax, W3C Recommendation, 10 February 2004." Available at http://www.w3.org/TR/2004/REC-rdf-concepts-20040210 (24 March 2005).
14. Kozuka, T. (2004) "The Adaptive Semantic Web."; Comp 591 Dissertation, Department of Computer Science, University of Waikato, Hamilton, New Zealand.
15. La Fontaine, R. (2001) "A Delta Format for XML: Identifying Changes in XML Files and Representing the Changes in XML." Proc XML Europe 2001, Berlin, Germany.
16. Michel, Y. and Hinze, A. (2005) "ApproxFilter – An Approximative, XML-based Event Filter." Technical Report 06/2005, Department of Computer Science, University of Waikato, Hamilton, New Zealand.
17. Patel-Schneider, P.F., Hayes, P. and Horrocks, I. (2004) "OWL Web Ontology Language Semantics and Abstract Syntax, W3C Recommendation 10 February 2004." Available at http://www.w3.org/TR/2004/REC-owl-semantics-20040210 (24 March 2005).

18. Qin, L. and Atluri, V. (2004) “Ontology-guided Change Detection to the Semantic Web Data.” Proc 23rd International Conference on Conceptual Modeling (ER2004), Shanghai, China, pp. 624–638.
19. Silva Filho, R.S., de Souza, C.R.B. and Redmiles, D. F. (2003) “The Design of a Configurable, Extensible and Dynamic Notification Service.” Proc 2nd International Workshop on Distributed Event-based Systems, San Diego, USA.
20. Whitehead, J. and Goland, Y. Y., (1999) WebDAV: A NetworkProtocol for Remote Collaborative Authoring on the Web. In Proc. of the European Computer Supported Cooperative Work Conference (ECSCW’99),. Available at http://www.webdav.org.

A Gestures and Freehand Writing Interaction Based Electronic Meeting Support System with Handhelds

Gustavo Zurita[1], Nelson Baloian[2], Felipe Baytelman[1], and Mario Morales[1]

[1] Department of Information System and Management of the Economy and Businesses School, University of Chile, Diagonal Paraguay 257, Santiago, Chile
gnzurita@usistemas.cl, felipe@baytex.net, mmorales@usistemas.cl
[2] Department of Computer Science of the Engineering School of the University of Chile, Blanco Encalada 2120, Santiago, Chile
nbaloian@dcc.uchile.cl

Abstract. In this work, we present an Electronic Meeting Support system for handhelds. The design principles applied for developing the system are aimed to help reduce the problems associated with having a small size screen to interact with. The human-handheld interaction is based only in gestures and freehand writing, avoiding the need of widgets and virtual keyboards. The content of the generated documents are organized as concept maps, which gives more flexibility to reorganize and merge the contributions of the meeting attendees. Our system is based on handhelds interconnected with an ad-hoc wireless network. The system architecture is a peer-to-peer one, avoiding the need of central repositories thus allowing meetings to take place anywhere.

1 Introduction

In any organization, small teams and work groups undertake a broad spectrum of face-to-face meeting processes, including deliberation, negotiation, consensus building, decision making, idea generation, problem solving, planning, [1], [9], [25], which are becoming increasingly important in most business initiatives, [2]. An Electronic Meeting Systems (EMS) is an interactive computer-based system for supporting meeting processes. [1]. EMS goal is to help group members to overcome the well-known problems that arise in a meeting, thus improving its productivity, efficiency, and effectiveness. Since a group meeting may involve different activities, the system should be flexible enough to support most of them.

Technology for supporting face-to-face meetings has been around for nearly two decades using mainly desktop computers, networks and software implementing sophisticated awareness mechanisms. However, adoption statistics measuring their "accessibility and availability" to organizational end-users have been somewhat disappointing. The findings of [25] indicate that EMS systems have been adopted in organizations in the USA; Australia and New Zealand only to a limited extent. Several possible barriers to EMS adoption have been suggested and a recent survey of EMS researchers also identified several potential obstacles to their adoption, [25]. Some organizational issues

R. Meersman, Z. Tari et al. (Eds.): OTM 2006, LNCS 4275, pp. 679 – 696, 2006.

such as compatibility of EMS with user cognitive styles, lack of organizational incentives, resistance to change combined with difficulty in measuring and demonstrating EMS benefits are the most commonly perceived barriers. These findings also are supported by [19], who stated "for Groups Support software to be an organizational success, one must plan for and overcome extreme resistance to change".

It has been also mentioned that another fundamental reason for the low level of adoption and use of EMS is that they still lack of fundamental functionalities supporting the creation of an agenda or an agenda-setting process, implementation of common workspace for participants, support for drawing up of minutes, procedures for supporting follow-up on commitments, and voting mechanisms [25]. Although some successful systems have been developed and tested based on desktop computers ([10], [11]) it has been demonstrated in [3] and [28] that the EMS interfaces on PC and notebook capture the attention and cognitive concentration of participants to such an extent that face-to-face interaction is reduced. According to [28], handhelds are a more suitable support tool for face-to-face meetings. In fact, handheld portable computer devices are non-obstructive and create a feeling of belonging because they are used in various activities and may be used in any place and used at any time [16], [17] and [20].

In this paper we propose a face-to-face EMS system based exclusively on the use of handhelds wirelessly connected through a peer-to-peer ad-hoc network. The aim of the system is to overcome in a simple way the cognitive styles, adoption, resistance to change, and some meeting problems (lack of common work space for members, difficulties while drawing up the minutes, lack of follow up commitments, and absence of voting mechanism among others). The system supports various processes of a meeting life-cycle, both individual and collaborative. The user interface is based on freehand input allowing: a) freehand writing and sketching over the screen of the handheld for taking notes; b) activation of special functionalities for, creating, managing and navigating through conceptual maps; c) synchronous and asynchronous coordination of the work and; d) voting.

2 Related Work

Some analyses have been carried out of both proposed and already-developed EMS systems that use freehand input, concept maps and especially handhelds, as well as the functionalities offered by handhelds for supporting face-to-face meetings: agenda creation, distribution and discussion support; task and processes development support; distributed on-screen viewing; individual note-making; and generation of minutes.

The We-Met project [30], supports face-to-face meetings although using Tablet PCs (TPC) for each of the participants all of whom are interconnected through a TPC. Attendees can work in the same virtual work area on their tablet screens, which is shared through the connection with the TPC and is freehand input-based. The project's objectives are (a) to facilitate communication between meeting participants, and (b) to facilitate documentation of knowledge and information generated by the meeting for easy review. Users of this system found that it was necessary to have private work areas where they can develop ideas that are not yet ready to be presented to the other attendees. The Pebbles project [18], though not conceived to be used exclusively for meetings, can be used to provide support to collaborative groups in various

contexts. It consists of applications that interconnect handhelds through a PC. The devices are used as though they were PC mice or keyboards. The project's objective is to mediate social interaction techniques between persons through a shared screen. RoamWare [28], is a handheld architecture that supports informal face-to-face reunions, including those held in such places as corridors. Each handheld can detect and interconnect to others located within a limited space, while the participants make notes on their devices. These notes are sent to a central computer where they are stored for later distribution. Costa et al. (2001) have developed the idea of combining handhelds and a PC to explore the relationships that may exist between a meeting and these technologies. They show that the use of handhelds is neither annoying nor obstructive to the flow of the meeting, and suggest the devices be utilized as tools to generate reports, a traditional technique for linking meeting processes to organizational ones. The authors of the study also attempt to improve meeting report generation by making use of the capacities handhelds can contribute to the EMS for managing individual and group information. [1] have studied the impact of including handhelds as a support to meetings, pointing out the important role they can play in managing individual information. The authors note the following requirements: a) creation, distribution and support for the development of an agenda; b) recording of decisions taken; c) inclusion of the foregoing in the minutes for later distribution; d) support for typical meeting structures; and e) support for various agenda, issue, decision, report and logistics templates.

We made a comparative analysis of the above-described meeting support systems. Antunes and Costa [5] are the only ones to propose the creation, distribution and discussion of the agenda. None of the systems provides any support for negotiations aimed at reaching agreements or for commitment follow-up. We also see that there is no system trying to overcome the typical problems of handhelds by exploiting the gesture based interaction or improving the structure of the document. This work focuses on incorporating those ideas in the design principles of a handheld based EMS.

3 Design Principles

Handhelds are considered to be a good platform for reading brief, concrete content because their interface is simple and insensitive to content formats, thus allowing information to be read quickly. They are also considered to be suitable for providing support to diverse collaborative work groups, [23]. However, their reduced screen size and use of virtual keyboards or widgets for entering and handling information introduces new complexities into the user-handheld interaction, [6]. In order to overcome these problems we propose the following design principles:

- Interaction is based exclusively on gestures thus minimizing the number of widgets and virtual keyboards and maximizing the space available for entering content. The content will be exclusively free handwriting. Although free handwritten text may take more space than typed text, it allows also a flexible combination of sketching and writing. According to [6] and [15], sketching and gesturing with pen-based systems are natural modes of design-task-oriented interaction. In [26] it is noted that a sketch is a quick way of making designs that a) facilitate the creator's idea generation process, b) stimulate communication of ideas with others, c) stimulate the use of early ideas thanks to the accessibility and interpretation

they provide, and d) gives the opportunity to see and get inspired by other group members' ideas.

Many systems use the metaphor of pages and/or scrolling bars for the documents generated in order to offer more "space" to the user. This is indeed a simple and very intuitive way of organizing the content of a document, but when the working area is extremely small, which is the case of handhelds, is seems to be better to organize the content as a structure which is also intuitive and may contain more information without having to enter more data. A structure like a concept map will be indeed more suitable to order the ideas generated during the meeting since this is an intuitive yet flexible structure which adds more information to the content than the "list of pages" (which in fact is a simple form of a concept map) without having to input more content.

A decision-meeting often follows the template of idea generation, idea organization and idea prioritization [1]. Idea generation is a divergent phase where group members typically engage in a kind of brainstorming activity, drawing, talking, and sketching. This has also been called *brainsketching*, [26]. This phase is characterized by creativity, freethinking, lack of critical analysis and lack of restrictions or controls. After ideas are generated or explained by text note-taking, or sketching, they need to be organized. This activity includes accepting/rejecting ideas, refining, rearranging, and consolidating items representing the ideas. The final phase should produce a summary of the ideas as a list of the issues or topics that are most important or relevant. From these three phases, the organization of the different ideas contributed by the group members is often the most problematic. The output of an electronic brainstorming or brainsketching session is typically extremely "dirty", and needs to be organized in an easy way by using editing, marking, deleting and cutting. The graph structure of a concept map offers a more flexible way to do this than a list of pages. The use of concept maps enables the drafting of a meeting summary through the follow-up of the structure, and facilitates the determination of actions to be taken.

Using a graph structure for a document opens new challenges which include the development of an easy gesture-based mechanism to create and fill with content this structure. Also the design of the document viewing options the system should provide is an interesting issue. These issues will be addressed later in this paper.

In addition to the principles mentioned above, an EMS should provide some fundamental functionality. Because EMS are mostly aimed at supporting small group work in a face-to-face setting, they should provide the capability to share information, exchange ideas, express opinions, create solutions, develop consensus to resolve problems, and collaborate on tasks. This can be achieved by a shared workspace where the members of the meeting would write, depict or sketch their ideas simultaneously in a common workspace by freehand input based system.

According to [27], voting before discussing certainly seems to increase the sense of group identity, reduce personality conflicts, and reduce needless discussion. Some experimental studies have shown that computer support helps to generate agreement among the members. In a meeting support system voting is used as a rational choice tool, being usually highly formal and used once at, or towards the end of the meeting.

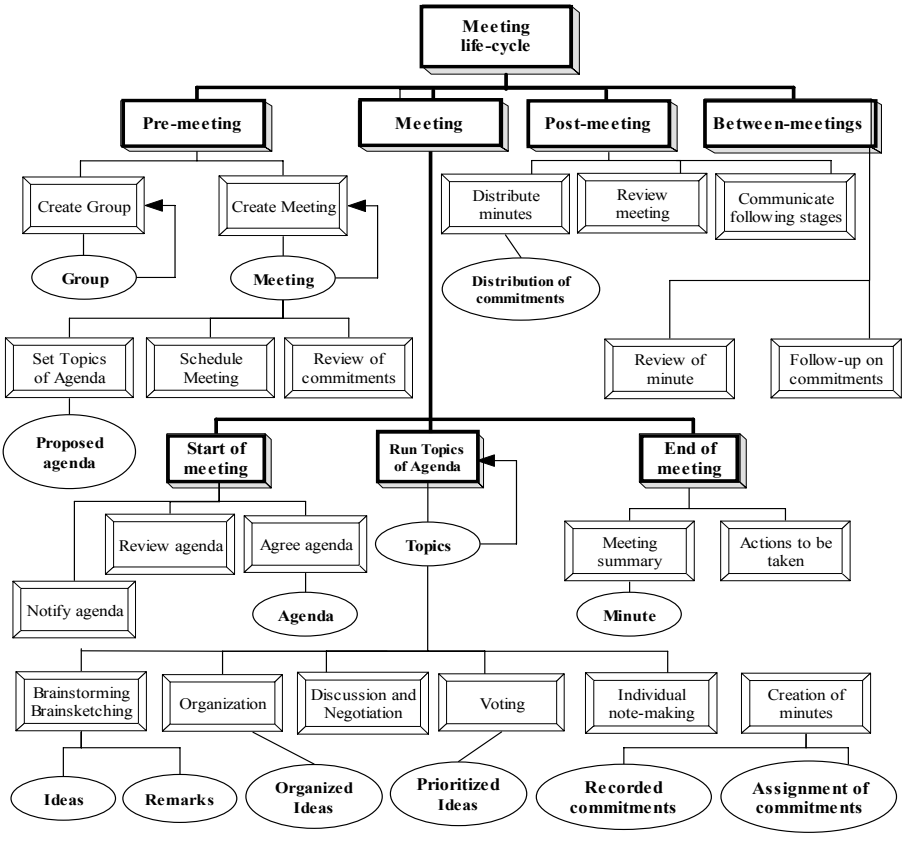


Fig. 1. Hierarchical diagram of the meeting system

Synchronizing the work of a group may represent a higher challenge, especially if we want to support the collaborative work synchronous and asynchronously at the same time. Problems arise when it comes to merge the work of participants who have been working asynchronously with people working synchronously. Simple rules for avoiding inconsistencies are needed. A method based on working on private notes and publishing them to the group serves to this purpose and at the same time allows a participant to try sketching/writing of new ideas before showing them to the rest.

4 Meeting System Structure

The Fig. 1 shows a hierarchical diagram of the meeting life-cycle, representing the stages for the tasks and processes that can be supported by the meeting system. A meeting (or a number of meetings) is triggered by a problem the group has to discuss and solve.

In any stage of the meeting, users record their ideas by making private remarks or comments about specific and ideas by individual note-taking. The brainstorming process is characterized by the generation of possible solutions, which are registered by the system and distributed to all members in order to be evaluated. This process can be carried out by anonymous and/or nominal contributions. In a face-to-face meeting group members can discuss about the convenience or inconvenience of the different ideas. Because all processes and tasks are reflected by annotations on the handheld, the minute of the meeting can be created as the different possible discussion topics are being proposed. This generated the documentation of a meeting which could also be complemented with a log of actions being taken, like additions, deletions or modifications made to the document for a posterior analysis and evaluation. Agreed commitments could be registered by mean of annotations or sketches, and distributed to the corresponding members at the end of the meeting. The system may manage a register which will be used to control the fulfillment of the assignments.

5 System Description

The basic content of a meeting document are the notes produced by the group member during the discussion by sketching and writing. In this section we will explain how our system combines the input of all the members in a structured document, based on nodes and relations among them thus building a concept map. Also, as mentioned in section 2, other tools and functionalities required enhance a group meeting will be presented. Current implementation has been developed with Microsoft .NET 2.0 Compact Framework's C#, and deployed on Windows Mobile 2005.

In order to describe how the system allows collaborative writing and content managing, we will first review each one of these individual functionalities in order to simplify the system understanding. After each feature, we will shortly describe how it has been implemented. At the end of this section, we describe the collaborative process and the system architecture, including the communication protocol.

The basic action a user is allowed to do in a meeting document is to enter free hand writing and sketching. A free-hand input is done by "writing" with the stylus any figure which is not recognized by the system as a special gesture. This hinders the user for entering a sketch with is similar to a gesture, but this is better than having to switch from a "sketching mode" to a "writing mode" back and forth with a special functionality. Therefore, the system must be able to recognize free-hand writing and sketching from special gestures. In order to achieve this, the system tries to match the drawn figure with each possible gesture only after the user raises the stylus tip. The analysis of the input is made instantly after each stroke is made, without interfering in the writing/drawing process. In case a gesture triggers an action, the system stops matching the figure with the next gestures and performs the corresponding action. In case no gesture has been acknowledged, the system classifies the stroke as writing and "writes" it in the document.

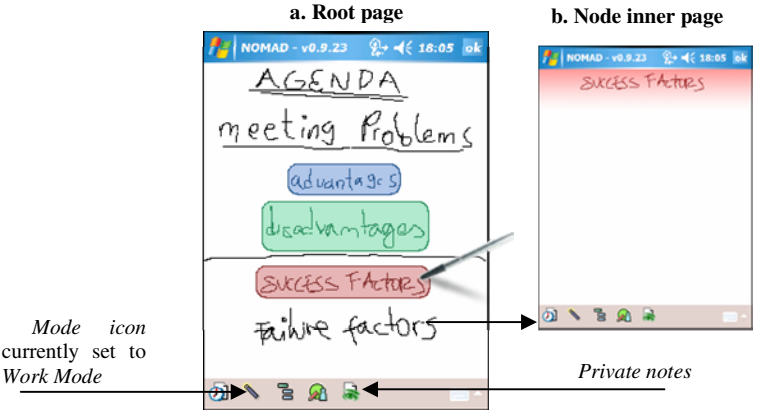
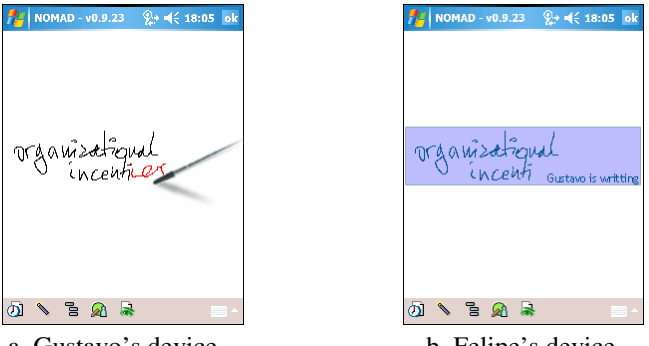


Fig. 2. System's screenshot during sample meeting: Double tapping a node (2.a) displays the node's inner page (2.b)

5.1 Annotations and Sketching

Writing and drawing is made through freehand input with the handheld stylus. The system will analyze the drawn figure and, after discarding every possible action-gesture, accepts it as written symbol. Repeating this process, the user is able to write sentences or draw complex figures, representing his or her idea.

Because the drawings are shared with other participants only when the pen tip is raised, other users might start drawing in the same areas where others are already writing. This may lead some users into confusion, because they were not aware someone was drawing in the same space. To address this issue, the system display a temporary message making other members aware that someone is already writing in a particular area (Fig. 3), warning other users to avoid drawing in the same space. Therefore, two or more users may still write in the same area, but they are aware that other participant's drawing may interfere, encouraging agreement among users before occur.



a. Gustavo's device b. Felipe's device

Fig. 3. Gustavo's writing (b) won't be shared until he raises the stylus pen. Meanwhile, Felipe's device shows the current received strokes and a warning saying Gustavo is still writing.

A user might need to correct or organize his or her writings. For this, methods to edit and manage writings and drawings are required. To allow this, we have included cut and move actions. When the user wants to remove something written from the screen, he or she needs to draw a cross gesture. This gesture will remove every stroke the cross touches, as shown in Fig. 4.a.

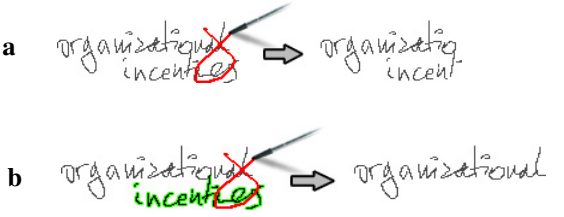


Fig. 4. a: A cross gesture removes touched strokes. b: A cross gesture removes selected elements only.

There are special situations where removing some elements might be difficult for the user (for example, removing a misspelling). For these cases we have included a second method for cutting: the user clicks one by one those lines he or her wishes to remove, selecting them; once all the strokes selected and ready to be removed, the user draws the same cross gesture, removing only those selected lines, leaving untouched the rest of the writing. An example of this process can is shown in Fig. 4.b.

Selection is an important tool to edit information in our system. We have quickly mentioned in the previous paragraph how to add single strokes to the selection. Clicking an already selected line deselects it. Also, clicking on an empty area clears the selection. In some cases, the user might want to select a complex group of strokes. We have included a method to do this with a single gesture, called double lasso gesture. This gesture consists in double surrounding an area with a certain closed shape. Every element included within the shape will be added to the selection, as illustrated in Fig. 5.

Fig. 5. A double lasso gesture selects enclosed items

Another way to manage the contents in our system is by moving them. The user may decide to join some terms, or to separate two concepts and write something in between. We have included the move action to allow this. After selecting some strokes, the user can hold and drag any of the selected lines in order to move all the selection.

Finally, our system allows the user to paste previously cut objects. This lets the user to undo the last removal, or to move objects between pages. In order to paste, the user must hold the stylus pressed in the clicked point for a certain time. The removed elements will re-appear in the clicked point.

5.2 Conceptual Map Building and Navigation

As mentioned in section 2, our system allows to hierarchically organize the content of a meeting document in pages in order to solve the handheld limited screen size problem. These pages are associated to parent nodes, which is usually labelled. In order to create a new page, the user must write or draw some strokes, as a title. Then, he or she has to surround them with a rectangle or a partial rectangle. The system will recognize this gesture as a node creation (Fig. 6). Automatically, a new page will be associated to this new node.

Fig. 6. Surrounding with a partial rectangle gesture creates a new node with the contained strokes as label

Once a node has been created, the user can enter into the associated page double clicking on the node. After this, the handheld screen will display the clicked node's content page, as shown in figure 2. In order to exit the page associated with the node and go back to the parent page, the user can double click in any empty area.

On every page, the user can use the same writing and drawing methods, as well as every gesture, including those for creating new nodes. In this way, the user builds an implicit conceptual map of the information being represented, through hierarchical contained pages.

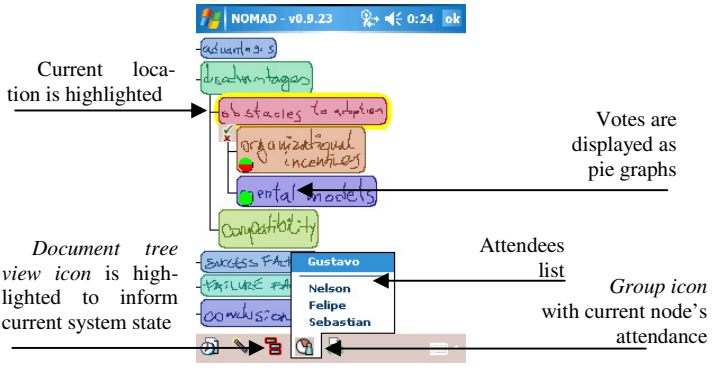


Fig. 7. The document tree view shows a scaled summary of the document, including user's current location

The user might get confused about his or her current location in the conceptual map if the document has too many nodes. To help the user get oriented within the document, we have included a *document tree view* button. This view shows an organized version of all the nodes in the document, sorted hierarchically. The node where the user is currently located shows a yellow highlight. Clicking on any node will lead the user to its content page, just like double clicking its label. This makes the hierarchical

view of the document also a powerful navigation tool. An example of the *document tree view* is shown in Fig. 7.

By default, the document tree is scaled down in order to display all the nodes at the same time. When the tree is big enough, some downscaled nodes could turn difficult to read or understand. Therefore, a method for zooming in and scrolling through the tree has been added to our system. In the document tree view, the user can hold the stylus down over an empty area, and then drag the pen right or left to zoom in and zoom out respectively, and up and down to scroll. In this way, with a single gesture, the user can zoom and scroll through an entire complex document tree. See Fig. 8 for a diagram of this two-in-one innovative gesture's functionality.

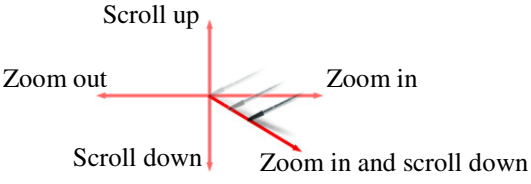


Fig. 8. Dragging the stylus on the document tree view allows users to zoom and scroll with a single gesture

5.3 Collaborative Process

During a group meeting, the user needs to get some information about other attendees. The system informs the user when other users enter or leave the session, but this is not enough to follow current attendees location. Our system provides two methods to retrieve some information about the participants group: the group icon displays user's current node attendance percentage (how many of the total participants are looking the same screen the user is), and the attendees' list (Fig. 7). The group icon warns the user when he or she is alone, with no other user's attention, becoming red. The more users share the same screen, the greener the icon will turn. This allows the user to quickly know how many other participants are sharing the writing space.

The system allows users to vote for or against nodes. Thus, at a certain point of the meeting, attendees could agree to vote about certain ideas. In order to vote, the user needs to click the *mode icon* from the menu bar (Fig. 2). Votes are represented as a pie graph at the lower left corner of the node, as shown in Fig. 7. The green portion of the pie represents positive votes, and the red portion represents negative votes. The black portion represents users who have not voted.

Once in the voting mode, the user can vote for a node by drawing a tick mark, starting on the node (Fig. 9.a: in the example, the pie shows 25 percent of positive votes: 1 of 4 attendees has voted). In order to vote against an element, the user draws the same *cross gesture* used for cutting (Fig. 9.b). A user can change his or her vote by replacing the old one (drawing a cross will replace a previous tick and vice-versa).

The full session tree can be represented as an XML document. During a meeting, each user can freely decide to save the current state of the document into an XML file, in his or her handheld's memory. Later, a new meeting could be started from this file, in order to continue the same reunion based on the previous work. A user who saved such files could revise a previous meeting to remember discussed subjects and conclusions.

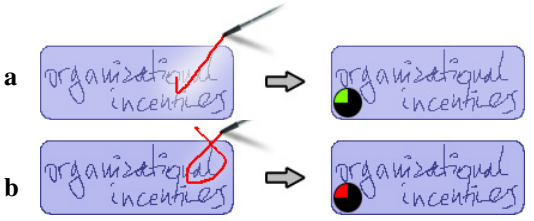


Fig. 9. a: Drawing a tick gesture under vote mode adds a positive (green) vote to the node. b: Drawing a cross gesture under vote mode adds a negative (red) vote to the node.

During a meeting the user might want to take some private notes, without sharing them with the rest of the participants. In order to allow this, our system counts with the *private note menu* (Fig. 2), which lets the user write private notes on a blank page associated to the user's current location, or over the current page. Later, the user can share this note with the group by choosing "Publish note" option, from the same menu. In the menu, the user can find his or her previous privates and published notes; as well as other users' published notes.

There are certain situations when meeting attendees could have previously worked by their own, developing new ideas or individually building their part of the meeting, based on a previous meeting, or on a brand new document. Later, during a meeting, the user decides to share his offline work with the other users. The system has two special options for this scenario: one where the work needs to be merged with the current session, called *merge versions*, and other where the user want to show as a separate version the offline work, called *open as version*. *Merge versions* merges the saved document with the current one. All new elements (writing and nodes) will be added to the current session, and repeated elements will be updated to the saved document version. On the other hand, *open as version* creates a new node in the document root page, using the file's name as the node label. Inside this node, as its content page, the saved document's root page is copied. All these changes, as described in next section, are shared and updated on everyone's handheld.

5.4 System Achitecture

Gesture matching and action triggering. Every concept of the system is considered a gesture object. Some of these elements live temporary, executing an action, and some persist and are represented graphically, called *drawbleGestures*. Some gestures are used in order to create another gesture. For example, a *cross gesture* mutates into a *cut gesture* under work mode; while under *vote mode* it becomes a *voteAgainst* gesture (see Fig. 10).

Every gesture class has a analyze method, which receives the analyzed figure (the last user pen-based input) as a parameter, and tries to check if this stroke follows the internal rules to be considered such gesture. After a gesture class accepts the drawn shape as its own, the class triggers its internal actions (like adding the clicked shape to selection, creating a new node, or cutting some writing) and returns a TRUE value, stopping the gesture matching chain. In case the class refuses the given shape (returning a FALSE), the system will assume the figure does not belong to the queried

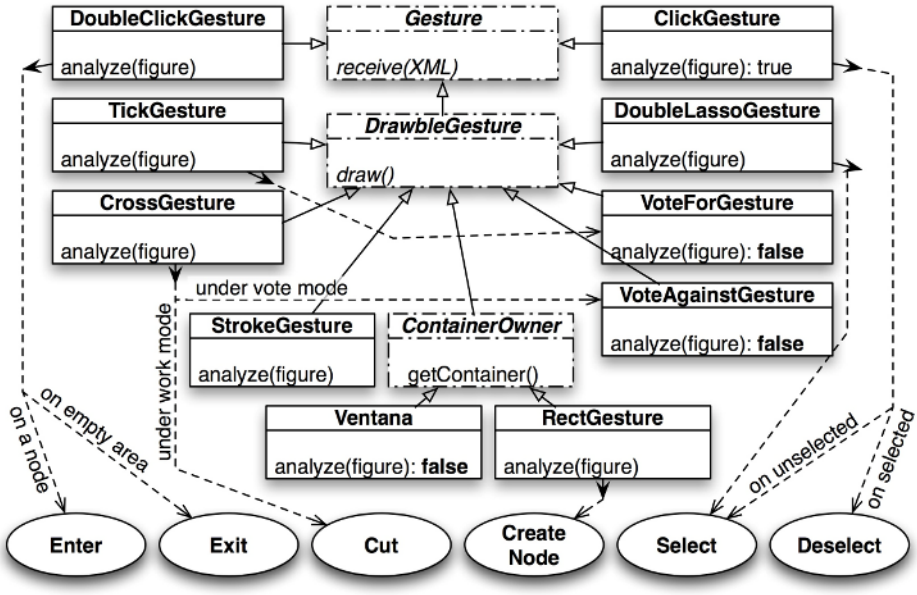


Fig. 10. Simplified class hierarchy with action triggering. Some gestures actions depend on the context.

family. As mentioned, every class does this process using rules matching, instead of pattern matching or neural networks, because rules are easy to be described without need to teach the system, and they can analyze an indeterminate number of conditions or parameters: for example, the double lasso figure cannot be limited to a certain amount of points.

To determine what action to follow, the system starts checking the gesture against the highest priority class and, in case it fails, continue this matching with the next class in the priority chain. Once a gesture has replied TRUE, the system stops checking the gesture. In case all other classes return FALSE, the last gesture (the click gesture) always returns TRUE.

Content structure. There are some special *drawbleGestures* called *containerOwner*, which own two instances of another type of element called *gesturesContainer*: one for its content page, and a second one for its label. These *gesturesContainers* can storage multiple nested gestures. *ContainerOwners* are referred in this paper as nodes, because their nature allows building hierarchical structures. The document root is a *containerOwner* itself. Another *containerOwner* subclass is the *rectGesture*, motioned in section 4.2: for *rectGestures* it is more clear the use of both *gesturesContainers*, because they clearly have some writing associated as a label and a whole set of nested elements as their content page, viewable with a *double click* (enter action).

Some special gestures exploit this hierarchical structure: for example, vote gestures are abstract objects contained in the contents *gestureContainer* of the voted node. Thus, in order to draw the vote pie, the system looks into the node content for vote gestures (*voteFor* and *voteAgainst gestures*), summarizing them in the round graph.

Communication Protocol. As soon as each user launches the system, it searches the local network for other participants: our system works over local area networks (wired and wireless), as well as for ad-hoc peer to peer wireless networks. Each time another peer is discovered, it is added to the known participants list. The user gets informed every time another user connects or disconnects from the current session.

When a new user is discovered, the system will send him the current state of the meeting: A complete version of the session tree will be transfer as an XML document. Also, every time the user modifies the document tree, by writing, adding or removing elements, or any other change, a short XML message describing the changes is sent to every known participant, ensuring the data consistency between peers.

On the other side, users will receive XML information: a complete document when joining a session or small XML chunks when receiving changes. Both messages are analyzed the same way: each XML entity is used as a parameter for a gesture interpreter, exactly as if it would have been drawn.

6 First Usability Testing and Analysis

We made a first usability analysis of the system using think-aloud and heuristic evaluation methods in order to detect and correct possible design or programming failures of the prototype developed. The evaluation was aimed to the detection of problems in: a) the collaboration supported by the system, identifying the circumstances where the system does not provide he adequate support for or hinders the collaborative work among the members; b) the user interface, identifying which aspects difficult the understanding and learning the way it works, the user satisfaction when using the diverse functionalities, particularly, when sketching, using gestures, and navigating; and c) combinations of both. We classified the possible problems we may found according to their severity, as suggested by [12], [14], [28] and [22]: a) Critical problems include crushes, breaks of the working flow, information lose, lose of the focus of the work; b) serious problems include lost of functionality, difficulties but not break of the working flow; and c) cosmetic problems include orthographic errors, minor visual problems which not affect the function of the system.

In the Think-aloud evaluation method users verbalize their thoughts about the system while using it, thus avoiding distortions, wrong cognitive translations or omissions that may be introduced by a guided questionnaire [4]. This type of evaluation is more user-oriented than the heuristic evaluation, where experts play an important role, [4], [13], and [14]. The process is guarded by a monitor and observed by examiners, who are in charge of gathering the oral information in real time or by analysing the video recording of the session. According to [12], this method can be used to evaluate the collaboration inside a working group supported by a computer system. En each session, the participants acted just as they were holding a real meeting and verbalizing their thoughts at the same time. The monitor has to first instruct the participants about the usage of the system explaining the functionalities. During the meeting the monitor mainly observes its development and the way the participants use the system. Only if the participants do not speak any word for 30 seconds the monitor asks the participants to start thinking aloud again, [4]. The examiners observe the development of the

meeting registering in real time the usability problems detected and verbalized by the participants. After finalizing the meeting, they are in charge of analysing the problems encountered. Two think-aloud test meetings were organized with 4 to 5 people each. In one of the sessions all participants were experts in the use and development of collaborative systems. The participants of the other two meetings were people who frequently use computers and often participate in meetings at work. The same goal and specific tasks were applied for both test meetings. The discussion subject was to identify 3 benefits and 3 drawbacks of face-to-face meetings. We also asked them to perform specific tasks during the meeting. The participants have to a) take all necessary notes they think will help to the development of the meeting; b) store in a file the notes taken during the meeting. Additionally, they were instructed to use the voting mechanism if there is any need to take decisions. During 5 to 10 minutes before starting the test meeting the monitor instructs the people how to use the system. Each session lasted for 35 to 45 minutes, and was registered by video-recorder. During the meeting, the monitor and the examiners took notes about usability problems detected and verbally expressed by the participants. Once the meeting was over, the monitor and the examiners discussed for about one hour the outcome of the test meeting according to the IDA method described in [13].

We also conducted a heuristic evaluation (HE) of the software. The heuristic evaluation is very popular because of its low cost and low time consumption. It is also

Table 1. Compilation and classification of the problems encountered and their solutions

Problem	SV	TP		Solution
Participants do not know who was the author of a certain action	CE	CL	SZ	Use different colour for user
Unwanted nodes are created *	CE	IU	GZ	Tuning gesture recognition
Confusion in the collaborative writing*	CR	IU	SZ	Use mechanism of collaborative writing awareness (Figure 3)
Free-hand writing is not accurate or swift. *	CR	IU	TR	Modify the graphic refreshment procedure for nodes
Confusion in the synchronization of the work, people add or delete to many things at the same time +	SE	CL	SZ	Provide flexibility for private notes, provide floor control policy options
Unwanted explosion (getting inside) of a node *	CR	UI	GZ	Tuning gesture recognition
Writing becomes slow when there are more strokes on the screen *	SE	UI	TR	Optimize synchronizing process
Slow distribution of the new information *	SE	CL	SZ	Optimize synchronizing process
Same strokes are not distributed*	SE	CL	TR	Optimize synchronizing process
Some nodes are not distributed *	SE	CL	GT	Optimize synchronizing process
It is difficult to tell if the current node corresponds to a private node or not+	CE	UI	ND	Provide a more evident awareness icon
It is difficult to know which node is currently being explored +	CE	UI	ND	Label the pages with the label of the corresponding node. (Figure 2b)

With the think-aloud test many critical and not so critical problems were found (those marked with *). These were solved before the HE testing, where those problems marked with + arose.

proved to be a very assertive one [12]. According to [12], HE is very good at detecting usability problems in mobile devices and cosmetic problems, so we applied it after the improvements and with experts as users. The results just confirmed those of the Think-Aloud evaluation.

The table 1 compiles the most relevant information obtained during the test meetings. Each one of the problems encountered was classified according to first its degree of severity (SV): critical (CR), serious (SE), or cosmetic (CE). Then according to the type of problems (TP): collaboration (CL) or user interface (UI). And within the type of problem, related to which aspect of the system it was encountered: synchronization (SZ), gestures (GZ), strokes (TR), or nodes (ND).

7 Conclusions

After correcting the problems found in the tests and taking into account the opinion of the people involved we can conclude that we designed and developed an electronic meeting support system for handhelds which is simple, easy to use, and supports effectively the development of a discussion, at least in the form we asked them to hold the meeting, which we think is general enough for validating a tool in a first stance. Of course, a deeper and wider study which does not only involve "laboratory conditions" like the one described in the previous chapter, but in "real meeting situations" should be carried out in order to validate this tool as a successful meeting support system in a concluding manner.

With the developing and first testing of this system we found evidence that the two most important design principles applied do really work. In fact, we can assert with a certain degree of confidence that developing a system for handhelds based only in gestures and handwriting input, avoiding the "traditional" widgets is not only feasible bud very good accepted by the user. We can also assert that the structuring of the document content as a concept map gives the necessary flexibility to manage the different contributions of the users. The idea of structuring a collaborative authored document as a concept map was already presented in [24] already in the year 1994, but the novelty of this work is to find a way to effectively manage such a structure with just gestures and in a reduced screen, allowing an easy navigation through the document. We believe that the contributions of the work reported here can be also applied in to various kinds of systems for handhelds with goals and aims for supporting different work as the one targeted here.

The tests have shown that after fixing the initial critical and serious problems detected the system has a good acceptance. An interesting issue about the testing is that they have been conducted in different cultural environment (Latin American and Asian). The absence of labeled widgets and text contributed to the fact that the cultural background is not and obstacle to use the system. In fact the Asian people took notes more easily using Chinese characters than the Latin American using Latin alphabet. The collaborative meeting support of the system evaluated based on gesture and sketch enhances the design in a natural and concordant way, enabling the share

and exchange of information. The user interface designed is away from the rigidity of traditional user interfaces, supporting instead the flexibility and ambiguity inherent in a natural mode of communication. Therefore, member's meeting can put their focus on specific meeting tasks, instead of things in low-level.

Finally, we think that the proposed system can also be used to support undergraduate/postgraduate students develop skills required during a meeting. In this scenario teachers may use the system for presenting ideas to the students and involve them in common meeting activities like taking decisions, solving problems collaboratively, etc. The students will be able to learn the diverse stages and processes of a meeting by doing them. This will allow a simulation of the real conditions of a meeting; therefore. it will produce positive motivation and participation levels in the students [31]. According to , the anonymity supported by handhelds helps reducing evaluation apprehension by allowing group members to submit their ideas without having to speak them in front of the group. This, and the fact that people can "express" their ideas concurrently, not having to wait for a slot, encourages the contribution from people who normally would play a more passive role in a discussion meeting. The system also helps members to follow the agenda of the meeting thus reducing coordination problems. Furthermore, the system can also support a wider spectrum of activities, including the generation and organization of ideas, group analysis, decision making, group writing and action planning.

Acknowledgements. This paper was partially supported by Fondecyt 1050601, and the DI - Universidad de Chile Nro. I2 04/01-2.

References

1. Antunes, P., and Costa, C. Handheld CSCW in the Meeting Environment. Proceedings of the CRIWG'02, LNCS 2440 (2002) 47-60.
2. Bates, S. E. What is IS' Role in Re-engineering? Business Leaders Should Lead. Computerworld 29(39) (1995) 134.
3. Bergqvist, J., Dahlberg, P., Kristoffersen, S., and Ljungberg, F. Moving out of the meeting room: exploring support for mobile meetings. Proceedings of the Sixth European conference on Computer supported cooperative work. Copenghagen, Denmark (1999) 81-98.
4. Boren, T., and Ramey, J. Thinking Aloud: Reconciling Theory and Practice. IEEE Transactions on Professional Communication, 30(3) (2003) 261-278.
5. Costa, C., Antunes, P., Dias, J. The Meeting Report Process: Bridging EMS with PDA. Third International Conference on Enterprise Information Systems, ICEIS 2001. Setubal, Portugal (2001) 821-826.
6. Dai, G. and Wang, H. Physical Object Icons Buttons Gesture (PIBG): A new Interaction Paradigm with Pen. Proceedings of CSCWD 2004, LNCS 3168, 11-20, 2005.
7. dos Santos, A. Vasconcelos, S., Fagundes, L. Kayo, R., Zanfolim, T., and Brunetto, C. SACE-CSCW: A Synchronous Asynchronous Common Environment for Computer Supported Cooperative. Work to Aid Concurrent Engineering Processes. SCCC (1997) 218-226.
8. Haag, S., M. Cummings, and Dawkins, J. Management Information Systems for the Information Age, 2nd Ed., Boston, MA: Irwin McGraw-Hill (2000).

9. Hayne, S. The facilitators' perspective on meetings and implications for group support System. Database, 30(4) (1999) 72-91.
10. Jessup, L., and Valacich, J. Group Support Systems: A New Frontier. New York: MacMillan (1993).
11. Kjeldskov J., and Skov M. B. Evaluating the Usability of a Mobile Collaborative System: Exploring Two Different Laboratory Approaches. Proceedings of the 4th International Symposium on Collaborative Technologies and Systems, Orlando, Florida (2003) 134-141.
12. Kjeldskov J., Skov M.B., and Stage J. Instant Data Analysis: Evaluating Usability in a Day. Proceedings of NordiCHI, Tampere, Finland. ACM (2004) 233-240.
13. Kjeldskov J., Skov M.B., and Stage J. Does Time Heal? A Longitudinal Study of Usability. Proceedings of OzCHI 2005, Canberra, Australia, ACM (2005).
14. Landay, J. and Myers, b. Sketching interfaces: Toward more human interface design, IEEE Computer, 2001 34(3) (2001) 56-64.
15. Luff, P., Heath, C. Mobility in Collaboration. Proceedings of Computer Supported Collaborative Work, CSCW'98. ACM Press, 1998, 305-314.
16. Marshall, C., Ruotolo, C. Reading-in-the-Small: A Study of Reading on Small Form Factor Devices. Proceedings of the JCDL'02, Portland, Oregon, USA, 2002, 13-17.
17. Myers, B.A., Stiel, H., and Gargiulo, R. Collaboration using multiple PDAs connected to a PC. Proceedings of the ACM, Conference on Computer Supported Cooperative Work. Seattle, WA (1998) 285-294.
18. Nunamaker, J. F. Future research in group support systems: needs, some questions and possible directions. International Journal of Human-Computer Studies, 47, 1997, p 357-385.
19. Perry, M. O'hara, K., Sellen, A., Brown, B., and Harper, R. Dealing with mobility: understanding access anytime, anywhere. ACM Transactions on Computer-Human Interaction-TOCHI, 4(8) (2001) 323-347.
20. Po, S., Howard, S., Vetere, F., and Skov, M. B. Heuristic Evaluation and Mobile Usability. Bridging the Realism Gap. In Proceedings of the 6th International Conference on Human Computer Interaction with Mobile Devices and Services, (Mobile HCI 2004), Springer-Verlag, LNCS 3160 (2004) 49-60.
21. Schmidt, A. Lauff, M., and Beigl, M. Handheld CSCW. Workshop on Handheld. Proceedings of the Computer Supported Collaborative Work - CSCW '98, November, Seattle (1998).
22. Streitz, N.A., Geißler, J., Haake, J.M., Hol, J. DOLPHIN: integrated meeting support across local and remote desktop environments and LiveBoards, Proceedings of the 1994 ACM conference on Computer supported cooperative work, October 22-26, Chapel Hill, North Carolina, United States (1994) 345-358,
23. Tropman, J. E. Effective meetings: Improving Group Decision Making. Sage Publications (1996).
24. van der Lugt, R. Functions of sketching in design idea generation meetings. In TT Hewett & T Kavanagh (Eds.), Creativity & cognition, New York: ACM, 2002, 2002, 72-79.
25. Whitwort, B. and McQueen, R. Voting before discussing: Computer voting as social communication. Proceedings of the 32nd Hawaii International Conference on Systems Sciences (1999) 1020-1031.
26. Wiberg, M. RoamWare: an integrated architecture for seamless interaction in between mobile meetings. Proceedings of the 2001 International ACM SIGGROUP - Conference on Supporting Group Work (2001) 288-297.
27. Wilson, C. Defining, Categorizing, and Prioritizing Usability Problems. UPA 2003 Idea Market, July, 2003. Available in http://www.upassoc.org/conferences_and_ events/ upa_conference/2004/call/sample/ideamarket_usability%20problems_wilson.doc

28. Wolf, C., and Rhyne, J. Communication and Information Retrieval with a Pen-Based Meeting Support tool. Proceedings of CSCW - ACM (1992) 322-329.
29. Burleson, W. Developing creativity, motivation, and self-actualization with learning systems International Journal of Human-Computer Studies 63(4-5) (2005) 436-451.
30. Tyran, G., Sherpherd, M. Collaborative Technology in the Classrom: A Review of the GSS Research and a Researh Framework. Information Technology and Management 2 (2001) 395-418.

ODBASE 2006 International Conference (Ontologies, DataBases, and Applications of Semantics) PC Co-chairs' Message

Welcome to the Fifth International Conference on Ontologies, Databases, and Applications of Semantics (ODBASE 2006). This year's ODBASE conference was held in Montpellier, France from October 29 to November 3, 2006.

The ODBASE conferences provide a forum for exchanging the latest research results on ontologies, data semantics, and other areas of computing related to the Semantic Web. We encourage participation of both researchers and practitioners in order to facilitate exchange of ideas and results on semantic issues in Web information systems. Towards this goal, we accepted both research and experience papers.

This high-quality program would not have been possible without the authors who chose ODBASE as a venue for their publications. Out of 82 submitted papers, we selected 19 full papers, 9 short papers, and 7 posters. To round up this excellent program, Marie-Christine Rousset agreed to be our keynote speaker.

We are grateful for the dedicated work of the 39 top experts in the field who served on the Program Committee. Special thanks goes to the external referees who volunteered their time to provide additional reviews. Finally, we are indebted to Kwong Yuen Lai, who was immensely helpful in facilitating the review process and making sure that everything stayed on track.

August 2006

Maurizio Lenzerini, Universitá di Roma "La Sapienza", Italy
Erich Neuhold, Darmstadt University of Technology, Germany
V.S. Subrahmanian, University of Maryland College Park, USA

R. Meersman, Z. Tari et al. (Eds.): OTM 2006, LNCS 4275, p. 697, 2006.

SomeWhere: A Scalable Peer-to-Peer Infrastructure for Querying Distributed Ontologies

M.-C. Rousset[2], P. Adjiman[1], P. Chatalic, F. Goasdoué, and L. Simon[1]

[1] LRI, bâtiment 490, Université Paris-Sud 11, 91405 Orsay Cedex, France
[2] LSR-IMAG, BP 72, 38402 St Martin d'Heres Cedex, France

Abstract. In this invited talk, we present the SomeWhere approach and infrastructure for building semantic peer-to-peer data management systems based on simple personalized ontologies distributed at a large scale. Somewhere is based on a simple class-based data model in which the data is a set of resource identifiers (e.g., URIs), the schemas are (simple) definitions of classes possibly constrained by inclusion, disjunction or equivalence statements, and mappings are inclusion, disjunction or equivalence statements between classes of different peer ontologies. In this setting, query answering over peers can be done by distributed query rewriting, which can be equivalently reduced to distributed consequence finding in propositional logic. It is done by using the message-passing distributed algorithm that we have implemented for consequence finding of a clause w.r.t a set of distributed propositional theories. We summarize its main properties (soundness, completeness and termination), and we report experiments showing that it already scales up to a thousand of peers. Finally, we mention ongoing work on extending the current data model to RDF(S) and on handling possible inconsistencies between the ontologies of different peers.

1 Overview of SomeWhere

SomeWhere promotes a "small is beautiful" vision of the Semantic Web [1] based on simple personalized ontologies (e.g., taxonomies of atomic classes) but which are distributed at a large scale. In this vision of the Semantic Web introduced by [2], no user imposes to others his own ontology but logical mappings between ontologies make possible the creation of a web of people in which personalized semantic marking up of data cohabits nicely with a collaborative exchange of data. In this view, the Web is a huge peer-to-peer data management system based on simple distributed ontologies and mappings.

For scalability purpose, we have chosen a simple class-based data model in which the data is a set of resource identifiers (e.g., URIs), the ontologies are (simple) definitions of classes possibly constrained by inclusion, disjunction or equivalence statements, and mappings are inclusion, disjunction or equivalence statements between classes of different peer ontologies. That data model is in accordance with the W3C recommendations since it is captured by the

R. Meersman, Z. Tari et al. (Eds.): OTM 2006, LNCS 4275, pp. 698–703, 2006.

propositional fragment of the OWL ontology language (http://www.w3.org/TR/owl-semantics).

Query answering through ontologies is achieved using a rewrite and evaluate strategy. In SomeWhere the query rewriting problem can be reduced to a consequence finding problem in distributed propositional theories. It is performed by a message-passing algorithm named DeCA: Decentralized Consequence finding Algorithm [3]. As a result, query answering in SomeWhere is sound, complete and terminates. Moreover, the detailed experiments reported in [4] show that it scales up to 1000 peers.

2 Illustrative Example

We illustrate the SomeWhere data model on a small example of four peers modeling four persons Ann, Bob, Chris and Dora, each of them bookmarking URLs about restaurants they know or like, according to their own taxonomy for categorizing restaurants.

Ann, who is working as a restaurant critic, organizes its restaurant URLs according to the following classes:

- the class $Ann{:}G$ of restaurants considered as offering a "good" cooking, among which she distinguishes the subclass $Ann{:}R$ of those which are rated: $Ann{:}R \sqsubseteq Ann{:}G$
- the class $Ann{:}R$ is the union of three disjoint classes $Ann{:}S1$, $Ann{:}S2$, $Ann{:}S3$ corresponding respectively to the restaurants rated with $1, 2$ or 3 stars:
  $Ann{:}R \equiv Ann{:}S1 \sqcup Ann{:}S2 \sqcup Ann{:}S3$
  $Ann{:}S1 \sqcap Ann{:}S2 \equiv \bot \quad Ann{:}S1 \sqcap Ann{:}S3 \equiv \bot$
  $Ann{:}S2 \sqcap Ann{:}S3 \equiv \bot$
- the classes $Ann{:}I$ and $Ann{:}O$, respectively corresponding to Indian and Oriental restaurants
- the classes $Ann{:}C$, $Ann{:}T$ and $Ann{:}V$ which are subclasses of $Ann{:}O$ denoting Chinese, Taï and Vietnamese restaurants respectively: $Ann{:}C \sqsubseteq Ann{:}O$, $Ann{:}T \sqsubseteq Ann{:}O$, $Ann{:}V \sqsubseteq Ann{:}O$

Suppose that the data stored by Ann that she accepts to make available deals with restaurants of various specialties, and only with those rated with 2 stars among the rated restaurants. The extensional classes declared by Ann are then: $Ann{:}ViewS2 \sqsubseteq Ann{:}S2$, $Ann{:}ViewC \sqsubseteq Ann{:}C$, $Ann{:}ViewV \sqsubseteq Ann{:}V$, $Ann{:}ViewT \sqsubseteq Ann{:}T$, $Ann{:}ViewI \sqsubseteq Ann{:}I$

Bob, who is fond of Asian cooking and likes high quality, organizes his restaurant URLs according to the following classes:

- the class $Bob{:}A$ of Asian restaurants
- the class $Bob{:}Q$ of high quality restaurants that he knows

Suppose that he wants to make available every data that he has stored. The extensional classes that he declares are $Bob{:}ViewA$ and $Bob{:}ViewQ$ (as subclasses of $Bob{:}A$ and $Bob{:}Q$): $Bob{:}ViewA \sqsubseteq Bob{:}A$, $Bob{:}ViewQ \sqsubseteq Bob{:}Q$

Chris is more fond of fish restaurants but recently discovered some places serving a very nice cantonese cuisine. He organizes its data with respect to the following classes:

- the class $Chris{:}F$ of fish restaurants,
- the class $Chris{:}CA$ of Cantonese restaurants

Suppose that he declares the extensional classes $Chris{:}ViewF$ and $Chris{:}ViewCA$ as subclasses of $Chris{:}F$ and $Chris{:}CA$ respectively:
$Chris{:}ViewF \sqsubseteq Chris{:}F$, $Chris{:}ViewCA \sqsubseteq Chris{:}CA$

Dora organizes her restaurants URLs around the class $Dora{:}DP$ of her preferred restaurants, among which she distinguishes the subclass $Dora{:}P$ of pizzerias and the subclass $Dora{:}SF$ of seafood restaurants.

Suppose that the only URLs that she stores concerns pizzerias: the only extensional class that she has to declare is $Dora{:}ViewP$ as a subclass of $Dora{:}P$: $Dora{:}ViewP \sqsubseteq Dora{:}P$

Ann, **Bob**, **Chris** and **Dora** express what they know about each other using mappings stating properties of class inclusion or equivalence.

Ann is very confident in Bob's taste and agrees to include Bob' selection as good restaurants by stating $Bob{:}Q \sqsubseteq Ann{:}G$. Finally, she thinks that Bob's Asian restaurants encompass her Oriental restaurant concept: $Ann{:}O \sqsubseteq Bob{:}A$

Bob knows that what he calls Asian cooking corresponds exactly to what Ann classifies as Oriental cooking. This may be expressed using the equivalence statement : $Bob{:}A \equiv Ann{:}O$ (note the difference of perception of Bob and Ann regarding the mappings between $Bob{:}A$ and $Ann{:}O$)

Chris considers that what he calls fish specialties is a particular case of Dora seafood specialties: $Chris{:}F \sqsubseteq Dora{:}SF$

Dora counts on both Ann and Bob to obtain good Asian restaurants : $Bob{:}A \sqcap Ann{:}G \sqsubseteq Dora{:}DP$

Figure 1 describes the resulting overlay network. In order to alleviate the notations, we omit the local peer name prefix except for the mappings. Edges are labeled with the class identifiers that are shared through the mappings between peers.

3 Query Rewriting in SomeWhere Through Propositional Encoding

In SomeWhere, each user interrogates the peer-to-peer network through one peer of his choice, and uses the vocabulary of this peer to express his query. Therefore, queries are logical combinations of classes of a given peer ontology.

The corresponding answer sets are expressed in intention in terms of the combinations of extensional classes that are *rewritings* of the query. The point is that extensional classes of several distant peers can participate to the rewritings, and thus to the answer of a query posed to a given peer.

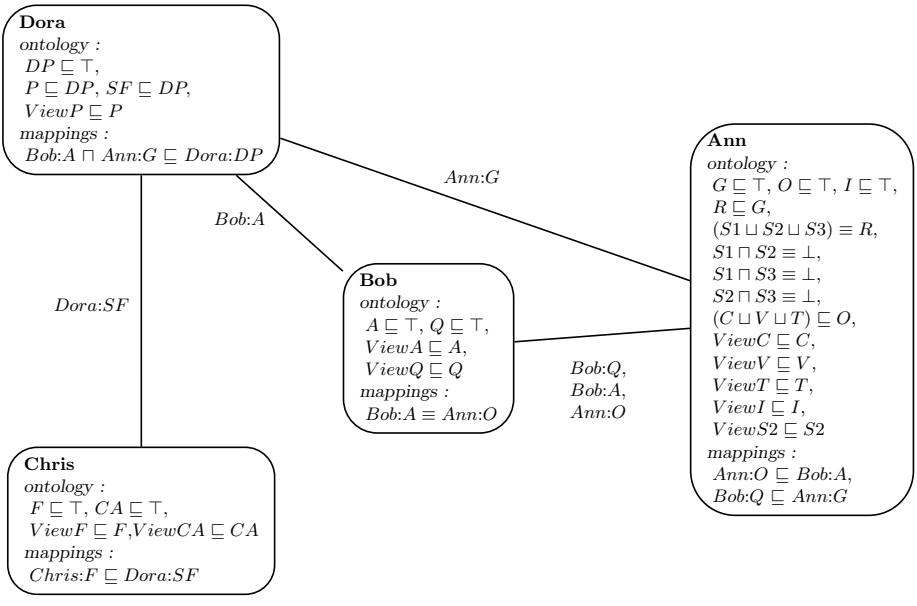


Fig. 1. The restaurants network

In general, finding all answers in a peer data management system is a critical issue [5]. In our setting however, we are in a case where all the answers can be obtained using *rewritings* of the query: it has been shown [6] that when a query has a *finite number of maximal conjunctive rewritings*, then *all* its answers (a.k.a. certain answers) can be obtained as the union of the answer sets of its rewritings.

In the SomeWhere setting, query rewriting can be equivalently reduced to distributed reasoning over logical propositional clausal theories by a straighforward propositional encoding of the query and of the distributed ontologies and mappings of a SomeWhere network. It consists in transforming the query and each ontology and mapping statement into a propositional formula using class identifiers as propositional variables.

The propositional encoding of a class description D, and thus of a query, is the propositional formula $Prop(D)$ obtained inductively as follows:

- $Prop(\top) = true$, $Prop(\bot) = false$
- $Prop(A) = A$, if A is an atomic class
- $Prop(D_1 \sqcap D_2) = Prop(D_1) \wedge Prop(D_2)$
- $Prop(D_1 \sqcup D_2) = Prop(D_1) \vee Prop(D_2)$
- $Prop(\neg D) = \neg(Prop(D))$

The global schema $\mathcal{S}$ of a SomeWhere peer-to-peer network is the union of the ontology and mapping statements distributed over the network. Its propositional encoding is the distributed propositional theory $Prop(\mathcal{S})$ made of the formulas obtained inductively from the statements in $\mathcal{S}$ as follows:

- $Prop(C \sqsubseteq D) = Prop(C) \Rightarrow Prop(D)$
- $Prop(C \equiv D) = Prop(C) \Leftrightarrow Prop(D)$
- $Prop(C \sqcap D \equiv \bot) = \neg Prop(C) \vee \neg Prop(D)$

We have shown in [3] that the maximal conjunctive rewritings of a query q in SomeWhere are the negation of the prime proper implicates of $\neg q$ w.r.t the propositional encoding of the schema. We use the distributed message-passing DeCA algorithm [3] to compute them.

4 Ongoing Work

We are involved in two extensions of SomeWhere. The first one concerns SomeRDFS which extends the very simple class-based data model of SomeWhere to RDFS. The second one deals with possible inconsistencies of the global schema of SomeWhere. Though the data model of SomeWhere is very simple, it allows negation. It is thus very likely that mappings between consistent ontologies cause inconsistencies at the global level. The problem is twofold: the inconsistencies have to be detected at join time ; they must then be handled at query time to compute only *well-founded* answers.

4.1 Extending the Data Model to RDFS

In SomeRDFS the ontologies and mappings are expressed in the core fragment of RDFS allowing to state (sub)classes, (sub)properties, typing of domain and range of properties. The mappings that we consider in SomeRDFS are inclusion statements between classes or properties of two distinct peers, or typing statements of a property of a given peer with a class of another peer. Therefore, mappings are RDFS statements involving vocabularies of different peers which thus establish semantic correspondances between peers.

We have shown that query answering in SomeRDFS can be achieved using a rewrite and evaluate strategy, and that the corresponding rewriting problem can be reduced to the same consequence finding problem in distributed propositional theories as in [3]. Moreover, the consequence finding problem resulting from the propositional encoding of the fragment of RDFS that we consider is tractable since the resulting propositional theories are reduced to clauses of length 2 for which the reasoning problem is in P. The experiments reported in [4] show that it takes in mean 0.07s to SomeWhere for a complete reasoning on randomly generated sets of clauses of length 2 distributed on 1000 peers.

4.2 Dealing with Inconsistencies

Since peer ontologies are personalized they model possibly different viewpoints. Therefore, the global schema made of the union of the local ontologies and the mappings may be inconsistent even if each local ontology is consistent. Given the lack of centralized control, all peers should be treated equally. It would be unfair to refuse the join of a new peer just because the resulting global schema

becomes inconsistent. Our choice is to accept the presence of inconsistency. The problem is first to *detect* inconsistencies, second to *reason* in spite of them in a satisfactory way. We thus compute only *well-founded* answers to a given query, i.e., answers that can be computed w.r.t. to a consistent subset of the global schema.

We assume each local ontology to be consistent. Therefore, the possible inconsistencies result from interactions between local ontologies and are caused by *mappings*. Before adding a mapping, a peer checks whether this mapping (possibly with other mappings) can be the cause of some inconsistency. In that case, the peer stores locally as a *nogood* the set of mappings involved in the corresponding inconsistency. At query time, the concerned distributed nogoods must be collected to check whether the answers under construction are well-founded: an answer is well-founded if the set of mappings used to infer it is not included in any nogood. This can be done using the distributed algorithms extending DeCA that are designed and studied in [7].

References

1. Rousset, M.C.: Small can be beautiful in the semantic web. In: ISWC. (2004)
2. Plu, M., Bellec, P., Agosto, L., van de Velde, W.: The web of people: A dual view on the WWW. In: WWW. (2003)
3. Adjiman, P., Chatalic, P., Goasdoué, F., Rousset, M.C., Simon, L.: Distributed reasoning in a P2P setting: Application to the semantic web. Journal of Artificial Intelligence Research (JAIR) (2006)
4. Adjiman, P., Chatalic, P., Goasdoué, F., Rousset, M.C., Simon, L.: Scalability study of P2P consequence finding. In: IJCAI. (2005)
5. Halevy, A., Ives, Z., Suciu, D., Tatarinov, I.: Schema mediation in peer data management systems. In: ICDE. (2003)
6. Goasdoué, F., Rousset, M.C.: Answering queries using views: a KRDB perspective for the semantic web. ACM Journal - Transactions on Internet Technology (TOIT) **4**(3) (2004)
7. Chatalic, P., Nguyen, G., M-C.Rousset: Reasoning with inconsistencies in propositional peer-to-peer inference systems. Proceedings of ECAI (2006)

Querying Ontology Based Database Using OntoQL (An Ontology Query Language)

Stéphane Jean, Yamine Aït-Ameur, and Guy Pierra

LISI-ENSMA and University of Poitiers
BP 40109, 86961 Futuroscope Cedex, France
{jean, yamine, pierra}@ensma.fr

Abstract. Nowadays, ontologies are used in several research domains by offering the means to describe and represent concepts of information sources. Therefore, several approaches and systems storing ontologies and their instances in the same repository (database) have been proposed. As a consequence, defining a query language to support ontology-based database (OBDB) becomes a challenge for the database community. In this paper, we present *OntoQL*, an ontology query language for OBDBs. Firstly, we present formally the OBDB data model supported by this language. Secondly, an overview of the algebra defining the semantics of operators used in *OntoQL* is described. Several query examples showing the interest of this language compared to traditional database query languages are given along this paper. Finally, we present a prototype of the implementation of *OntoQL*.

1 Introduction

It is well accepted to state that database models have been defined in order to store collections of data that are instances of given concepts defined by conceptual models. Usually, the developed conceptual models are specific to the designed database and the meanings, definitions, descriptions and identifications of these concepts are not formalized nor stored in the database. Moreover, the definition of these conceptual models and their translation to logical models have been well studied by a wide community of researchers. Formal data models and algebras have been proposed for managing such databases.

Nowadays, several data models recommending the use of domain ontologies to describe data and their semantics have been developed. These models enable an user or a system to retrieve the definition, meaning, translation and/or identifier of a given data item corresponding to a given data concept stored in an ontology. Therefore, the idea of storing the ontology as well as the data in a single database model emerged. We call Ontology Based Databases (OBDB) such a database model.

In parallel to the work consisting in equipping database models with ontologies led by the database community, the semantic web community made several persistence proposals to store the data described by instances of RDFS [1] or OWL [2] schemas. These approaches either define persistence models for RDF

R. Meersman, Z. Tari et al. (Eds.): OTM 2006, LNCS 4275, pp. 704–721, 2006.

triples or use specific storage approaches on these instances [3,4,5]. Languages exploiting such models have been proposed. Examples are RQL [6], OWL-QL [7] etc. They support several specific features like instance resonating, graph traversal etc. However, they do not preserve any database compatibility with database models. Therefore, migration of instances is necessary.

If several persistence data models and query languages have been proposed by artificial intelligence community for the semantic web, few work have been conducted and originated from a database-oriented perspective. In this paper, we present *OntoQL*, an exploitation language for an OBDB data model, named OntoDB designed by a layered approach on top of a relational database model. It has been proved useful for several database applications (e.g. semantic integration [8]) in several application domains (electronic, automotive, medical data).

This paper is structured as follows. Next section presents the formalization of the OBDB data model addressed in this paper. Section 3 presents the algebra designed for the *OntoQL* query language and section 4 discusses optimization techniques for this algebra. Section 5 introduces the *OntoQL* data definition and query languages by a set of examples showing the differences with traditional query languages. Section 6 discusses the *OntoQL* implementation and processing issues when implemented on top of an object-relational database (ORDBMS). Section 7 describes, on a toy example, how our approach runs. Section 8 discusses related work. Finally, section 9 concludes the paper by summarizing the main results and suggesting future work.

2 The OBDB Data Model

The OBDB database model is based on the definition of two main related parts: *ontology* and *content*. Instances are stored in the content part while ontologies are stored in the ontology part.

2.1 Ontology

The ontology part allows to store ontologies as instances of an ontology model. It is formally defined by a 7-Tuple as $<$ `E`, `OC`, `A`, `SuperEntities`, `TypeOf`, `AttDomain`, `AttRange`, `Val` $>$, where:

- `E` is a set of entities representing the ontology model. It provides with a global super entity `Concept`, the predefined entities `C` and `P` described below and user-defined entities.
- `OC` is the set of concepts of ontologies (classes, properties ...) available in the database or that can be constructed by a query. All the concepts of ontologies have an unique identifier.
- `A` is the set of attributes describing each ontology concept.

- $\texttt{SuperEntities} : \texttt{E} \rightarrow 2^{\texttt{E}}$[1] is a partial function associating a set of super entities to an entity. It defines a lattice of entities. Its semantics is inheritance and it ensures substitutability.
- $\texttt{TypeOf} : \texttt{OC} \rightarrow \texttt{E}$ associates to each concept of an ontology the lower (strongest) entity in the hierarchy it belongs to.
- $\texttt{AttributeDomain}, \texttt{AttributeRange} : \texttt{A} \rightarrow \texttt{E}$ define respectively the domain and the range of each attribute.
- $\texttt{Val} : \texttt{OC} \times \texttt{A} \rightarrow \texttt{OC}$ gives the value of an attribute of an ontology concept.

The OBDB data model provides with atomic types (`Int`, `String`, `Boolean`) and with two parameterized types `Set`[`T`] and `Tuple`. `Set`[`T`] denotes a type for collections of elements of type `T` and $\{\texttt{o}_1, \ldots, \texttt{o}_n\}$ is an object of this type (the $\texttt{o}_i$'s are objects of type T). The $\texttt{Tuple}[< (\texttt{A}_1, \texttt{T}_1), \ldots, (\texttt{A}_n, \texttt{T}_n) >]$ parameterized type creates relationships between objects. It is constructed by providing a set of attribute names ($\texttt{A}_i$) and attribute types ($\texttt{T}_i$). $\texttt{Tuple}[< (\texttt{A}_1, \texttt{T}_1), \ldots, (\texttt{A}_n, \texttt{T}_n) >]$ denotes a type tuple constructed using the $\texttt{A}_i$ attribute names and $\texttt{T}_i$ attribute types. $< \texttt{A}_1 : \texttt{o}_1, \ldots, \texttt{A}_n : \texttt{o}_n >$ is an object of this type (the $\texttt{o}_i$'s are objects of type $\texttt{T}_i$). The `Tuple` type is equipped with the $\texttt{Get\_A}_i\texttt{\_value}$ functions to retrieve the value of a `Tuple` object `o` for the attribute $\texttt{A}_i$. The application of this function may be abbreviated using the dot-notation ($\texttt{o.A}_i$)

`E` provides the predefined entities `C` and `P`. Instances of `C` and `P` are respectively the classes and properties of the ontologies. Entity `C` defines the attribute $\texttt{SuperClasses} : \texttt{C} \rightarrow \texttt{SET}[\texttt{C}]$ and entity `P` defines the attributes $\texttt{PropDomain} : \texttt{P} \rightarrow \texttt{C}$ and $\texttt{PropRange} : \texttt{P} \rightarrow \texttt{C}$. The description of these attributes is similar to the definitions given for `SuperEntities`, `AttributeDomain` and `AttributeRange` replacing entities by classes and attributes by properties. Moreover, a global super class `Root` is predefined.

Finally, an ontology gives a precise definition of concepts with more attributes (comment, version, multi-lingual definition, synonymous names, ...) to describe classes and properties of ontologies. These predefined entities and attributes constitute the kernel of the ontology models we have considered. User-defined entities (illustration, document, ...) and attributes (note, remark, ...) may be added to this kernel in order to take into account other specific ontology models.

Notice that the ontologies stored in this part are defined with the different characteristics shared by the standard ontology models PLIB[9] and OWL[2]. The *multi-instantiation* and *property subsumption* (subproperty) specific features of OWL are not taken into account in this formalization. They have been removed to get an optimal database representation (OBDB Light). However, a full version of the OBDB model with these features is under development (OBDB Full). Nevertheless, for several examples and significant applications[2,3] we have developed, this limitation had neither effect nor impact on the expressive power of the treated problems. [10] gives our precise view of ontologies with respect to their exploitation in a database context.

[1] We use the symbol $2^{\texttt{E}}$ to denote the power set of E.
[2] See http://www.plib.ensma.fr for references to IEC/ISO standards ontologies.
[3] For example, the EPICEM project (http://www.episem-action.org)

2.2 Content

The content part stores instances of ontology classes. It is formalized by a 5-tuple $< \texttt{EXTENT}, \texttt{I}, \texttt{TypeOf}, \texttt{SchemaProp}, \texttt{Val} >$ where:

- `EXTENT` is a set of *extensional definitions* of ontology classes.
- `I` is the set of instances of the OBDB. Each instance has an identity.
- $\texttt{TypeOf} : \texttt{I} \rightarrow \texttt{EXTENT}$ associates to each instance the extensional definition of the class it belongs to (collection of its instances).
- $\texttt{SchemaProp} : \texttt{EXTENT} \rightarrow 2^{\texttt{P}}$ gives the properties used to describe the instances of an extent (the set of properties valued for its instances).
- $\texttt{Val} : \texttt{I} \times \texttt{P} \rightarrow \texttt{I}$ gives the value of a property occurring in a given instance. This property must be used in the extensional definition of the class the instance belongs to.

2.3 Relationship Between Each Part

The relationship between ontology and its instances (content) is defined by the partial function $\texttt{Nomination} : \texttt{C} \rightarrow \texttt{EXTENT}$. It associates a definition by intension with a definition by extension of a class. Classes without extensional definition are said to be *abstract*. The set of properties used in an extensional definition of a class must be a subset of the properties defined in the intensional definition of a class ($\texttt{propDomain}^{-1}(\texttt{c}) \supseteq \texttt{SchemaProp}(\texttt{nomination}(\texttt{c}))$).

2.4 Related Work

Storing ontologies and their instances in databases has been the subject of several research studies and proposals. In the context of the semantic web, several OBDB models [3,4,5,11] have been proposed to manage data described by ontologies represented in the standard ontology models RDFS [1] or OWL [2]. In these approaches, an instance, often called an individual, has its own property and class structure. Therefore, to manage instances, a generic database schema, not meaningful to an user and not customizable, is used. The simplest and more general one uses an unique table of triples [11] for storing both the ontology and its instances. Other approaches [3,4] separate the representation of the ontology and its instances in two parts. In these approaches, the most common practice for storing instance data is to use the so-called vertical model [12] where information is stored in triples `(subject, property, value)` a variant of which, called the binary model is to have one table per property that contains only pairs of the form `(subject, value)`.

Our approach differs from the ones listed previously. Indeed, the conceptual model of the instances is part of the OBDB data model (`EXTENT`, `SchemaProp` in the formalization of this data model). This conceptual model may be created and customized by users from the ontology (see section 5.1). Thus, many different logical models may be derrived/related to the same ontology. This possibility promotes a database approach preserving compatibility with RDBMs and promotes semantic integration of OBDBs by offering an ontology for different logical models.

3 Query Algebra for OBDB

The specific aspects of our OBDB data model, where not all the values of properties of a class are required in an instance of this class, raised the necessity to define an exploitation language for such OBDBs. [13] provides with the precise requirements we have set up at the beginning of our work for designing such a language. More details about the positioning of this language among the different database models are given in section 8. To reach this goal, we suggest to build an algebra *OntoAlgebra*, for managing OBDB databases.

Since the OBDB model uses extensively object-oriented database (OODB) features, we suggest to specialize, extend and reuse the operators issued from the *ENCORE* algebra [14] in order to get benefits of the work achieved in the context of OODBs. Signatures of the operators defined on the OBDB data model belong to $(\mathtt{E} \cup \mathtt{C}) \times 2^{\mathtt{OC} \cup \mathtt{I}} \rightarrow (\mathtt{E} \cup \mathtt{C}) \times 2^{\mathtt{OC} \cup \mathtt{I}}$. These main operators of this algebra are *OntoImage*, *OntoProject*, *OntoSelect* and *OntoOJoin*. For clarity purpose, solely these operators are formally presented below restricted to the signature $\mathtt{C} \times 2^{\mathtt{I}} \rightarrow \mathtt{C} \times 2^{\mathtt{I}}$. However, the defined semantics is adapted for querying both ontology, content and simultaneously ontology and content parts.

- OntoImage. The *OntoImage* operator returns the collection of objects resulting from applying a function to a collection of objects. Its signature is $\mathtt{C} \times 2^{\mathtt{I}} \times \mathtt{Function} \rightarrow \mathtt{C} \times 2^{\mathtt{I}}$. `Function` contains all the properties in `P` and all properties that can be defined by composing properties of `P` (path expressions). Differently from the object-oriented data model, the OBDB data model authorizes the fact that one or more of the properties occurring in the function parameter may not be valued in the extensional definition of the class. Notice that this capability weakens the data model in order to support richer and flexible descriptions than those allowed in classical OODBs. Thus, it becomes necessary to extend the domain of the `Val` function to the properties defined on the intensional definition of a class but not valued on its extensional definition. This extension requires the introduction of the `UNKNOWN` value. We call `OntoVal` this extension of `Val`. It is defined by:

$$\mathtt{OntoVal}(\mathtt{i},\mathtt{p}) = \mathtt{Val}(\mathtt{i},\mathtt{p}), \text{ if } \mathtt{p} \in \mathtt{SchemaProp}(\mathtt{TypeOf}(\mathtt{i})) \text{ else, } \mathtt{UNKNOWN} \ .$$

`UNKNOWN` is a special instance like `NULL` is a special value for SQL. Whereas `NULL` may have many different interpretations like value unknown, value inapplicable or value withheld, the only interpretation of `UNKNOWN` is value unknown, i.e., there is some value, but we don't know what it is. To preserve composition, `OntoVal` applied to a property which value is `UNKNOWN` returns `UNKNOWN` (strict interpretation). So, the semantics of `OntoImage` is defined by:

$$\mathtt{OntoImage}(\mathtt{T}, \{\mathtt{i}_1, \ldots, \mathtt{i}_n\}, \mathtt{f}) = (\mathtt{PropRange}(\mathtt{f}), \{\mathtt{OntoVal}(\mathtt{i}_1, \mathtt{f}), \ldots, \mathtt{OntoVal}(\mathtt{i}_n, \mathtt{f})\}) \ .$$

- OntoProject. The *OntoProject* operator extends *OntoImage* allowing the application of more than one function to an object. The result type is a `Tuple` which attribute names are taken as parameter. It is defined by:

$$\texttt{Project}(\texttt{T}, \texttt{I}_\texttt{t}, \{(\texttt{A}_1, \texttt{f}_1), \ldots (\texttt{A}_\texttt{n}, \texttt{f}_\texttt{n})\}) =$$
$$(\texttt{Tuple}[< (\texttt{A}_1, \texttt{PropRange}(\texttt{f}_1)), \ldots, (\texttt{A}_\texttt{n}, \texttt{PropRange}(\texttt{f}_\texttt{n})) >],$$
$$\{< \texttt{A}_1 : \texttt{OntoVal}(\texttt{i}, \texttt{f}_1), \ldots, \texttt{A}_\texttt{n} : \texttt{OntoVal}(\texttt{i}, \texttt{f}_\texttt{n}) > | \texttt{i} \in \texttt{I}_\texttt{t}\}) \ .$$

It returns the type of elements together with the set of corresponding values.
- OntoSelect. It creates a collection of objects satisfying a selection predicate. Its signature is $\texttt{C} \times 2^\texttt{I} \times \texttt{Predicate} \rightarrow \texttt{C} \times 2^\texttt{I}$ and its semantics is defined by:

$$\texttt{OntoSelect}(\texttt{T}, \texttt{I}_\texttt{t}, \texttt{pred}) = (\texttt{T}, \{\texttt{i} | \texttt{i} \in \texttt{I}_\texttt{t} \wedge \texttt{pred}(\texttt{i})\}) \ .$$

If the predicate taken as parameter of *OntoSelect* contains function applications, then `OntoVal` must be used. So, operations involving `UNKNOWN`, that may appear in a predicate, must be extended to handle this value (interpreted like `NULL`). If any operator involves this value as parameter, then it returns `UNKNOWN` (strict interpretation).

- OntoOJoin. It creates relationships between objects of two collections.

$$\texttt{OntoOJoin}(\texttt{T}, \texttt{I}_\texttt{t}, \texttt{R}, \texttt{I}_\texttt{r}, \texttt{A}_1, \texttt{A}_2, \texttt{pred}) =$$
$$(\texttt{Tuple}[< (\texttt{A}_1, \texttt{T}), (\texttt{A}_2, \texttt{R}) >], \{< \texttt{A}_1 : \texttt{t}, \texttt{A}_2 : \texttt{r} > | \texttt{t} \in \texttt{I}_\texttt{t} \wedge \texttt{r} \in \texttt{I}_\texttt{r} \wedge \texttt{pred}(\texttt{t}, \texttt{r})\}) \ .$$

- Operator *. It is the explicit polymorphic operator to distinguish between queries on instances of a single class `C` and instances of all the classes *subsumed* by `C` and denoted $\texttt{C}^*$. $\texttt{ext} : \texttt{C} \rightarrow 2^\texttt{I}$ returns the instances of a class and $\texttt{ext}^* : \texttt{C} \rightarrow 2^\texttt{I}$ its deep extent. If `c` is a class and $\texttt{c}_1, \ldots \texttt{c}_\texttt{n}$ are the direct sub-classes (in the sense of the subsumption relationship) of `c`, `ext` and $\texttt{ext}^*$ are derived recursively[4] by:

$$\texttt{ext}(\texttt{c}) = \texttt{TypeOf}^{-1}(\texttt{Nomination}(\texttt{c})) \ .$$
$$\texttt{ext}^*(\texttt{c}) = \texttt{ext}(\texttt{c}) \cup \texttt{ext}^*(\texttt{c}_1) \cup \ldots \cup \texttt{ext}^*(\texttt{c}_\texttt{n}) \ .$$

The `ext` and $\texttt{ext}^*$ make it possible to define the * operator as $^* : \texttt{C} \rightarrow \texttt{C} \times 2^\texttt{I}$ where$^*(\texttt{T}) = (\texttt{T}, \texttt{ext}^*(\texttt{T}))$.

In addition to these main operators, *OntoAlgebra* includes set operations (*OntoUnion*, *OntoDifference*, and *OntoIntersection*) and collection operations (*OntoFlatten*, *OntoNest* and *OntoUnNest*).

Next section shows how operators of *OntoAlgebra* can be optimized using the characteristics of the OBDB data model.

4 Optimizations of *OntoAlgebra*'s Operators

As identified when defining the OBDB model, some of the properties occurring in the ontology part may not be valued (and thus not available) in the corresponding instances of the content part. This is a main difference with OODBs where properties of a class are valued in the instances. As a consequence, important optimizations based on partial evaluation techniques can be set up. Indeed,

[4] To simplify notation, we extend all functions `f` by $\texttt{f}(\emptyset) = \emptyset$.

it is not necessary to search the values of a property in the instances of a class using a property defined as not available in the extensional definition of this class. This source of optimizations is characterized by the formal logical expression(invariant property) (1). When this expression evaluates to true, it becomes possible to reduce, and sometimes avoid, accesses and traversals of the content part which is cost effective.

Let $\mathtt{p} \in \mathtt{P}$, let $\mathtt{C}$ be a class, let $\mathtt{I_c}$ be a set of instances of $\mathtt{C}$,

$$\mathtt{p} \notin \mathtt{SchemaProp}(\mathtt{Nomination}(\mathtt{C})) \Rightarrow \mathtt{OntoImage}(\mathtt{C}, \mathtt{I_c}, \mathtt{p}) = \{\mathtt{UNKNOWN} | \mathtt{i} \in \mathtt{I_c}\} \ . \quad (1)$$

This optimization can be generalized to the *OntoProject* operator. It can also be applied to the *OntoSelect* and *OntoOJoin* operators when the predicate taken in parameter of these operators involves the use of a property. More precisely, assume this predicate is in conjunctive normal form and that one of the conjunctive element involves a property satisfying the logical expression (1), then, the *OntoSelect* and *OntoOJoin* operators return the empty set.

This optimization is only available for local queries on instances of a class $\mathtt{C}$. The operator * and path expressions introduce polymorphism. To optimize the polymorphic operators of the *OntoAlgebra*, it is necessary to translate them into non polymorphic operators acting on the class at each level of the polymorphic hierarchy (flattening the hierarchy).

Assume $\mathtt{p_1}$ and $\mathtt{p_2}$ are two properties which domains are respectively $\mathtt{C_1}$ and $\mathtt{C_2}$. Path expressions involving these two properties can be decomposed according to the following algebraic law:

$$\mathtt{OntoImage}(\mathtt{C_1}, \mathtt{ext}(\mathtt{C_1}), \mathtt{p_1} \circ \mathtt{p_2}) \Leftrightarrow \mathtt{OntoImage}(\mathtt{LeftOuterOntoOJoin}(\mathtt{C_1}, \mathtt{ext}(\mathtt{C_1}),^* (\mathtt{C_2}), \mathtt{A_1}, \mathtt{A_2}, \mathtt{A_1.p_1} = \mathtt{A_2.oid}), \mathtt{A_2.p_2}) \quad (2)$$

Since a result is required for each instance of $\mathtt{C_1}$, a *left outer join* is necessary. This decomposition is easily extended to paths of any length. Moreover, the same equivalence may be used for the *OntoProject* operator by decomposing each path expression taken in parameter in the same way.

Path expressions may also appear in the predicate of the *OntoSelect* and *OntoOJoin* operators. For example, assume the predicate is $\mathtt{p_1} \circ \mathtt{p_2}\ \theta\ \mathtt{c}$ where θ is a comparison operator and $\mathtt{c}$ a constant, then, the following equivalence can be used to decompose this predicate :

$$\mathtt{OntoSelect}(\mathtt{C_1}, \mathtt{ext}(\mathtt{C_1}), \mathtt{p_1} \circ \mathtt{p_2}\ \theta\ \mathtt{c}) \Leftrightarrow \mathtt{OntoImage}(\mathtt{OntoSelect}(\mathtt{LeftOuterOntoOJoin}(\mathtt{C_1}, \mathtt{ext}(\mathtt{C_1}),^* (\mathtt{C_2}), \mathtt{A_1}, \mathtt{A_2}, \mathtt{A_1.p_1} = \mathtt{A_2.oid}), \mathtt{A_2.p_2}\ \theta\ \mathtt{c}), \mathtt{get\_A_1\_Value}) \ . \quad (3)$$

The * operator can be removed using the set operator *OntoUnion*. Assume $\theta_1 \ldots \theta_n$ to be a set of *OntoAlgebra* operators, $\mathtt{C}$ a class having $\mathtt{C_1} \ldots \mathtt{C_n}$ as direct sub-classes. Removing * operator is preformed by applying the following equivalence recursively (unfolding operation).

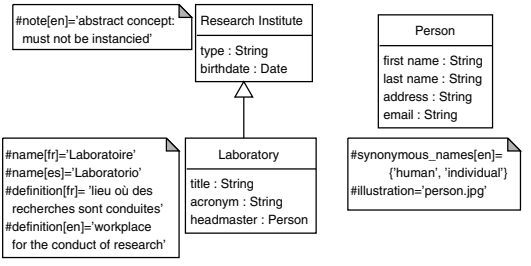


Fig. 1. Ontology example

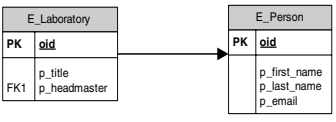


Fig. 2. Content example

$$\theta_1(\ldots\theta_n(^*(\mathtt{C})) \Leftrightarrow \theta_1(\ldots\theta_n(\mathtt{C},\mathtt{ext}(\mathtt{C}))\ \mathtt{OntoUnion}$$
$$\theta_1(\ldots\theta_n(^*(\mathtt{C_1}))\ \mathtt{OntoUnion}\ldots\mathtt{OntoUnion}\ \theta_1(\ldots\theta_n(^*(\mathtt{C_n})) \quad (4)$$

The (1), (2), (3) and (4) algebraic laws will be exploited by the query plans presented in section 6 below.

5 The OntoQL Language

OntoQL is the OBDB exploitation language built on the defined database model. An overview of the querying capabilities of OntoQL has been given in [15]. Its semantics is given by the *OntoAlgebra* previously defined. This section presents the details of this language (DDL and QL). A toy example, presented on figure 1, is used along this description in order to avoid complex syntactic definitions. Next subsections focus on the specific parts of this language. We will discuss the positioning of this language among database and semantic web languages in section 8.

5.1 The Data Definition (DDL) and Manipulation (DML) Parts of OntoQL

OntoQL allows to create, alter and drop concepts of ontologies (classes, properties ...) as well as their attributes values (name, definition ...). Let's consider the following expression :

```
CREATE #CLASS Laboratory EXTENDS "Research Institute" (
 DESCRIPTOR(#name[fr,es] = ('Laboratoire','Laboratorio'),
            #definition[fr] = 'lieu où des recherches sont conduites',
            #definition[en] = 'workplace for the conduct of research')
 PROPERTIES(title String, headmaster Person, acronym String) );
```

This expression creates an ontology class named `Laboratory` in English (the default language) as a subclass of `Research Institute`. Thus, it inherits the properties of `Research Institute`. `DESCRIPTOR` and `PROPERTIES` clauses make the distinction between the definition of the attributes values describing a class (name,

remark, note . . .) and the definition of its characterisation properties (those fixed by the user). This distinction is carried by the attributes prefix `#` (see section 5.2).

On the content part, an extent can be attached to this class by the following expression:

```
CREATE EXTENT OF Laboratory (title, headmaster);
```

Notice that the `acronym` property will not be valued in the content part. One may define another content for the class `Laboratory` according to another logical database model. When executed, this expression creates a relational schema presented in figure 2 to store instances of this class.

Data Manipulation Language (DML) operators are also provided by OntoQL. These operators may add, delete and update each parts of an OBDB. Indeed, concepts and their instances may be managed by these operators. Moreover, since the metadata describing the ontology model are themselves stored in a relational database, the same operators allow adding/deleting/updating attributes of the ontology model (e.g. adding the attributes `unit`, `comment`, etc. defining a new translation language more than English, etc.).

5.2 The Query Language Part of OntoQL

The query language part of *OntoQL* is designed as an extension of SQL to query ontologies, their contents and both ontologies and contents stored in an OBDB. The syntax of a query is given by:

```
  SELECT attributeList, propertyList, iteratorList
    FROM iteratorDeclarationList
   WHERE condition
GROUP BY attributeList, propertyList, iteratorList
  HAVING condition
ORDER BY attributeList, propertyList
```

where `attributeList` (resp. `propertyList`) is a list of attributes (resp. properties); `iteratorList` is a list of iterators declared in the `FROM` clause;
`iteratorDeclarationList` introduces iterators over a set of entity and/or class instances. Moreover, the `SELECT` block of *OntoQL* supports the following features, all expressed by *OntoAlgebra* operators composition.

- *Path expressions.* Associations may be traversed using the dot notation.
- *Polymorphic query.* The `*` operator is provided to distinguish between local queries on instances of a class/entity `C` and instances of all the classes/entities denoted `C*` subsumed by `C`.
- *Dependent collection.* A collection returned as the value of a property/attribute may be traversed using an iterator introduced in the `FROM` clause.
- *Nested queries.* Queries may be nested not only in the `WHERE` clause but also in the `SELECT` and `FROM` clauses.
- *Aggregate functions. OntoQL* provides aggregate functions `count`, `sum`, `avg`, `min` and `max`.

- *Quantification.* Existential (`ANY`, `SOME`) and universal (`ALL`) quantification may be expressed.
- *Set operators.* `Union`, `Intersection` and `Except` operators are provided.

Next subsections show how this general model of an *OntoQL* query is used to express query on ontology, on content and both on ontology and on content. Moreover, these sections show on several query examples, specific usages of *OntoQL* to exploit ontology characteristics, content characteristics or both.

Ontology Querying. Ontologies querying retrieve descriptive information from the ontology part. The `FROM` clause of an ontology query introduces iterators over instances of predefined entities (`class`, `property`) of the ontology model as well as on user-defined entities. The `SELECT` clause defines projection on predefined attributes (`name`, `definition`, `scope`, `superclasses` ...) and user-defined attributes. The value of some attributes, such as `name` are given in different natural languages. The query `Q1` searches for the English name of the class which French name is "`Institut de Recherche`".

```
Q1. SELECT #name[EN] FROM #class WHERE #name[FR]='Institut de Recherche'
```

Remember that the prefix `#` is used to distinguish between attributes of entities and properties of classes.

Moreover classes and properties are implicitly named. For example, the query `Q2` uses `Research Institute` class name to retrieve the name in French of the properties defined on this class.

```
Q2.SELECT p.#name[FR] FROM "Research Institute".#properties
```

This capability is also offered by OQL. However, to use it on a given object of a class, users must explicitly name this object.

Content Querying. The `FROM` clause of a content query introduces iterators over instances of ontology classes and the `SELECT` clause defines projection on properties defined on this class but not necessary provided by the `CREATE EXTENT` clause. The following queries search for the names of all laboratories with an English (`Q3a`) a French (`Q3b`), using external identifiers (`Q3c`) and internal identifiers (`Q3d`) queries:

```
Q3a.SELECT acronym   FROM Laboratory  Q3c.SELECT @710C-01 FROM @7194-01
Q3b.SELECT accronyme FROM Laboratoire Q3d.SELECT !1012 FROM !1068
```

Here `!x` and `@x` are respectively internal identifiers (known by database developers) and external references (like URI).

Ontology and Content Querying. *OntoQL* introduces an iterator over instances of classes retrieved by an ontology query. `Q4a` query illustrates this feature using a *dependant collection.*

```
Q4a.SELECT i.oid, i.p, p.#name[en]
      FROM C in #class, p in C.#properties, i in C*
     WHERE C.#name[fr] like 'Per%'
```

`Q4b` is equivalent to `Q4a`. It uses a `SELECT` operator in the `FROM` clause to access simultaneously the ontology and the content parts (*nested query*).

```
Q4b.SELECT i.oid, i.p, p.#name[en]
      FROM C in (SELECT C FROM C in #class
                 WHERE C.#name[fr] like 'Per%'),
           p in C.#properties, i in C*
```

This query retrieves classes which name begin with `"Per"`. The iterator `i` ranges over instances of these classes and is used to access and return property English name and value for these instances.

OntoQL proposes the operator `typeof` to retrieve the base class of a content instance. This operator allows to express queries from the content to the ontology. For example, the query `Q5` searches for the address and email of all polymorphic instances of the class `Person` and uses the operator `typeof` to retrieve the French name of the base class of these instances.

```
Q5.SELECT i.address, i.email, typeof(i).#name[fr] FROM i in Person*
```

The `typeof` operator returns only one base class for an instance. This query could not be written this way if multi-instantiation was allowed.

6 Processing of OntoQL

We have implemented *OntoQL* and the *OntoAlgebra* operators on an OntoDB prototype. Demonstrations of this prototype are available at http://www.plib.ensma.fr/plib/demos/ontodb/index.html. This section briefly outlines how these operators are processed on this platform.

The prototype considered is an implementation of the OntoDB data model in a object-relational database (ORDBMS), namely PostgreSQL [16]. In this prototype, the link between the ontology and its content parts are defined using an identifier. To simplify, assume that the identifier of the ontological (intensional) definition of a class is `cid`, then its content (extensional) definition is represented by a table identified by `ecid`. A similar mechanism for properties is used. Indeed, assume that the identifier of a property is `pid` in the intensional definition of a class, then it is represented by a column identified by `ppid` if used in the content definition.

To process *OntoAlgebra* operators, they are translated in the underlying query language, i.e. SQL99. With this approach, the optimization process is split into two sub-processes. The first one is related to the *OntoQL* engine and the second one is performed by the underlying database engine. The translation process follows six identified steps.

1. **Logical query plan generation.** The query, written in *OntoQL*, is parsed and turned into an expression tree involving *OntoAlgebra* operators in its nodes (logical query plan). Entities, classes, properties and attributes occur in leaves.
2. **Logical query plan transformation.** Path expressions and * operators are removed from the logical query plan using equivalence algebraic laws (2), (3) and (4) defined on *OntoAlgebra* (see section 4). In this step, we use an algorithm avoiding multiple decomposition of identical paths and thus avoiding unnecessary join operations.
3. **Optimize the tree.** The optimization situations, identified in section 4, are used to reduce the logical query plan. This step is performed together with the previous step to avoid duplicating unnecessary parts of the tree.
4. **Translation of an *OntoAlgebra* tree to relational algebra trees.** This translation is achieved by applying the following rules:
   (a) the identifier of the intensional definition of a class is replaced by the identifier of its extensional definition. If the class is abstract then its identifier is replaced by the name of an empty table (e.g., Dual in Oracle);
   (b) if the property is not used in the extensional definition of a class, `UNKNOWN` is translated to `NULL`.
   (c) *OntoImage* and *OntoProject* are translated into the projection operator of the relational algebra. Other operators of *OntoAlgebra* are translated into their relational counterpart.

   An *OntoQL* query often requires access to the content of an OBDB according to the query on the ontology part. Thus, this translation may require to build more than one relational algebra tree.
5. **Optimization of the relational algebra trees.** This step consists in using the different algebraic laws that hold for relational algebra to turn the relational trees into equivalent trees that may be executed more efficiently by the underlying ORDBMS. The ORDBMS optimizer may perform other optimizations it supports.
6. **Translation of the relational algebra trees into SQL queries.** The optimized relational trees are translated into SQL queries according to the underlying ORDBMS and executed to get the *OntoQL* query result.

7 Example

To illustrate our language proposal, let us develop a practical example showing how a query, written in *OntoQL*, is processed. This example extends the previous one by precising the address property in the class person.

An Example of Data Model. Figure 3 shows an UML data model. Specific annotations `d`/`v` are added to the property names to take into account the specific features of the OBDB data model. `d` means that this property is defined on this class ; `v` means that this property is valued in the extensional definition of this class. This schema is defined to manage names of persons. Students and

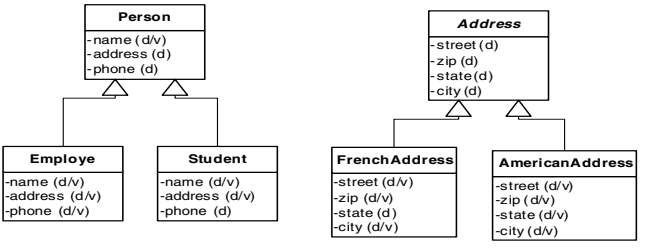


Fig. 3. Schema example

employees are also described by their addresses. Since `Address` is an abstract class (its name is in italic), addresses are located either in the USA or in France. Notice that for French addresses, the property `state` is not valued on instances.

An Example of Query. Assume we want to find in which cities of the state Utah some persons are living. To answer this query, an *OntoQL* statement using English attributes (left) or French ones (right) may be written:

```
SELECT address.city                SELECT adresse.ville
  FROM Person*              <=>      FROM Personne*
 WHERE address.state='Utah'         WHERE adresse.etat='Utah'
```

Query Processing Steps. The first step of our processing generates the logical plan presented in Fig. 4 (a). The path expressions are removed from this query plan. If the application of the property `address` is decomposed in each path, the transformed logical plan is the one presented in Fig. 4 (b). In this logical plan, the upper left outer join is unnecessary. Therefore, application of the property `address` is decomposed only once and paths composed with this property are changed using an alias of the class `Address`. The query plan resulting from this processing is presented in Fig. 4 (c). In step 3 and 4, * operators applied to `Person` and `Address` are removed. Because `address` is not used on classes `Person`, optimization (1) allows to deduce that the result of the *LeftOuterOntoOJoin* is $\{< \mathtt{p}, \mathtt{UNKNOWN} > | \mathtt{p} \in \mathtt{ext}(\mathtt{Person})\}$. Thus, the predicate $\mathtt{a.state} =' \mathtt{Utah}'$ is always `UNKNOWN`. As a consequence, this query doesn't return any result for the class `Person`. Therefore our logical query plan must be duplicated for the classes `Employee` and `Student` only. Let's consider removing of * from class `Address`. Because the property `state` is not used on classes `Address` and `FrenchAddress` the result of the *OntoSelect* operator is empty and it is not necessary to run it. Finally, our logical query plan must only be duplicated for the `AmericanAddress` class. Logical query resulting from this processing is presented in Fig. 4 (d).

In this example, translation of this query plan to a relational query plan is straightforward. It consists, first, in replacing names of classes and properties by names of tables and columns corresponding to their extensional definitions, and, second, in switching *OntoAlgebra* operators to their corresponding relational

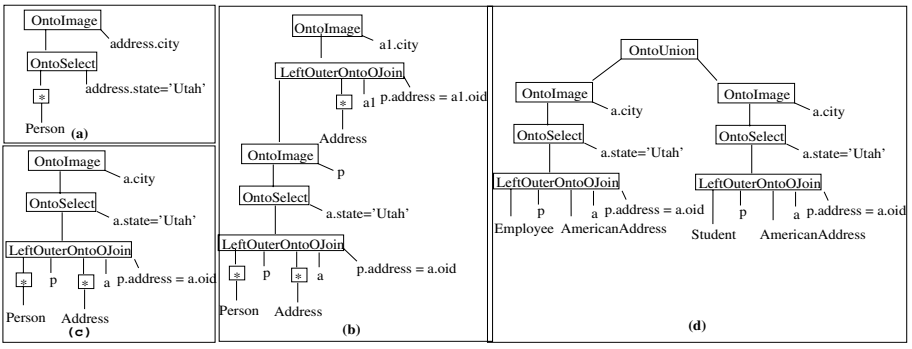


Fig. 4. Logical query plans for our query example

operators. In step 6, the relational tree may be modified to replace operators not supported by the underlying DBMS (e.g., *outer join*) or to optimize it according to the generation and optimization of the logical query plan supported by the DBMS. Without modification of our logical query plan, the obtained SQL query running on the underlying DBMS is:

```
SELECT a.pcity
  FROM eEmployee p LEFT OUTER JOIN eAmericanAddress a
                                   ON p.paddress = a.oid
 WHERE a.pstate='Utah'
UNION
SELECT a.pcity
  FROM eStudent s LEFT OUTER JOIN eAmericanAddress a
                                   ON s.paddress = a.oid
WHERE a.pstate='Utah'
```

This query returns the results of our *OntoQL* query.

8 Related Work

OntoQL is a language based on a database model for exploiting ontologies and the knowledge they describe. Therefore, one can compare it with database languages on the one hand and with semantic web based languages on the other hand.

8.1 Database Exploitation Languages

Compared to classical database languages, *OntoQL* preserves upward compatibility with existing database exploitation languages associated to different layered database models.

– RDB. When an user is aware of internal identifiers for tables and columns of the OBDB model, classical SQL can be used to retrieve and manage table data.

- ORDB. When all properties values of a class are provided at the instance level, one can use the OBDB model as an OODB model and use *OntoQL* constructs as an exploitation language.
- When the previous conditions are not fulfilled, we can use *OntoQL* as an ontology exploitation language as shown in this paper.
- Finally, the top layer is the one of a linguistic based exploitation of an OBDB. Indeed, when the attributes #name, #remark, #synonymous, #translations are exploited by *OntoQL* constructs, it is possible to envisage a linguistic exploitation as shown in queries Q2 and Q3 given in section 5.2 .

Moreover,*OntoQL* has been defined as an extension of SQL to exploit an OBDB model defined for exploiting data and their semantics and for semantic integration. According to these applications, related languages are multidatabase languages like SchemaSQL [17] or MSQL [18]. *OntoQL* shares with these languages the capability to express queries on data independently of their schemas. However, whereas these languages use the system catalog as an abstraction for database schemas, *OntoQL* uses the ontologies themselves to encode this abstraction providing a dynamic approach for encoding different abstractions corresponding to different point of views of application domains. Consequently, *OntoQL* presents many differences with these languages such as its object-oriented nature or its independency w.r.t the model (relational, object-relational, object) used to represent the schemas of the data.

The SOQA-QL [19] language (SIRUP project) allows querying ontologies and the data they describe independently of the ontology model and of the hardware/software used platform. Like *OntoQL*, the main application of SOQA-QL is semantic integration. Moreover, they are both based on SQL and defined on a core ontology model (the SOQA Ontology Meta Model for SOQA-QL) representing the shared modelling capabilities of some ontology models in order to provide an access to data independently of the used ontology model. Nevertheless, there are crucial differences between these two languages. First, in the opposite of SOQA-QL, the core ontology model of *OntoQL* can be extended to take into account particularities of some ontology models (e.g. adding new attributes that characterize ontology model concepts). To provide this capability, *OntoQL* is based on an algebra not tight to the core ontology model whereas the SOQA-QL algebra (i.e, encoded in the SIRUP Ontology Query API) provides access methods for all ontological components defined in the SOQA Ontology Meta Model and the user does not have the possibility to dynamically update this API. Another difference is that SOQA-QL and OntoQL do not keep the same level of compatibility with SQL. Indeed, whereas SOQA-QL queries on ontologies are expressed in a SQL-like syntax, SOQA-QL queries on data require to call the `value` function for each projection. Moreover, SOQA-QL doesn't provide all the useful operators of the object-oriented paradigm introduced in SQL99 like path expressions or collection manipulation. Last, SOQA-QL is a platform independent language whereas *OntoQL* is a language for OBDBs. As a consequence, *OntoQL* assumes that the data queried are stored in an OBDB and therefore it addresses some database problems such as query optimization or data definition

and manipulation specific to OBDBs that cannot be considered by SOQA-QL due to its platform independency.

8.2 Semantic Web Exploitation Languages

Over the last years, many semantic web query languages have been proposed. Recently, a survey [20] classifies these languages into six categories with three main categories:

1. the SPARQL [21] category which groups query languages considering all data, both ontologies and their instances, as a set of triples;
2. the RQL [6] category which gathers query languages that make the distinction between the ontology and the data information (ontology instances). These languages provide operators to exploit the subsumption hierarchies of classes/properties and to combine data and schema querying;
3. the deductive languages (e.g, OWL-QL [7]) category for query languages expressing rules that define how new data can be derived from existing ones and thus be in the answer of a query.

OntoQL shares many characteristics with the second category. Indeed, like these languages, *OntoQL* offers the possibility to query ontologies, instances and both ontologies and instances but it does not offer rule based reasoning. However, contrary to these languages, *OntoQL* presents the following characteristics:

- *SQL Upward Compatibility. OntoQL* extends the SQL syntax and semantics. Thus, it has the benefits of SQL and it can be implemented as additional components of existing ORDBMS.
- *Schema manipulation.* In a lot of semantic integration approaches, the manipulation of the structure of the data is useful. *OntoQL* allows retrieving, creating, altering and dropping the schema of the data thanks to the possibility left to manage the metadata in both of the instance part or of the ontology part. Moreover, *OntoQL* uses this schema for query optimization;
- *Exploitation of multi-lingual definitions.* Concepts describe by an ontology may be associated with a linguistic representation in different natural languages. Using *OntoQL*, one can retrieve this representation and express queries in different natural languages.
- *Ontology model independency. OntoQL* is based on a core ontology model which can be extended to take into account specific features of a given ontology model. The RDF meta-model may also be extended. However, query languages such as RQL restrict this extension to specializing the meta-classes `rdfs:Class` (the class of all classes) and `rdfs:Property` (the class of all of properties) to ensure a clear separation of the three abstraction layers of RDF and RDFS (data, ontologies and meta-schema). There is no such restriction with *OntoQL*. As a consequence, new attributes (e.g, comment, remark, illustration) and new entities (e.g, document, restriction) may be added and managed using *OntoQL*.

Regarding the expressive power, *OntoQL* doesn't allow to express query without specifying the search scope (the `FROM` clause is mandatory) and doesn't support yet the multi-instanciation capability. However, *OntoQL* is equipped with *grouping operators* (`GROUP BY`), *sorting operator* (`ORDER BY`) and *collection manipulation* operators not yet provided by semantic web query languages [22].

9 Conclusion

In this paper, we have formally presented an OBDB data model called OntoDB. This model differs from classical database models as well as other OBDB data models propositions. The need for a new exploitation language to manage this OBDB data model was a result of this constatation.

As a consequence, we proposed a formal algebra of operators together with the definition of the *OntoQL* database exploitation language for managing OBDBs. We have shown on some query examples how this language exploits the characteristics of the OBDB data model to support the multilingual querying of OBDBs at the ontology, content and both ontology and content levels. As a further step, we have defined an operational approach implementing these operators on top of a relational database model. The interested reader is invited to see the demonstrations of this prototype available at

http://www.plib.ensma.fr/plib/demos/ontodb/index.html.

OntoQL differs from the semantic web languages in the sense that it originates from databases approaches. It is built on top of RDBs and ORDBs preserving an upward compatibility and getting benefits of the power of database approaches keeping the possibility to exploit Web Semantic data.

For the future we plan to work in two directions related to database and to the semantic web. From a database perspective, it is important to study the query optimization on large databases and then the scalability of the OntoQL implementations in order to address large sets of data. Optimizations on the algebra operators and their composition shall be studied as well.

From a semantic web point of view, it it is planned to relax some assumptions made in the OBDB data model in order to offer an efficient storage capability for the instances described in the logic based approaches for ontologies like in OWL. The objective is to unify the proposition issued from the semantic web community which extensively use triples and descriptive logic and their derivatives, and the object orientation and database communities which use strong typing approaches.

References

1. Brickley, D., Guha, R.: RDF Vocabulary Description Language 1.0: RDF Schema. World Wide Web Consortium. (2004)
2. Bechhofer, S., van Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D.L., Patel-Schneider, P.F., Stein, L.A.: OWL Web Ontology Language Reference. World Wide Web Consortium. (2004)

3. Alexaki, S., Christophides, V., Karvounarakis, G., Plexousakis, D., Tolle, K.: The ics-forth rdfsuite: Managing voluminous rdf description bases. In: SemWeb. (2001)
4. Broekstra, J., Kampman, A., van Harmelen, F.: Sesame: A generic architecture for storing and querying rdf and rdf schema. In: International Semantic Web Conference. (2002) 54–68
5. Pan, Z., Heflin, J.: Dldb: Extending relational databases to support semantic web queries. In: PSSS. (2003)
6. Karvounarakis, G., Alexaki, S., Christophides, V., Plexousakis, D., Scholl, M.: Rql: a declarative query language for rdf. In: WWW. (2002) 592–603
7. Fikes, R., Hayes, P.J., Horrocks, I.: Owl-ql - a language for deductive query answering on the semantic web. J. Web Sem. **2** (2004) 19–29
8. Bellatreche, L., Pierra, G., Xuan, D.N., Dehainsala, H., Aït-Ameur, Y.: An a priori approach for automatic integration of heterogeneous and autonomous databases. In: DEXA. (2004) 475–485
9. Pierra, G.: Context-explication in conceptual ontologies: Plib ontologies and their use for industrial data. Journal of Advanced Manufacturing Systems (2004)
10. Jean, S., Pierra, G., Aït-Ameur, Y.: Domain ontologies: a database-oriented analysis. In: Web Information Systems and Technologies (WEBIST'2006). (2006) 341–351
11. Harris, S., Gibbins, N.: 3store: Efficient bulk rdf storage. In: PSSS. (2003)
12. Agrawal, R., Somani, A., Xu, Y.: Storage and querying of e-commerce data. In: VLDB '01: Proceedings of the 27th International Conference on Very Large Data Bases, Morgan Kaufmann Publishers Inc. (2001) 149–158
13. Jean, S., Pierra, G., Aït-Ameur, Y.: Ontoql: an exploitation language for obdbs. In: VLDB PhD Workshop. (2005) 41–45
14. Shaw, G.M., Zdonik, S.B.: A query algebra for object-oriented databases. In: ICDE. (1990) 154–162
15. Jean, S., Aït-Ameur, Y., Pierra, G.: Querying ontology based databases. the ontoql proposal. In: 18^{th} International Conference on Software Egineering and Knowledge Engineering (SEKE'2006). (2006) 166–171
16. Douglas, K., Douglas, S.: PostgreSQL. New Riders Publishing (2003)
17. Lakshmanan, L.V.S., Sadri, F., Subramanian, I.N.: Schemasql - a language for interoperability in relational multi-database systems. In: VLDB. (1996) 239–250
18. Litwin, W., Abdellatif, A., Zeroual, A., Nicolas, B., Vigier, P.: Msql: A multidatabase language. Inf. Sci. **49** (1989) 59–101
19. Ziegler, P., Sturm, C., Dittrich, K.R.: Unified querying of ontology languages with the sirup ontology query api. In: BTW. (2005) 325–344
20. Bailey, J., Bry, F., Furche, T., Schaffert, S.: Web and semantic web query languages: A survey. In: Reasoning Web. (2005) 35–133
21. W3C: Sparql. visited on (2005) retrieved from http://www.w3.org/TR/rdf-sparql-query/.
22. Haase, P., Broekstra, J., Eberhart, A., Volz, R.: A comparison of rdf query languages. In: SemWeb. (2004)

Description Logic Reasoning with Syntactic Updates

Christian Halashek-Wiener[1], Bijan Parsia[2], and Evren Sirin[1]

[1] Department of Computer Science,
University of Maryland, College Park, MD USA
{halasche, sirin}@cs.umd.edu
[2] School of Computer Science,
The University of Manchester, UK
bparsia@cs.man.ac.uk

Abstract. Various data sources on the Web tend to be highly dynamic; this is evident in prominent Web services frameworks in which devices register or deregister their descriptions quite rapidly and in Semantic Web portals which allow content authors to modify or extend underlying ontologies and submit content. Such applications often leverage Description Logic (DL) reasoning for a variety of tasks (e.g., classifying Web service descriptions, etc); however, this can introduce substantial overhead due to content fluctuation, as DL reasoners have only been considered for relatively static knowledge bases. This work aims to provide more efficient DL reasoning techniques for frequently changing instance bases (ABoxes). More specifically, we investigate the process of incrementally updating tableau completion graphs used for reasoning in the expressive DLs $\mathcal{SHOQ}$ and $\mathcal{SHIQ}$, which correspond to a large subset of the W3C standard Web Ontology Language, OWL-DL. We present an algorithm for updating completion graphs under the syntactic addition and removal of ABox assertions. We also provide an empirical analysis of the approach through an implementation in the OWL-DL reasoner, Pellet.

1 Introduction

Recently, there has been increased interest in providing formal representation of Web content using ontologies that correspond to expressive Description Logics (DLs). Due to data sources that produce fluctuating data, there exists a variety of reasoning use cases which require frequent updates at both the assertional (ABox) and terminological (TBox) levels, some of which are briefly highlighted below:

- Prominent web services frameworks (e.g., OWL-S) use Description Logics for service discovery and matchmaking [24, 25, 21]. Services, especially device services in pervasive contexts, may register or deregister their descriptions (and supporting ontologies) quite rapidly, yet the matchmaking service must remain responsive.
- Semantic Web portals often allow content authors to modify or extend the ontologies which organize their site structure or page content. While in some cases a "defer update" strategy is acceptable, in general it is more gratifying to users if changes they make are reflected immediately in the site.

R. Meersman, Z. Tari et al. (Eds.): OTM 2006, LNCS 4275, pp. 722–737, 2006.

- Perhaps the single most common use of Description Logic reasoners is in ontology editors. Most editors do not do continuous reasoning while one is editing (one exception is [18]), relying on an analogue of the edit-compile-test loop of most programming environments. However, if this cycle is very long (e.g., hundreds of seconds) then users will be forced to perform larger sets of edits before testing. This discourages experimentation, particularly in debugging contexts.
- Syndication (publish/subscribe) systems on the Web have recently been transitioning to more expressive approaches; that is subscribers (and publishers) are provided with more expressive means for describing their interests (resp. published content), enabling more accurate dissemination. Recently, there has been interest in utilizing OWL and DL reasoning for the purpose of matching published content with subscription requests [8,9,26,17]. When disseminating time sensitive information (e.g., in the financial domain), efficient reasoning for frequently changing KBs (due to continuously published information) will be critical in achieving practical DL-based approaches.

While there exists such use cases for reasoning under changing data, current DL reasoning algorithms have been developed considering relatively static knowledge bases. In this paper, we investigate performing incremental consistency checking in the expressive Description Logics $\mathcal{SHOQ}$ and $\mathcal{SHIQ}$, which correspond to a large subset of the W3C standard Web Ontology Language, OWL-DL. In particular, we present an approach for incrementally updating tableau completion graphs created during consistency checks under syntactic addition and removal of ABox (instance) assertions. This provides a critical step towards DL reasoning over fluctuating data, as it has been shown that in KBs with substantially sized ABoxes, consistency checking can dominate reasoning time; further, all standard reasoning tasks are reduced to consistency checks. Lastly, we provide an empirical analysis of the optimizations through an experimental implementation in an open source OWL-DL reasoner, Pellet.

2 Preliminaries

In this section, we briefly discuss background information directly relevant to this work. First, we present the syntax and semantics of the Description Logic $\mathcal{SHOIQ}$, which corresponds to OWL-DL, with the slight extension of qualified cardinality restrictions. Additionally, we provide a brief overview of tableau algorithms for Description Logic reasoning.

2.1 $\mathcal{SHOIQ}$ Description Logic

Let N_C, N_R, N_I be non-empty and pair-wise disjoint sets of *atomic concepts*, *atomic roles* and *individuals* respectively. The set of $\mathcal{SHOIQ}$ roles (roles, for short) is the set $N_R \cup \{R^- | R \in N_R\}$, where R^- denotes the inverse of the atomic role R. Concepts are inductively defined using the following grammar:

$$C \leftarrow A \mid \neg C \mid C_1 \sqcap C_2 \mid C_1 \sqcup C_2 \mid \exists R.C \mid \forall R.C \mid \bowtie nS.C \mid \{a\}$$

where $A \in N_C$, $a \in N_I$, $C_{(i)}$ a $\mathcal{SHOIQ}$ concept, R a role, S a *simple* role [1] and $\bowtie \in \{\leq, \geq\}$. We write $\top$ and $\bot$ to abbreviate $C \sqcup \neg C$ and $C \sqcap \neg C$ respectively.

A *role inclusion axiom* is an expression of the form $R_1 \sqsubseteq R_2$, where R_1, R_2 are roles. A *transitivity axiom* is an expression of the form $Trans(R)$, where $R \in V_R$. An RBox R is a finite set of role inclusion axioms and transitivity axioms. For C, D concepts, a *concept inclusion axiom* is an expression of the form $C \sqsubseteq D$. A TBox T is a finite set of concept inclusion axioms. An ABox A is a finite set of concept assertions of the form $C(a)$ (where C can be an arbitrary concept expression) and role assertions of the form $R(a, b)$. A *knowledge base* $\mathsf{K} = (\mathsf{T}, \mathsf{R})$ consists of a TBox and an RBox.

An *interpretation* $\mathcal{I}$ is a pair $\mathcal{I} = (\mathcal{W}, .^{\mathcal{I}})$, where $\mathcal{W}$ is a non-empty set, called the *domain* of the interpretation, and $.^{\mathcal{I}}$ is the *interpretation function*. The interpretation function assigns to $A \in N_C$ a subset of $\mathcal{W}$, to each $R \in N_R$ a subset of of $\mathcal{W} \times \mathcal{W}$ and to each $a \in N_I$ an element of $\mathcal{W}$. The interpretation function is extended to complex roles and concepts as given in [14].

The satisfaction of a $\mathcal{SHOIQ}$ axiom α in an interpretation $\mathcal{I}$, denoted $\mathcal{I} \models \alpha$ is defined as follows: (1) $\mathcal{I} \models R_1 \sqsubseteq R_2$ iff $(R_1)^{\mathcal{I}} \subseteq (R_2)^{\mathcal{I}}$; (2) $\mathcal{I} \models Trans(R)$ iff for every $a, b, c \in \mathcal{W}$, if $(a, b) \in R^{\mathcal{I}}$ and $(b, c) \in R^{\mathcal{I}}$, then $(a, c) \in R^{\mathcal{I}}$; (3) $\mathcal{I} \models C \sqsubseteq D$ iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$; The interpretation $\mathcal{I}$ is a model of the RBox R (respectively of the TBox T) if it satisfies all the axioms in R (respectively T). $\mathcal{I}$ is a model of $\mathsf{K} = (\mathsf{T}, \mathsf{R})$, denoted by $\mathcal{I} \models \mathsf{K}$, iff $\mathcal{I}$ is a model of T and R.

2.2 Tableau Algorithms

The algorithm presented here is based on the tableau decision procedure for ABox consistency checking in $\mathcal{SHOQ}$ [12] and $\mathcal{SHIQ}$ [13]. DL tableau-based algorithms decide the consistency of an ABox A w.r.t a TBox T (by TBox, we are additionally referring to all axioms for roles) by trying to construct (an abstraction of) a common model for A and T, called a *completion graph* [14]. Formally, a completion graph for an ABox A with respect to T is a directed graph $G = (\mathcal{V}, \mathcal{E}, \mathcal{L}, \dot{\neq})$. Each node $x \in \mathcal{V}$ is labeled with a set of concepts $\mathcal{L}(x)$ and each edge $e = \langle x, y \rangle$ with a set $\mathcal{L}(e)$ of role names. The binary predicate $\dot{\neq}$ is used for recording inequalities between nodes.

The completion graph is constructed by repeatedly applying a set of *expansion rules*. Whenever a contradiction is encountered, a DL reasoner will either backtrack and select a different non-deterministic choice, or report the inconsistency and terminate if no choice remains to be explored. While there may exist more than one model for A and T, the tableau algorithm will only find one such model. We also point out that *blocking* is utilized to ensure termination of tableau algorithms [11]. Blocking is used to detect cycles encountered during the application of tableau expansion rules, therefore stopping the expansion of a branch when the same labels reoccur. For example, blocking is clearly necessary to ensure termination for the following KB, $\mathsf{K} = \{a{:}C, C \sqsubseteq \exists R.C\}$. Further details can be found in [11]. We lastly note that in our work, we utilize a slightly modified version of the $\mathcal{SHOQ}$ and $\mathcal{SHIQ}$ expansion rules, which are augmented for axiom tracing as defined in [10,16,15].

[1] See [14] for a precise definition of simple roles.

3 Syntactic ABox Updates

In this work, we consider ABox addition and deletion of individual equality and inequality assertions $x = y$ and $x \neq y$, concept assertions $x{:}C$ and role assertions $\langle x, y\rangle{:}R$. In general updating ABox assertions in the presence of a TBox and an RBox brings up several issues with the semantics.

For the purpose of this work, we adopt syntactic changes/updates of ABox assertions, which we refer to as *Syntactic Updates*. By *syntactic* we refer to the explicitly asserted ABox facts; this is similiar to the distiction between belief bases and belief sets in belief revision literature [20]. Intuitively, *Syntactic Updates* can be described as an update in which all new ABox assertions are directly added (or removed) to the asserted (base) axioms; therefore the only changes that occur are those explicitly stated in the ABox update. Removing an assertion from the ABox under these semantics does not guarantee that the removed assertion will not be entailed anymore. Furthermore, in this work we do not address resolving inconsistencies introduced by the addition of a new axiom. Formally, we describe this update as follows:

Definition 1. *(Syntactic Updates) Let S be the set of assertions in an initial ABox* A*. Then under* Syntactic Updates*, updating S with an ABox addition (resp. deletion) α, written as $\mathsf{A} + \alpha$ (resp. $\mathsf{A} - \alpha$), results in an updated set of ABox axioms S' such that $S' = S \cup \{\alpha\}$ (resp. $S' = S \setminus \{\alpha\}$).*

This type of ABox update is different when compared to related work in formal update semantics [19,23] and belief revision [5,6] for DLs, however real world use of this type of changes is directly present in many document-oriented services, including Semantic Web Service repositories, publish/subscribe (syndication) application, Semantic Web portals, etc. For example, an OWL-S Web Service description can be seen as a set of ABox assertions and publishing/retracting this service description to/from a repository would be typically done under *Syntactic Updates*; therefore we feel these semantics is warranted. Lastly, we note that ABox additions under *Syntatic Updates* is similar to the *expansion* operator as traditionally defined belief revision [1].

4 Update Approach

The goal of the approach presented here is to update a previously constructed completion graph in such a way that it is guaranteed to correspond to a model of the updated KB (in the event some model exists). The approach presented here is applicable to the Description Logics $\mathcal{SHOQ}$ [12] and $\mathcal{SHIQ}$ [13], as the difference between the two completion algorithms is independent of the update algorithm.

4.1 ABox Additions

Conceptually, tableau algorithms for $\mathcal{SHOQ}$ [12] and $\mathcal{SHIQ}$ [13] can be thought of as incremental in that *expansion rules* are repeatedly applied in a non-deterministic manner to labels in the completion graph. Hence, new ABox assertions can be added even after previous completion rules have been fired for existing nodes and edges in

the graph. After the addition, expansion rules can subsequently be fired for the newly added nodes, edges and their labels.

In order to update a completion graph upon the addition of a type assertion, $x{:}C$, the approach first checks if the individual exists in the completion graph. If $x \notin \mathcal{V}$, then x is added to $\mathcal{V}$. Then C is added to $\mathcal{L}(x)$ if it does not already occur in $\mathcal{L}(x)$. If an individual inequality relation, $x \neq y$, is added to the KB, the algorithm checks if $x \in \mathcal{V}$ and $y \in \mathcal{V}$. If either does not exists, then they are added to the graph. Additionally, if the $\langle x, y\rangle \notin x \dot{\neq} y$, then it is added. Alternatively, if an individual equality relation, $x = y$, is added to the KB, the approach checks if $x \in \mathcal{V}$ and $y \in \mathcal{V}$. If either does not exists, then they are added to the graph. Additionally, x and y are merged. Lastly, if a role, $\langle x, y\rangle{:}R$, is added to the KB and $\langle x, y\rangle \notin \mathcal{E}$, then $\langle x, y\rangle$ is added to $\mathcal{E}$ and R is added to $\mathcal{L}(\langle x, y\rangle)$. If however, $\langle x, y\rangle \in \mathcal{E}$ but $R \notin \mathcal{L}(\langle x, y\rangle)$, then R is added to $\mathcal{L}(\langle x, y\rangle)$.

After the graph has been updated, the completion rules must be reapplied to the updated completion graph, as the update may cause additional expansion rules to be fired. We note here that if there has previously been a deletion update, then previously explored branches which had a clash must be re-explored as the deletion could have removed the axiom which caused the clash.

4.2 ABox Deletions

When updating a completion graph under ABox axiom removals, *components* (nodes, edges, labels, etc.) in the graph that correspond to the removed axiom cannot be simply removed. This is because as completion rules are applied, newly added portions of the graph are dependent on the presence of the original axioms in the KB. By deleting an ABox assertion, components of the graph that are dependent on that assertion need to be updated as well.

In order to account for this, we propose using axiom pinpointing [2,15,16], which tracks the dependencies of completion graph components on original source axioms from the ontology through the tableau expansion process. More specifically, the application of the expansion rules triggers a set of *events*, denoted EV, that change the state of the completion graph, or the flow of the algorithm. In [15,16] a set of change events is defined, which include the following:

- Add $(C, \mathcal{L}(x))$ represents the action of adding a concept C to the label of a node x, i.e. the operation $\mathcal{L}(x) \leftarrow \mathcal{L}(x) \cup \{C\}$.
- Add $(R, \mathcal{L}(\langle x, y\rangle))$ represents the addition of a role R to the label of an edge $\langle x, y\rangle$, i.e., $\mathcal{L}(\langle x, y\rangle) \leftarrow \mathcal{L}(\langle x, y\rangle) \cup \{R\}$.
- x E y is the action of *merging* the nodes x, y. A detailed description of the *merge* operation can be found in [14].
- x NE y stands for the addition of an inequality relation between two nodes x,y i.e. $\dot{\neq} \leftarrow \dot{\neq} \cup \{\langle x, y\rangle\}$.

In order to record the changes on the completion graph, [15,16] introduces a *tracing function*, which keeps track of the asserted axioms responsible for changes to occur. The *tracing function*, τ, maps each event $e \in EV$ to a set of sets, each set containing

a fragment of the KB that cause the event to occur. This tracing function is maintained throughout the application of tableau expansion rules.

For purpose of this work, the original set of *change events* has been extended [10] to include all possible events that can occur during the application of expansion rules. The extension of events includes the following additional *events*:

- Add $(x, \mathcal{V})$ represents the action of adding a node x to the vertex set, $\mathcal{V}$, a completion graph, i.e. the operation $\mathcal{V} \leftarrow \mathcal{V} \cup \{x\}$.
- Add $(\langle x, y\rangle, \mathcal{E})$ represents the action of adding a edge $\langle x, y\rangle$ to the edge set, $\mathcal{E}$, in a completion graph, i.e. the operation $\mathcal{E} \leftarrow \mathcal{E} \cup \{\langle x, y\rangle\}$.
- Remove $(x, \mathcal{V})$ represents the action of removing a node x from the vertex set, $\mathcal{V}$, a completion graph, i.e. the operation $\mathcal{V} \leftarrow \mathcal{V} \setminus \{x\}$.
- Remove $(\langle x, y\rangle, \mathcal{E})$ represents the action of removing an edge $\langle x, y\rangle$ from the edge set, $\mathcal{E}$, in a completion graph, i.e. the operation $\mathcal{E} \leftarrow \mathcal{E} \setminus \{\langle x, y\rangle\}$.
- Remove $(C, \mathcal{L}(x))$ represents the action of removal a concept C from the label of a node x, i.e. the operation $\mathcal{L}(x) \leftarrow \mathcal{L}(x) \setminus \{C\}$.
- Remove $(R, \mathcal{L}(\langle x, y\rangle))$ represents the removal of a role R from the label of an edge $\langle x, y\rangle$, i.e., $\mathcal{L}(\langle x, y\rangle) \leftarrow \mathcal{L}(\langle x, y\rangle) \setminus \{R\}$.
- Remove x E y represents the action of undoing a previous *merge* between the nodes x, y. A detailed description of the *merge* operation can be found in [14].
- Remove x NE y represents the removal of an inequality relation between two nodes x,y, i.e. $\dot{\neq} \leftarrow \dot{\neq} \setminus \{\langle x, y\rangle\}$

Additionally, the tracing function maintenance through expansion rule application has been extended [10]. Although the tracing extension has been provided for $\mathcal{SHOIQ}$ [14], it is trivial to see how they are applied to $\mathcal{SHIQ}$ and $\mathcal{SHOQ}$ completion strategies. Further details can be found in [10].

The general idea is to utilize axiom tracing in order to track the dependencies of parts of the completion graph, so that the effects of ABox assertions being removed can be *rolled-back*. We note here that portions of the completion graph can be introduced by multiple (explicitly) asserted axioms, however each concept or role name in a label will only occur once. Therefore, axiom traces must be maintained for *each* asserted axiom sets that can potentially introduce a particular concept or role name in a label.

To clarify, consider the following KB, $\mathsf{K} = \{1.\ a{:}C \sqcap D,\ 2.\ D \sqsubseteq B\}$ (axiom numbers are provided for tracing purposes). After the completion graph is initially created for this KB, the node in the graph corresponding to a would have a variety of concept names in $\mathcal{L}(a)$, including D with an axiom trace of $\{1\}$ (this would be added from the $\sqcap$-rule) and B with an axiom trace of $\{1, 2\}$ (this would be added from the $unfolding$-rule). Next consider an incremental update of (Add $a{:}D$); since there already exists a trace for D, another axiom set is added to its trace. Therefore the axiom traces are sets of traces, as defined in [10,15,16]. In this case the axiom trace for $D \in \mathcal{L}(a)$ would be $\{\{1\}, \{3\}\}$, where $\{3\}$ corresponds to the added assertoin. We note here that in [15,16], the tableau algorithm runs to saturation (i.e, it continues applying all expansion rules until no rules are applicable, even if a clash is detected). However for purpose of this work, we use the normal tableau termination procedure, in which the algorithm proceeds until either

all completion graphs are closed or one complete and clash-free completion graph is found. We additionally note that this does not affect the tracing procedure.

In general, the update algorithm for deletion works as follows. When an ABox axiom is removed, the algorithm performs a lookup in the graph for all *change events* whose axiom traces include the axiom number of the deleted axiom. These events are *rolled-back* if and only if their axiom trace sets only include sets which contain the deleted axiom, possibly among other axioms. By *roll-back* we refer to simply undoing (the inverse) the event (e.g., rolling back the event $Add(x, \mathcal{V})$ would be the process of removing x from $\mathcal{V}$). If there exists additional axiom traces for that particular event that do not include the removed axiom, then only the sets including the removed axiom are deleted from the axiom trace set; in this situation the actual event is not *rolled-back*. This holds as there still exists support for that particular event.

As in the approach for additions, the completion rules must be reapplied to the updated completion graph as it is possible for the graph to be incomplete. Axiom tracing additionally requires a slight modification to the update approach for ABox additions in order to maintain axiom traces. For example, in the case that a individual type assertion is added to the KB, the algorithm must add a new tracing set to the axiom trace for the affected components (this set will consist of the new axiom number).

4.3 Update Algorithm

We now define the update algorithm $UPDATE(G, \alpha)$, which takes as input a completion graph G (for a knowledge base K) and an update α and returns a new completion graph; the algorithm is shown in Figure 1. Note that τ is the tracing function and $deps$ (dependents) is the inverted tracing function index (asserted axiom to a set of change events).

Now we provide the correctness proof for update algorithm under ABox additions. We first make the following observation: due to the *generating* tableau expansion rules, namely the $\exists - rule$ and the $\geq -rule$, new individuals could have been introduced to the graph that would not have been added in the completion graph if it were built from scratch; therefore it cannot simply be shown that $UPDATE(G, \alpha)$ will obtain a completion graph that could have been built if we applied the expansion rules to the updated KB (explicit ABox and TBox assertions). For example, this is evident if there is an addition $b{:}C$ to an ABox that consists of $a{:}\exists R.C$ and $\langle a, b\rangle{:}R$. If the completion graph where built from scratch, the $\exists$-rule would be blocked for the node with label $\exists R.C$ as there already exists a R-neighbor b (i.e., $R \in \mathcal{L}(b)$). In contrast, in $UPDATE(G, \alpha)$ the $\exists - rule$ would have already fired prior to the addition.

Theorem 1. *Under ABox additions, $UPDATE(G, \alpha)$ is sound, complete, and terminating.*

Proof Sketch

It is obvious that every graph generated from a subpart of a KB is a possible state of the completion graph for the KB, though it is potentially incomplete as more rules can fire. In $UPDATE(G, \alpha)$, the newly induced structure from the syntactic update is added to the graph, as it would've existed if the completion graph for $\mathsf{K} \cup \{\alpha\}$ were built from

function UPDATE(G, α)
  if α is an *addition* **then**
    if α is a $x \neq y$ *or* $x = y$ **then**
      let op be the operation, such that $op \in \{=, \neq\}$
      if $x \notin \mathcal{V}$ **then**
        $\mathcal{V} \leftarrow \mathcal{V} \cup \{x\}$
      if $y \notin \mathcal{V}$ **then**
        $\mathcal{V} \leftarrow \mathcal{V} \cup \{y\}$
      if op is $\neq$ **then**
        if $\langle x, y\rangle \notin x \dot{\neq} y$ **then** add it
      if op is = **then**
        Merge x and y
      $\tau(x\ op\ y) \leftarrow \tau(x\ op\ y) \cup \{\{\alpha\}\}$
      $\tau(\mathsf{Add}(x, \mathcal{V})) \leftarrow \tau(\mathsf{Add}(x, \mathcal{V})) \cup \{\{\alpha\}\}$
      $\tau(\mathsf{Add}(y, \mathcal{V})) \leftarrow \tau(\mathsf{Add}(y, \mathcal{V})) \cup \{\{\alpha\}\}$
      $deps(\alpha) \leftarrow deps(\alpha) \cup \{\{x\ op\ y\}, \{\mathsf{Add}(x, \mathcal{V})\}, \{\mathsf{Add}(x, \mathcal{V})\}\}$
    else if α is a *individual type addition*, $x{:}C$ **then**
      if $x \notin \mathcal{V}$ **then**
        $\mathcal{V} \leftarrow \mathcal{V} \cup \{x\}$
      $\mathcal{L}(x) \leftarrow \mathcal{L}(x) \cup \{C\}$
      $\tau(\mathsf{Add}(x, \mathcal{V})) \leftarrow \tau(\mathsf{Add}(x, \mathcal{V})) \cup \{\{\alpha\}\}$
      $\tau(\mathsf{Add}(C, \mathcal{L}(x))) \leftarrow \tau(\mathsf{Add}(C, \mathcal{L}(x))) \cup \{\{\alpha\}\}$
      $deps(\alpha) \leftarrow deps(\alpha) \cup \{\{Add(x, \mathcal{V})\}, \{Add(C, \mathcal{L}(x))\}\}$
    else if α is a *role assertion addition*, $\langle x, y\rangle{:}R$ **then**
      if $\langle x, y\rangle \notin \mathcal{E}$ **then**
        $\mathcal{E} \leftarrow \mathcal{E} \cup \{\langle x, y\rangle\}$
      $\mathcal{L}(\langle x, y\rangle) \leftarrow \mathcal{L}(e) \cup \{R\}$
      $\tau(\mathsf{Add}(x, \mathcal{V})) \leftarrow \tau(\mathsf{Add}(x, \mathcal{V})) \cup \{\{\alpha\}\}$
      $\tau(\mathsf{Add}(y, \mathcal{V})) \leftarrow \tau(\mathsf{Add}(y, \mathcal{V})) \cup \{\{\alpha\}\}$
      $\tau(\mathsf{Add}(\langle x, y\rangle, \mathcal{E})) \leftarrow \tau(\mathsf{Add}(\langle x, y\rangle, \mathcal{E})) \cup \{\{\alpha\}\}$
      $\tau(\mathsf{Add}(R, \mathcal{L}(\langle x, y\rangle))) \leftarrow \tau(\mathsf{Add}(R, \mathcal{L}(\langle x, y\rangle))) \cup \{\{\alpha\}\}$
      $deps(\alpha) \leftarrow deps(\alpha) \cup \{\{Add(x, \mathcal{V})\}, \{Add(y, \mathcal{V})\}, \{Add(\langle x, y\rangle, \mathcal{E})\}, \{Add(R, \mathcal{L}(\langle x, y\rangle))\}\}$
    Apply expansion rules to G
    if there is a clash **then**
      Perform backjumping
  else if α is a *deletion* **then**
    $events \leftarrow deps(\alpha)$
    $deps(\alpha) \leftarrow \emptyset$
    for each $e \in events$ **do**
      $traces \leftarrow \tau(e)$
      for each $t \in traces$ **do**
        if $\alpha \in t$ **do**
          $traces \leftarrow traces \setminus t$
      if $traces = \emptyset$ **do**
        roll-back e
      $\tau(e) \leftarrow traces$
    Apply expansion rules to G
    if there is a clash **then**
      Perform backjumping
  return G

Fig. 1. Pseudo-code of update procedure for $\mathcal{SHOQ}$ and $\mathcal{SHIQ}$ KBs

scratch. From our discussion earlier, it is clear the initially updated completion graph can contain clashes; however the completion rules are then re-fired. It can therefore be shown that the resulting completion graph must correspond to a model if and only if at least one exists, as if it were the case that the completion graph did not correspond to a model (via some clash), then the soundness or completeness of [12,13] would be contradicted (i.e., if a model exists yet the initially updated completion graph contains a clash, it would be removed by shrinking expansion rules or backjumping). Further, if a non-deterministic choice were taken and backjumping occurs, the newly added

structures imposed by the update will remain, as they are explicitly asserted. It is clear that if some model exists, the tableau algorithm then proceeds as usual and a closed, clash-free completion graph is found. Therefore, it can be shown that under ABox additions, $UPDATE(G, \alpha)$ is sound, complete, and terminating. □

Additionally, the correctness proof for the update algorithm under ABox deletions is presented.

Theorem 2. *Under ABox deletions, $UPDATE(G, \alpha)$ is sound, complete, and terminating.*

Proof Sketch

As shown in [16,15,10], the tracing function captures all *change events* in G that were caused in part by α. It can be shown that by undoing all events that are *only* reliant on an axiom trace that includes α, G is effectively *rolled-back* to a state in which the effects of the rule firings caused by α are removed. This holds because the tracing algorithm is shown to be complete [16,15,10], thus *all* necessary components will be rolled-back. Because $UPDATE(G, \alpha)$ performs this rollback, it can be shown that the updated completion graph is a possible intermediate state of the a completion graph for the KB after the deletion. While this graph is potentially incomplete (e.g., due to blocking), reapplying expansion rules guarantees (by [12,13]) the algorithm will arrive at some completion graph that could be obtained by simply applying the completion rules to $\mathsf{K} \cup \{\alpha\}$. It can therefore be shown that under ABox deletions, $UPDATE(G, \alpha)$ is sound, complete, and terminating. □

5 Implementation and Evaluation

We have implemented the approach presented in this paper in an open source OWL-DL reasoner, Pellet [22]. In order to evaluate the algorithm, we have performed an emperical evaluation using two different KBs with large ABoxes - the Lehigh University Benchmark (LUBM)[2] and AKT Reference Ontology [3].

For the LUBM test case, three experiments were run over three different KBs consisting of one, two, and lastly three universities created by the LUBM dataset generator. First an initial consistency check was performed and then in each test a random update was selected which was used to update the KB. In the regular version of the reasoner, the cached completion graph was discarded, while in the optimized reasoner the update algorithm was utilized. For each KB size, varying sized additions and deletions were randomly selected from the dataset. Update sizes include single axiom, twenty-five axioms, and fifty axioms (individuals and/or roles). Each test was averaged over twenty-five times iterations. Expressivity and KB statistics are provided in Table 1.

Results for additions and deletions in the LUBM test are presented in Figures 2 and 3 (timing results are shown in milliseconds and the scale is logarithmic). We note

[2] LUBM Ontology: http://swat.cse.lehigh.edu/projects/lubm/
[3] AKT Ontology: http://www.aktors.org/publications/ontology/

Table 1. LUBM and AKT Portal Dataset Statistics

Name	Classes / Properties / Individuals / Assertions	Expressivity
LUBM-1 Univ	43 / 32 / 18,257 / 97,281	$\mathcal{SHI}$
LUBM-2 Univ	43 / 32 / 47,896 / 254,860	$\mathcal{SHI}$
LUBM-3 Univ	43 / 32 / 55,110 / 295,728	$\mathcal{SHI}$
AKT-1	160 / 152 / 16,505 / 70,948	$\mathcal{SHIF}$
AKT-2	160 / 152 / 32,926 / 143,334	$\mathcal{SHIF}$

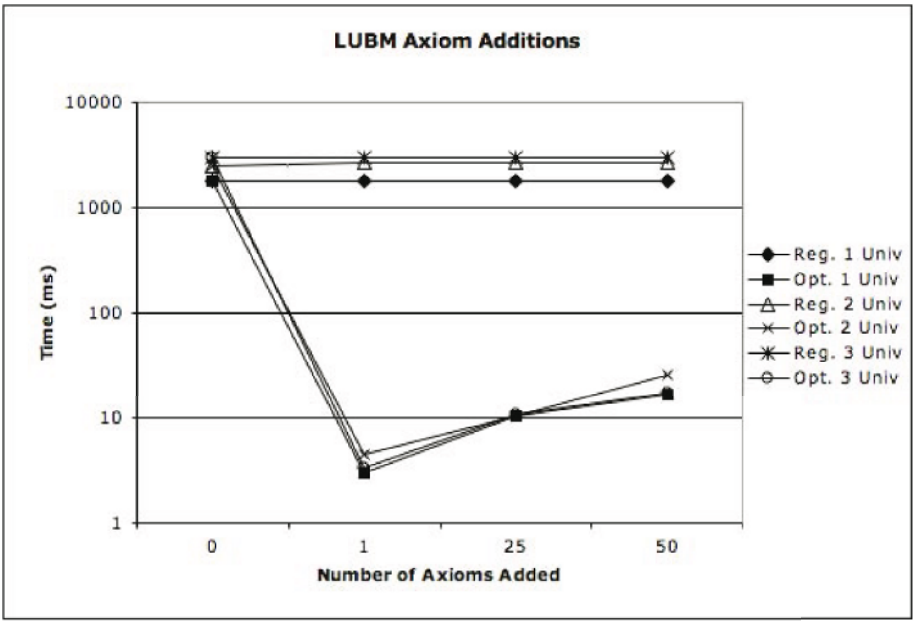


Fig. 2. Addition Updates of LUBM datasets

that the '0' axiom value represents the initial consistency check. In both versions of the reasoner, initial consistency checks are comparable. However for both update types (additions and deletions), performance improvements ranging from one to three orders of magnitude are achieved under updates in the reasoner with the optimized update algorithm. This is due to the avoidance of re-firing of completion rules by maintaining the previous completion graph; therefore very few (if any in some cases) completion rules must be fired. It can also be observed that as the update size is increased, the performance of the update approach scales well. This provides direct empirical evidence for the effectiveness of the update algorithm.

In a second evaluation, two datasets[4] adhering to the AKT Reference ontology were used (statistics shown in Table 1). The tests were structured in the same manner as the LUBM test. Again, each test was performed twenty-five times and the results are

[4] Hyphen-REA: http://www.hyphen.info/rdf/hero_complete.zip

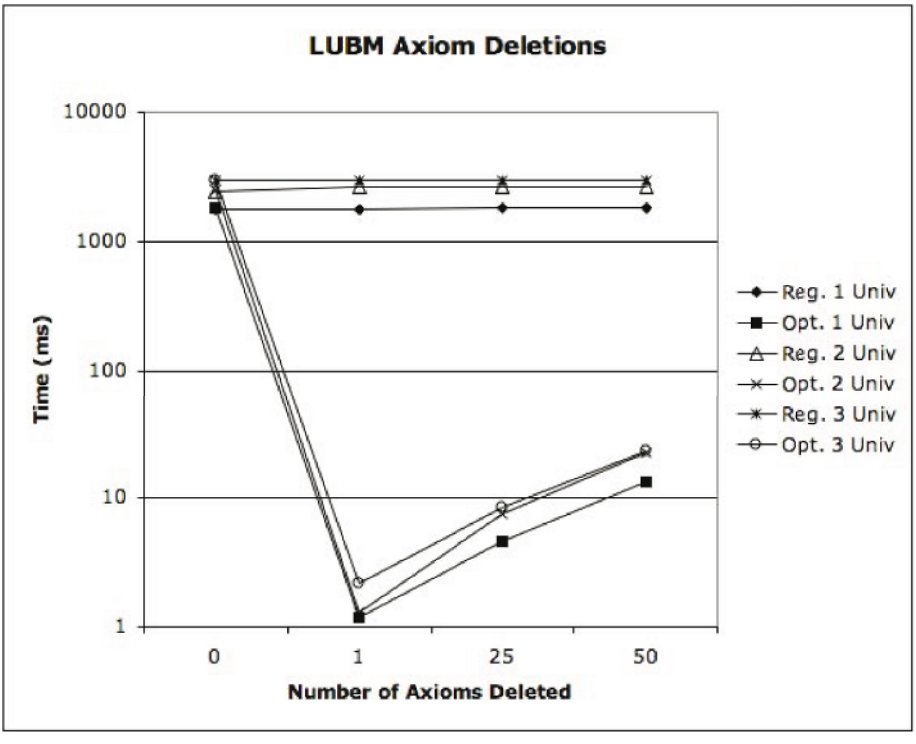


Fig. 3. Deletion Updates of LUBM datasets

averaged over these iterations. All timings are in milliseconds and the scale is logarithmic. Similar to the LUBM test, update performance is improved between one to three orders of magnitude (as shown in Figures 4 and 5). It is interesting to observe that the performance of the deletion updates is slightly better in the LUBM test cases for larger sized updates. This is primarily due to the increased complexity of the AKT Reference Ontology; therefore, a larger number of expansion rules are applied after the update. However, the update algorithm greatly outperforms the regular reasoner again demonstrating the effectiveness and overall impact of the update approach.

6 Discussion and Future Work

One limitation of the approach presented in the work is related to potential overhead introduced by the algorithm, specifically related to tracing. However, we point out here that in [16] the tracing approach was shown to introduce small memory overhead and only marginally increase the running time of the normal reasoning procedure. For example, in the Tambis [5] ontology, tracing introduced only 56ms to the running time and 3.65mb memory overheard [16]. Therefore, we feel the approach is acceptable due to the observed performance improvements.

[5] Tambis ontology: http://www.cs.man.ac.uk/ horrocks/OWL/Ontologies/tambis-full.owl

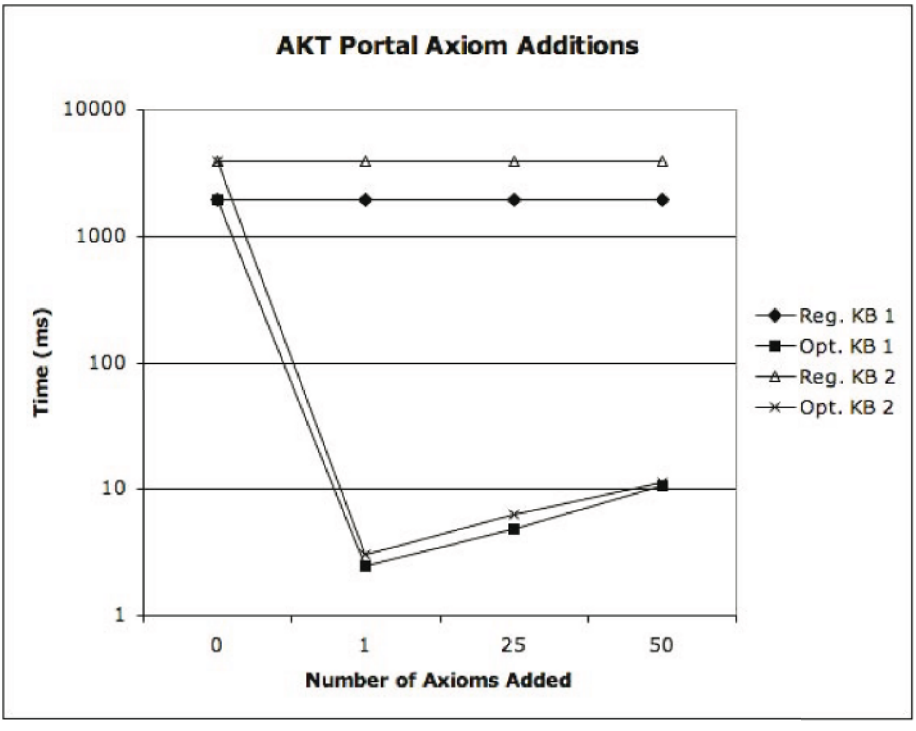


Fig. 4. Addition Updates of AKT datasets

The approach presented in this work is applicable to the Description Logics $\mathcal{SHOQ}$ and $\mathcal{SHIQ}$; this is primarily due to fact that there is no expansion rule ordering imposed by the tableau algorithms [12,13]. We are currently working to extend the approach to $\mathcal{SHOIQ}$ (and therefore all of OWL-DL), which will be addressed in future work.

While we achieved dramatic results for consistency checking under syntactic ABox updates, one may wish to update TBox axioms as well. This presents the additional issue of rolling back through pre-processing steps, such as absorption. We are currently investigating this problem and plan to address it in future work.

In the current approach, if the KB is inconsistent after the update, nothing is done to resolve the inconsistency. As future work, we are working towards developing a revision algorithm for OWL-DL KBs; with such a technique, inconsistencies resulting after the update would be resolved using a revision operator.

This work only directly addresses consistency checking under ABox changes; however, standard reasoning tasks in DL reasoners, including classification, realization, and query answering are all reducible to KB consistency checking [3]. Currently, we are investigating the utility of the update algorithm for generalized reasoning services. We note here that initial results on leveraging the approach presented in this paper for continuous conjunctive query answering demonstrates orders of magnitude improvements in performance.

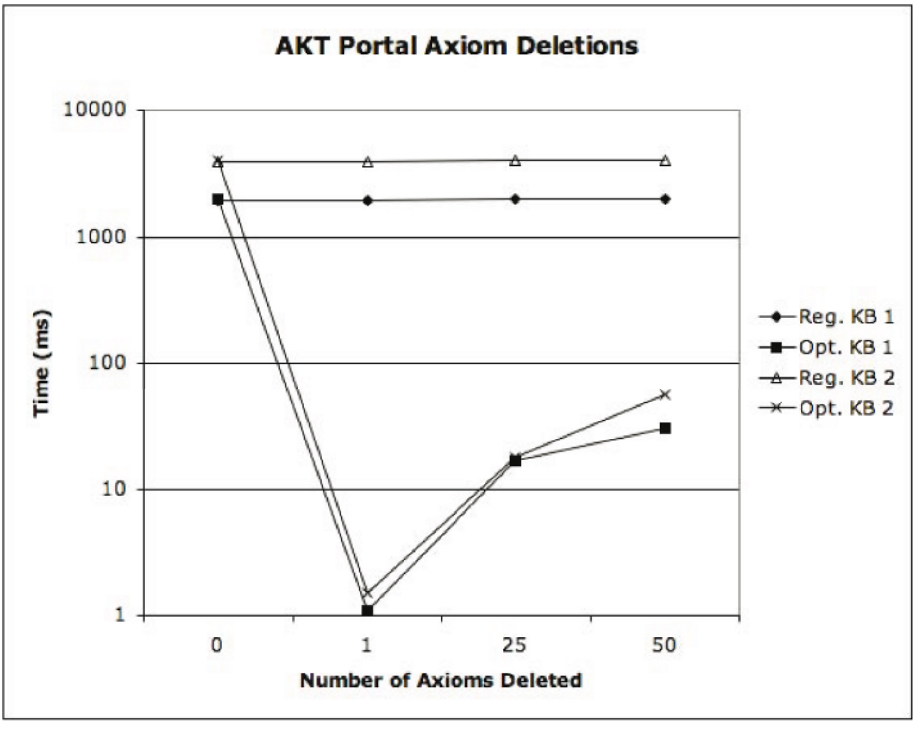


Fig. 5. Deletion Updates of AKT datasets

7 Related Work

To our knowledge there has been no previous work in DL reasoning algorithms for incremental maintenance of completion graphs. We do note that this work can be paralleled to view maintenance [7] and truth maintenance [4]; however we deal with a more expressive logic and a different proof theory.

In this work, we have leveraged and extended previous work in axiom tracing [2,15,16]. [2] introduces the notion of axiom pinpointing for the purpose of computing extensions of default theories. [15,16] extends [2] to support a more expressive logic. As discussed earlier, we have further extended [15,16] as further effects of axioms were needed to be identified.

Recently, there has been interest in specifying formal update semantics for descriptions logics [19,23]. Additionally, [5,6] has investigated applying traditional AGM belief revision [1] theory to DL knowledge bases; the authors show however, that in certain expressive DLs (including those considered in this work) the AGM postulates for contraction cannot be satisfied. This finding does not impact this work, as we assume syntactic updates of asserted axioms. Further, this work is independent of belief revision and formal update semantics as we are concerned with maintaining the internal state of the reasoner.

8 Conclusion

Current Description Logic reasoners are traditionally oriented toward providing reasoning services for relatively static ontologies. However, there are numerous use cases in which the ontology itself is in flux, requiring frequent updates. This includes prominent Web services frameworks (e.g., OWL-S), in which Web ontologies are used for service discovery and matchmaking. In such settings devices register or deregister their descriptions quite rapidly. Additionally, Semantic Web portals often allow content authors to modify or extend the ontologies which organize their site structure or page content. Lastly, there has been recent interest in utilizing DL reasoning for the purpose of matching published content with subscription requests [8, 9, 26, 17]. When disseminating time sensitive information efficient reasoning for frequently changing KBs will be critical in achieving practical DL-based approaches.

While there exists such use cases for reasoning under changing data, current DL reasoning algorithms have been developed considering relatively static knowledge bases. In this paper, we have presented an algorithm for updating tableau completion graphs for the Description Logics $\mathcal{SHIQ}(\mathcal{D})$ and $\mathcal{SHOQ}(\mathcal{D})$ under both the addition and removal of ABox assertions, providing a critical step towards reasoning procedures for fluctuating or streaming data. We have provided an empirical analysis of the algorithm through an experimental implementation in the Pellet reasoner, in which our initial results are very promising as they demonstrate orders of magnitude performance improvement.

Acknowledgments

This work was supported in part by grants from Fujitsu, Lockheed Martin, NTT Corp., Kevric Corp., SAIC, the National Science Foundation, the National Geospatial Intelligence Agency, DARPA, US Army Research Laboratory, and NIST. We would also like to thank Aditya Kalyanpur, Yarden Katz, and Vladimir Kolovski for all of their contributions to this work.

References

1. Carlos E. Alchourrón, Peter Gärdenfors, and David Makinson. On the logic of theory change: Partial meet contraction and revision functions. *Journal of Symbolic Logic*, 50(2):510–530, 1985.
2. F. Baader and B. Hollunder. Embedding defaults into terminological representation systems. *J. Automated Reasoning*, 14:149–180, 1995.
3. F. Baader and W. Nutt. Basic description logics. In Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors, *The Description Logic Handbook: Theory, Implementation, and Applications*, pages 43–95. Cambridge University Press, 2003.
4. Jon Doyle. A truth maintenance system. *Artificial Intelligence*, 1979.
5. G. Flouris, D. Plexousakis, and G. Antoniou. On applying the agm theory to dls and owl. In *4th International Semantic Web Conference (ISWC 2005)*, 2005.

6. G. Flouris, D. Plexousakis, and G. Antoniou. Updating description logic using the agm theory. In *7th International Symposium on Logical Formalizations of Commonsense Reasoning*, 2005.
7. A. Gupta and I. Mumick. Materialized views: Techniques, implementation, and applications. In *MIT press*, 1999.
8. V. Haarslev and R. Moller. Description logic systems with concrete domains: Applications for the semantic web. In *In Int. Workshop on KR meets Databases, 2003.*, 2003.
9. V. Haarslev and R. Möller. Incremental query answering for implementing document retrieval services. In *Proceedings of the International Workshop on Description Logics (DL-2003), Rome, Italy, September 5-7*, pages 85–94, 2003.
10. Christian Halaschek-Wiener, Aditya Kalyanpur, and Bijan Parsia. Extending tableau tracing for abox updates. In *UMIACS Tech Report*, 2006. http://www.mindswap.org/papers/2006/aboxTracingTR2006.pdf.
11. I. Horrocks. Implementation and optimisation techniques. In Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors, *The Description Logic Handbook: Theory, Implementation, and Applications*, pages 313–355. Cambridge University Press, 2003.
12. I. Horrocks and U. Sattler. Ontology reasoning in the SHOQ(D) description logic. In B. Nebel, editor, *Proc. of the 17th Int. Joint Conf. on Artificial Intelligence (IJCAI 2001)*, pages 199–204. Morgan Kaufmann, 2001.
13. I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for expressive description logics. In *Proc. of the 6th Int. Conference on Logic for Programming and Automated Reasoning (LPAR'99)*, number 1705 in Lecture Notes in Artificial Intelligence, pages 161–180. Springer-Verlag, 1999.
14. Ian Horrocks and Ulrike Sattler. A tableaux decision procedure for SHOIQ. In *Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI 2005)*. Morgan Kaufman, 2005.
15. Aditya Kalyanpur. Debugging and repair of owl ontologies. In *Ph.D. Dissertation, University of Maryland, College Park*. http://www.mindswap.org/papers/2006/AdityaThesis-DebuggingOWL.pdf.
16. Aditya Kalyanpur, Bijan Parsia, Evren Sirin, and James Hendler. Debugging unsatisfiable classes in owl ontologies. In *Journal of Web Semantics - Special Issue of the Semantic Web Track of WWW2005*, 2005.
17. L. Li and I. Horrocks. A software framework for matchmaking based on semantic web technology, 2003.
18. T. Liebig and O. Noppens. Ontotrack: Combining browsing and editing with reasoning and explaining for owl lite ontologies. In *Proceedings of the 3rd International Semantic Web Conference (ISWC) 2004*, Japan, November 2004.
19. H. Liu, C. Lutz, M. Milicic, and F. Wolter. Updating description logic aboxes. In *International Conference of Principles of Knowledge Representation and Reasoning(KR)*, 2006.
20. Bernhard Nebel. Base revision operations and schemes: Semantics, representation, and complexity, 1994.
21. Massimo Paolucci, Takahiro Kawamura, Terry R. Payne, and Katia Sycara. Semantic matching of web services capabilities. In *The First International Semantic Web Conference*, 2002.
22. Bijan Parsia and Evren Sirin. Pellet: An owl dl reasoner. In *Third International Semantic Web Conference - Poster*, 2004.
23. Mathieu Roger, Ana Simonet, and Michel Simonet. Toward updates in description logics. In *International Workshop on Knowledge Representation meets Databases*, 2002.
24. Evren Sirin, Bijan Parsia, and James Hendler. Composition-driven filtering and selection of semantic web services. *IEEE Intelligent Systems*, 19(4):42–49, 2004.

25. Katia Sycara, Massimo Paolucci, Anupriya Ankolekar, and Naveen Srinivasan. Automated discovery, interaction and composition of semantic web services. *Journal of Web Semantics*, 1(1):27–46, 2003.
26. Michael Uschold, Peter Clark, Fred Dickey, Casey Fung, Sonia Smith, Stephen Uczekaj Michael Wilke, Sean Bechhofer, and Ian Horrocks. A semantic infosphere. In Dieter Fensel, Katia Sycara, and John Mylopoulos, editors, *Proc. of the 2003 International Semantic Web Conference (ISWC 2003)*, number 2870 in Lecture Notes in Computer Science, pages 882–896. Springer, 2003.

From Folksologies to Ontologies: How the Twain Meet

Peter Spyns, Aldo de Moor, Jan Vandenbussche, and Robert Meersman

Vrije Universiteit Brussel - STAR Lab,
Pleinlaan 2 Gebouw G-10, B-1050 Brussel - Belgium
{Peter.Spyns, Aldo.De.Moor, javdbuss, Robert.Meersman}@vub.ac.be

Abstract. Ontologies are instruments for capturing and using formal semantics, and are often the result of a "central committee controlled" style of working. A new trend on the Web is the increasing popularity of folksologies in the form of social bookmarking sites. Folksologies provide informal semantics and can be created and adopted by anybody anytime anywhere on the Internet. Shared meaning in a folksology emerges through the use of tags that are used to bookmark web pages, their usage frequency being considered a reliable indicator of their usefulness and acceptance.

Rather than choosing for either ontologies or folksologies, hybrid emergent semantics systems are needed that combine elements of both perspectives, depending on the particular application. There is a need to analyse the larger picture (including the full range of semantics' functionalities in their context of use.

In this paper, we outline a number of key design characteristics of emergent semantics systems (ESS). We examine the functionalities of two existing examples of well-known ESSs: del.icio.us and Piggy Bank. Using the results of this comparison, we introduce DogmaBank as a proof of concept implementation of a next-generation ESS that introduces a more advanced combination of lexical and conceptual emergent semantics functionalities.

1 Introduction

Many definitions of ontologies exist [18, 20, 58, 57]. We prefer the one from Guarino [22, p.7][1]. Combining various definitions, an ontology can be seen as a formal, shared, explicit but partial specification of the commonly agreed upon intended meaning of a conceptualisation. With some form of simplification, one could say that an ontology is like a dictionary with unambiguous meaning defining entries linked by many formal relationships. A taxonomy on the other hand has only a single structuring relationship between its constituents - e.g., is_a_kind_of, is_a_part_of.

The term 'folksology' has been coined by Stefano Mazzocchi [36] in analogy with 'folksonomy', created by Thomas Vander Wal [2]. It basically means that ontologies resp.

[1] An ontology is a logical theory accounting for the intended meaning of a formal vocabulary, i.e. its ontological commitment to a particular conceptualisation of the world. The intented models of a logical language using such a vocabulary are constrained by its ontological commitment. An ontology indirectly reflects this commitment (and the underlying conceptualisation) by approximating these intended models.

[2] http://en.wikipedia.org/wiki/Folksonomy

R. Meersman, Z. Tari et al. (Eds.): OTM 2006, LNCS 4275, pp. 738–755, 2006.

taxonomies emerge on the fly thanks to individuals who autonomously create and use any tag they like to annotate any web page they deem worthwhile[3]. The tag as well as the tagged web page (URL) is published in a repository (= a social bookmarking site). Anybody can also re-use any tag from anybody else if it has been published on the social bookmarking site. Within what is called the Web2.0 community these social bookmarking sites are becoming very popular [25]. Flickr[4], an image sharing site, and del.icio.us[5], a bookmark collection site are seen as the two most successful representatives of such sites. In the academic world, there is, for instance, Bibsonomy[6], a site to share bibliographic references.

As a consequence, some (e.g., Shirky [50]) even hail folksologies as the best way to create meaning on the Internet as opposed to ontologies as defined in the tradition of Gruber [18] and Guarino [20]. Note that an ontology by definition is a social construct as it constitutes a shared agreement resulting from some form of meaning negotiation by the various domain stakeholders, although this social aspect in practice is often at best assumed by ontology engineers - in many cases acting on their own.

For those familiar with the world of database schema modelling, the current social bookmarking services closely correspond, albeit it in a more global and networked environment, to the situation in which a database schema modeller invents his own table and column names and expects other modellers to understand them as such. The shared meaning, however, only resides in the brain of each individual that uses and tries to understand such a column name (or tag in the folksology case), as it basically equals a natural language word. As soon as multiple database applications are supposed to interoperate with each other (federated databases, data warehouse, ...), problems arise as more often than not the meaning of the labels has not been made explicit. Well-known issues like ambiguity, cryptic abbreviations, multilinguality etc. inhibit database interoperability and are bound to also manifest themselves in the area of folksologies as the many people involved in creating and using the tags have different mental models.

Much research effort in the database community has been dedicated to schema matching, schema merging, DB-mediators (e.g., [3, 28]) and so on to solve or at least reduce interoperability problems. Ontologies are recognised by many as instrumental for solving the interoperability problem as they are basically a meaning repository for labels and, in principle, independent of any specific application [38].

1.1 Bridging the Gap

Folksologies and ontologies are not two opposite ways of organising a (more) meaningful Internet, but rather constitute two ends of a range. The strenghts of one approach can offset the weaknesses of the other. Shirky distinguishes some characteristics that makes

[3] There is a lot of confusion about and inconsistent use of both terms. Folksonomies relate to folksologies in the same way as taxonomies relate to ontologies, i.e. a single vs. multiple types of relationships between the concepts. Their commonality, however, is the focus on the social process leading to informal semantics. In the remainder of this paper we will use the term 'folksology', since that is the clearest counterpart of ontology.

[4] http://flickr.com/

[5] http://del.icio.us

[6] www.bibsonomy.org

Table 1. ontologies vs. folksologies: their characteristics and participants [50]

characteristics	*ontology*	*folksology*
corpus	small	large
categories	formal	non formal
entities	stable and restricted	non stable and unrestricted
edges	clear	unclear
paticipants	*ontology*	*folksology*
users	coordinated and expert	uncoordinated and amateur
authority	authoritative sources of judgment	no authority
cataloguers	expert	naive

a domain more (or less) suited for an ontology to be built for it [50] [7]. In the same way, he contrasts the participants of ontologies and folksologies. Table 1 lists these two aspects.

Whereas Shirky overall favours folksologies for real-world applications, Gruber makes the case that ontologies have an important role to play as well in supporting the tagging process. This view is reflected in the TagOntology project, which is about identifying and formalizing a conceptualisation of the activity of tagging, and building technology that commits to the ontology at the semantic level [19].

Although the debate is ongoing about the pros and cons of folksologies versus ontologies, still little attention is paid to the next step: what do the *systems* look like that are going to be built on top of these approaches? We define *emergent semantics systems* as information systems that combine informal and formal semantics approaches (i.e. folksologies and ontologies) to optimally serve the evolving requirements of communities of human and machine information providers and users. To support our points, we have implemented a proof of concept ESS combining a social bookmarking tool with ontology technology. It should help to build bridges not only between these informal and formal semantics approaches, but also between the theoretical research and practical systems development communities. In this respect, this paper can be seen as a position paper complemented by a proof of concept implementation.

1.2 Structure of the Paper

The remainder of this paper is organised as follows. The next section treats the notion of emerging semantics theory and systems in more depth. In section 3, we outline six characteristics that need to be taken into account in the design of an ESS. In section 4, we illustrate our ideas on two existing Web2.0 systems (del.icio.us - section 4.1 - and Piggy Bank - section 4.2) as well as propose a new system for semantic

[7] Shirky sees an ontology mainly as a (library) classification system with, for instance, the Yahoo directory as its Internet version, which leads Gruber to state that "Shirky uses 'ontology' where he should have used 'taxonomy', and therefore he "misses the point [on ontologies] ... so beautifully [19]. See also [41] for a rebuttal of Shirky's opinions. Note that in general statements in the folksologies vs. ontologies debate represent individual positions that have not been submitted to scientific peer review.

annotation (section 4.4) based on the DOGMA ontology framework (section 4.3). We end the paper with a section on related and future work - section 5 - and a conclusion (section 6).

2 Emerging Semantics: From Theory to Systems

Research on the Semantic Web and ontologies posits that these technologies are meant to be used by *software agents* (rather than human agents) that offer and look for all kinds of services [5]. For these kinds of Semantic Web applications, a formal and explicit definition of meaning shared by these agents is prerequisite as they are expected to reason about or infer new knowledge. For a domain of discourse or application domain a conceptualisation is created and later on implemented as an ontology [22]. Consequently, many resources are spent focusing on defining (semantic) web (service) languages and (reasoning) formalisms. This in contrast to social bookmarking services that are more tailored towards humans interacting. As an implicit consequence, the Semantic Web seems to be primarily oriented towards business rather than social use. Many of the classical Semantic Web motivating examples involve humans, but through the use of software agents, merely as individual consumers or clients who book travels, buy books and CDs, use smart house-hold devices and enjoy health monitoring and personal scheduling services. Typical usage of current Web2.0 applications, in particular community building and social aspects, is less considered by the Semantic Web research community. On the other hand, the Semantic Web interoperability topics, crucial for the intelligent software agents that will ease our on-line life, are hardly valued (or even plainly critised e.g., [49]) by the Web2.0 community.

A novel topic in the Semantic Web domain is called "Emergent Semantics". Although the original theme addressed by Emergent Semantics was rather restricted to database query meaning transformation and preservation in a peer-to-peer setting [2], the meaning of the expression has been widened to the result of dynamic and distributed local processes during which new or additional semantics are obtained and applied on a global level [8]. Another application domain concerns multimedia annotation (e.g., [17]). Aberer et al. [1] discuss the following characteristics of emergent semantics. It concerns agreements that

1. are a semantic handshake protocol
2. emerge from negotiations
3. emerge from local interaction
4. are dynamic and self-referential approximations
5. induce semantic self-organisation

In that respect, folksonomy and folksology techniques and software facilitate the emergence of distributed semantics. The semantics emerge from the implicit but immediate feedback from the community in the form of usage frequencies of tags as well as the listing of tags per URL and URLs per tag. A tag creator can conform his usage of tags to the "average", and thus implicitly agrees on the meaning attributed to a tag.

[8] Conclusions of the scientific meeting of the IFIP WG 2.6 on Databases at 30/10/2005 - see [8].

Our overall position is that there will many emergent semantics systems consisting of multiple semantic webs and light-weight ontologies that are "valid" for a specific application or business domain, (professional) organisation and the like. These semantic webs will include social aspects in their creation, maintenance, and evaluation stages. Nowadays company-wide controlled vocabularies (e.g. within Boeing and General Motors), and even world-wide classification systems for a specific domain (e.g., the 10th edition of the International Classification of Diseases) exist, are permanently used and regularly updated. As ontologies are, said in a simplifying way, formal representations of such vocabularies and classifications, there is in principle no reason why the transition to the Semantic Web of these existing mechanisms of establishing meaning agreements should not work. On the other hand, ontology engineers should have the candour to admit that this transition phase has not yet been studied thoroughly.

In a nutshell, folksologies have to differentiate between natural language words and a language-independent artificially created label indicating a concept or sense, and ontologies have to adopt strategies and tools as used for social bookmarking sites to make the meaning of concepts spontaneously emerge and converge.

3 Characteristics of Emergent Semantics Systems

In this section, we describe six key characteristics of emergent semantics systems including human agents. The core issue is how informal and formal meaning emerges, is agreed upon and shared by a community, and is used subsequently for various, including unforeseen, scenarios on a global or potentially world-wide scale by human and software agents that can organise themselves in (virtual) communities or societies. The choice of characteristics has been inspired by contrasting essential features proposed by proponents of the Semantic Web and the Web2.0, respectively. For each characteristic, we sketch how the informal and formal semantics approaches do or could meet.

3.1 Natural Language Words vs. Language Independent Concept Labels

Folksologies are based on a very basic but implicit assumption: everybody speaks the same language - in practice US English. The meaning agreement happens unconsciously: individuals understand what a tag means since the tag is a single[9] plain US English word [10]. We've already mentioned synonyms and homonyms, but what with other languages ? E.g., is a Dutch speaking individual simply expected to use US English to conceptualise his small world ? What happens if somebody uses a wrong or inappropriate translation ? Maybe somebody doesn't understand the meaning of a tag ? A word lexicalises a concept that is an idea, notion or meaning (cf. the meaning triangle of Ogden and Richards [45]). There is an m:n mapping of words on concepts, i.e. one word expresses several meanings (homonym) or several words express the same meaning (synonym)[11]. In addition, according to the context and language, a concept can be

[9] Del.icio.us does not allow compound words - e.g., 'semantic web' is considered as two tags.

[10] We avoid to use 'term' as it means logical term on the one hand or technical term of a special purpose (natural) language on the other.

[11] See [34] for some "life examples" from Del.icio.us regarding ambiguity and synonyms.

expressed differently (e.g., fish is translated in Spanish by 'pez' or 'pescado' depending on whether or not the fish is still living ('pez') or on your plate to be eaten ('pescado') [12]. The point is that one should not use words to label concepts but rather unique identifiers that stand for language independent (or neutral [43]) concepts to which words are mapped [54]. Of course, this requires that the meaning uniquely identified by a tag is explicitly defined - cf. e.g., WordNet synsets. Meaning definitions can come from dictionaries, encyclopedias, company controlled vocabularies etc. An important consequence is that additional software support is needed to avoid that duplicate and redundant tags are created and to steer a tag user into selecting the most appropriate tag. Unfortunately, also ontology engineers often fail to make this fundamental difference between a word and a concept [43].

3.2 Informal vs. Formal Semantics

Folksologies suffer from the fact that they stay on the language level and thus are unable to cope with issues such as synonymy, notational variants and homonymy[13]. Ontologies (formal models) on the other hand are unable to "connect" to the real world as they are represented in some first order language [14, pp.30-31][14].

Informal and formal semantics can, for example, be aligned resulting in explicit meaning circumscriptions that are linked to a concept identifier or label (= the logical term) and that group various lexicalisations (= the language term) depending on the context and language [54]. A concept label is a unique identifier for a synset or sense (as is the case for (Euro)WordNet). Formal semantic restrictions (e.g., expressing uniqueness or mandatoriness) and reasoning rules (e.g., defining transitivity for a certain relationship) use the concept identifiers as logical terms. In most of the cases, a system internal (alpha-)numerical ID exists next to a more human-friendly pseudo-word string. The signature of an ontology is now explicitly rooted in "natural" semantics (e.g., by means of dictionary definitions), while at the same time meaning ambiguity is avoided (a tag is unambiguously linked to a specific sense). Thanks to the additional formal semantic restrictions and definitions, the fuzziness of informal natural language definitions is further reduced. Formal definitions are also needed for inferencing.

3.3 A Bag of Loose Words vs. a Well Structured Semantic Network

As already mentioned earlier, a folksonomy consists of words. In addition, there is a complete absence of any structuring relationship. Some (e.g., [34]) consider that the absence of a hierarchy to agree upon favours cooperation for social bookmarking. In fact, the term 'folksonomies' is semantically speaking too broad as there is no taxonomy

[12] Cf. also the difference in U.K. English between 'pig' and 'pork' (but in Dutch there is only the word 'varken').

[13] Simply adding an OWL "SameAs" or "DifferentFrom" statement, as suggested by Mazzocchi [36] is insufficient as it only marks an equivalence/difference between two language terms but does not provide any explicit definition of the meaning of the tag.

[14] Model-theoretic semantics does not pretend, and has no way to determine what certain words and statements "really" mean. (...) It [= model theoretic semantics] offers no help in making the connection between the model (the abstract structure) and the real world.

involved, but only a vocabulary (a bag of words) [15]. 'Folksabularies' (in analogy with vocabularies) would thus be a more accurate term. Relationships are a fundamental part of ontologies. A classical relationship is the 'subclass_of' relation. Description Logics reasoners rely on these relationships. Ontologies allow for the formal representation and analysis of many other types of relationships and constraints as well, thus allowing, for instance, for advanced retrieval services on large knowledge bases.

3.4 Effort at Creation Time vs. Effort at Usage Time

As usual, there is a trade-off: the higher quality results have to be, the more (preparative) time and resources it takes. A folksology is relatively easy to build (anybody is able to create tags or use anybody else's tags), but at the cost of potential misunderstandings and noise at browsing and retrieval time for which an individual user has to compensate by evaluating and selecting the look-up results himself. An ontology requires a substantial effort of experts on reaching meaning agreements and formally representing them, but then has the advantage of setting the semantics as precise as possible and grounded in the domain, which should remove potential ambiguities and misunderstandings in favour of interoperability. The situation is comparable to the classical word-based search engines that produce a large amount of noise. Semantics-based search engines, such as OntoSeek [21] and Swoogle [12], offer the promise of returning more accurate results and reducing noise. Typically IE/IR results are measured in terms of recall (how many elements of the ideal answer set are contained in the actual answer set?) and precision (how many elements of the actual answer set are correct ?). It is widely assumed that indexing pages with ontology tags will improve IE/IR results. Likewise, bookmarking pages with tags for which there exists an agreement and explicit definition of their meaning will result in more accurate look-up results.

3.5 Individual Creation Decision vs. Group-Wise Creation Agreement

The fact that in a folksological setting, any individual is the allmighty "deus ex machina" who can very easily create, rename, group, split, merge and delete tags and thus categorise and classify his small universe at will largely explains the success of folksonomies - see e.g., [34]. In addition, by making use of social bookmarking sites and their software, the individual receives an immediate feedback on how other individuals use (the same) tags (e.g., all sites bookmarked by the 'tools' tag are listed by browsing http://del.icio.us/tags/tools) and can adapt/conform his tagging behaviour. Although in principle, ontologies should be the result of a group-wise negotiation process, the current practice unfortunately still is that a single ontology engineer encodes his conceptualisation of a domain (or in many cases an application) in an ontology language such as OWL and subsequently hopes that other stakeholders in the domain are happy with it and adopt it. The immediate feedback loop of folksologies is missing here. In order to support the agreement process that links a unique label to a meaning description, a distributed negotiation and decision support system and logging mechanism is needed. The suggestion of using folksologies as the start for professional

[15] Del.icio.us allows a user to define a label regrouping tags, but the system then only uses it for local display purposes.

controlled vocabularies ([40]) should be retained [16] and could serve in the same way as quantitative linguistic corpus analysis does to determine the most important terms for a domain.

3.6 The (Unbearable?) Lightness of Being a Tag Creator vs. the Authoritative Weight of a Relevant Stakeholder

As anybody is allowed to create a social bookmark, no single individual is able to reliably assess the value of the tags. Only the frequency of use gives an indication of how others perceive the relevance and value of a tag. On the Internet, all tag creators are equal. But also here, some are more equal than others. Some persons simply are more knowledgeable and have better expertise so that their view on a part of the world is accepted as more relevant, better organised, etc. Not just anybody should participate in a committee of representative experts and prime stakeholders that decides on the content of an ontology. In a professional environment this way of working is widely applied to build controlled vocabularies or decide on the use of XML tags, for instance. The creators of an ontology are attributed an authoritative weight that inspires trust to potential users of this ontology about its quality, especially as objective measures to evaluate ontologies - see e.g. [55] - are still in their infancy [6, 24]. Web pages annotated with qualitative tags will be preferentially visited by intelligent software agents. Merely looking at frequencies of tag use is too simplistic a measure [17]. One could attribute weights to authoritative users, but this implies a preliminary agreement (and limited in time ?) on the weigths attributed to these authorities. We prefer a combination of authoritative weights and frequency of usage as, for instance, has been implemented in the FolkRank algorithm [26]. In the areas of cultural anthropology and group communication theory, interesting research on consensus analysis is done taking into account the authority of an information source [4].

4 Implementing an Emergent Semantics System

In this section, we describe how a social bookmarking system has been combined with an ontology server in order to create an ESS along the lines as described in the previous section and summarised by the following requirements:

- differentiate words from senses
- combine formal and informal semantics
- structure the domain of discourse
- combine individual suggestions with committee style decisions
- combine spontaneous trends with expert opinions

Before discussing the actual implementation (section 4.4), we summarise the characteristiscs of two existing social bookmarking tools, namely del.icio.us (section 4.1) and Piggy Bank (section 4.2) to mark their inadequateness. Our system called DogmaBank extends the Piggy Bank open source software [18] with DOGMA technology (section 4.3) in order to meet the requirements mentioned above.

[16] However, the fact that only simple words are used as tags seriously biases the outcomes.
[17] E.g., one cannot simply state that a definition is incorrect only because it is hardly used.
[18] Piggy Bank is licensed under the BSD license - see http://simile.mit.edu/piggy-bank/ .

4.1 Del.icio.us

Del.icio.us is representative of most social bookmarking sites - other well known ones are Connotea [31] and Bibsonomy. Any human (tagger) is able to create, rename, delete, merge and split tags (= a single word) and associate them with a web page. The system stores a tag, its associated URL and tagger and makes this information available to any other user (subscriber) who can annotate his web pages with these tags. Relations between tags are either based on frequencies ("Related tags") or aggregates of tags (using the "Bundle" functionality to create containers of tags).

Homonyms nor synonyms can be detected at creation time. Only when a subscriber examines with which pages a certain tag is associated he discovers potential cases of homonymy and synonymy. No informal or formal definitions are provided. As there is no possibility to create conceptual representations, only human agents can make use of this tool. There is no explicit group interaction foreseen (only implicitly as via usage frequencies one sees how the group behaves).

4.2 Piggy Bank

Piggy Bank [27] is another, more recently developed, social bookmarking tool that disposes of an extended functionality compared to del.icio.us, especially regarding the sharing, organising and re-using of the information collected during bookmarking (and scraping) activities. A user creates a new tag (= keyword) but "under the hood" a URI to an RDF resource (with the keyword as the OWL label) is created. However, the user interface does not yet support applying the OWL:DifferentFrom and OWL:SameAs primitives to cope with synonymy and homonymy.

Nevertheless, except for the merely syntactic definition of the RDF resource, no explicit formal or informal description of its exact meaning is available. As a consequence, only human agents are intended to make use of this tool. But at least, the Piggy Bank system designers are aware of the difference between a keyword and a concept and have foreseen an initial mechanism for it - although not yet applied in practice. It implies that no relationships between the tags (or rather the keywords pointed to) are available, except for a listing of URLs per tag.

4.3 DOGMA

The DOGMA (**D**eveloping **O**ntology-**G**rounded **M**ethodologies and **A**pplications) ontology engineering framework developed in our group builds on the fundamental distinction between an ontology base and a commitment layer [29]. This double articulation [51] in an intuitive ontology base and formal commitment layer (formal semantic constraints on a selection of the represented domain knowledge) is combined with basic insights from linguistics to offer a powerful framework for ontology engineering and modelling [39] that takes into account the difference between a natural language word and a concept label (logical term) identifying some notion or concept [54]. To cope with homonyms and synonyms we state that a concept is lexicalised for a certain language in a certain context by a specific word [56] [19] (cf. the famous example

[19] We refer to [9] for a formal description of these ideas.

of the many meanings of "bank" [7] or the various denominations for beer glasses in Australia [23, p.206]. As the DOGMA Concept Definition Server implementation incorporates the principles mentioned above, its functionalities (e.g., one method returns the concept label associated with a particular term) constitute a good basis to add a conceptual layer to Piggy Bank. To our knowledge, there are not so many ontology servers that distinguish between a natural language term and a concept - another one is for example KAON-2[20].

Methods and practices of distributed collaboration and negotiation for ontology engineering are an important aspect of ontology engineering - [11] describes a collaborative approach within the DOGMA framework. Not only "committee-style" approaches should be looked at, but also spontaneously emerging meaning agreements. Especially for the latter scenario we hope to incorporate some of the "folksological" techniques next to methods from social sciences, possibly supported by ontologies [10].

4.4 DogmaBank

In the following, we outline how we have (partially) complemented and extended the Piggy Bank software with an ontology server that is able to link words or terms to concepts and vice versa. Remember that the del.icio.us and Piggy Bank lack any conceptual machinery and collapse words and meaning but offer an intuitive way to create annotations, while the DOGMA technology should include means to cater for the phenomenon of emergent meaning definition in its formal conceptual engineering framework.

We could have chosen to build YABT (Yet Another Bookmarking Tool) from scratch, using one of the rapid Web2.0 development frameworks that are increasingly becoming available. However, we feel this would have been an isolated attempt. Instead, we have adapted a well known system, Piggy Bank (see section 4.2), of which we preserved all of the existing functionality in our DogmaBank implementation. This makes the transition easier for a Piggy Bank user. We have implemented a new user interface to support *ontology-based tag creation and tagging* processes (the upper part of Figure 1). This interface replaces the standard Piggy Bank interface. It is implemented as a Firefox plug-in [21]. The plug-in connects to the DOGMA Concept Definition Server to retrieve WordNet [22] (or other WordNet-style) senses that are associated with the keyword entered (the right part of Figure 1). We refer for the remainder of this paragraph to Figure 2 to illustrate how one can tag web pages with concepts (step 1). The user types in a keyword (step 2), hits the fetch button or enter key (step 3) and a list of potential senses (i.e., informal lexicographic definitions of its different meanings or concepts) appears as well as some additional explanatory descriptions or glosses (e.g., examples of use). Retrieving senses is performed in two stages: (i) the synsets are checked on the presence of the term (e.g., ‘car’) and (ii) associations between a term and concept(s) as recorded in the DOGMA Ontology Server are looked up. Duplicates are removed and the resulting senses are displayed. The user now selects a particular sense (step 4)

[20] http://kaon2.semanticweb.org/

[21] The same internet technology as the original Piggy Bank interface (Chrome, XUL, Javascript, XPCOM, Jetty and Ajax) is used.

[22] As WordNet mainly contains general vocabulary, more domain specific terminological sources will be needed. A suggestion found in the literature is to refer to Wikipedia entries.

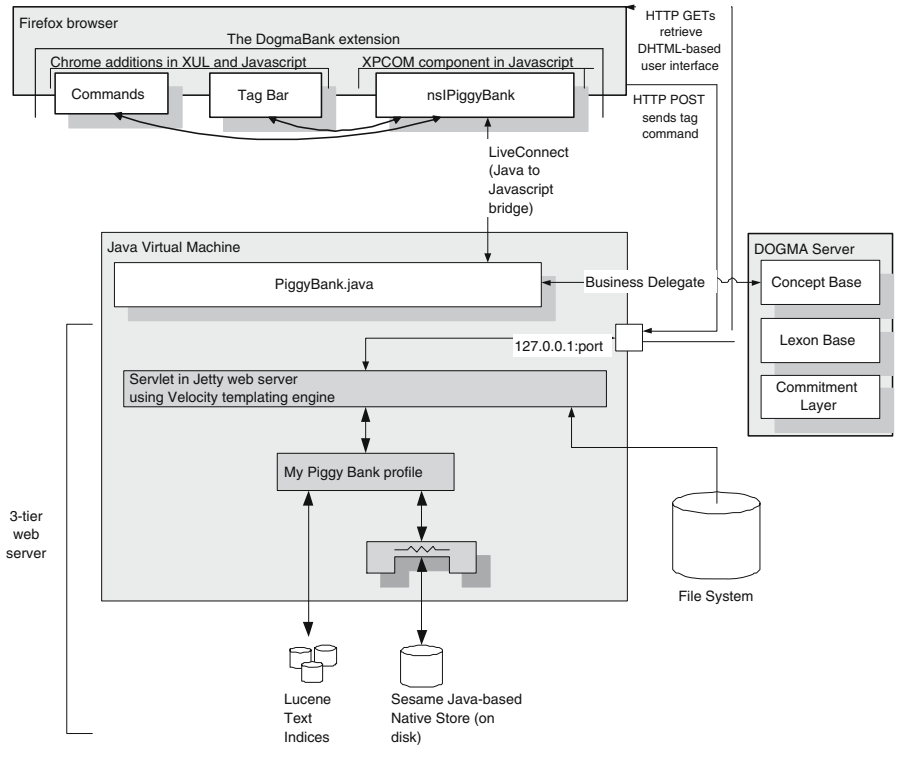


Fig. 1. The system architecture - partially adapted from [27]

to tag the web page (step 5). It is also still possible to tag a web page with a *freely chosen keyword* (arrow 6). A user might be waiting for a concept to be added to the ontology by the moderator (see below), and decides to use such a keyword for the time being. Or he might want to mark a web page as “toread”, which is a popular tag on current bookmarking tools, but without any relationship at all with the content or topic of the web page. All tags being alphanumeric, the original Piggy Bank system modules are re-used as they are (the lower part of Figure 1). Only, we have subverted (or rather ‘smartened’) the Piggy Bank system by having it store concept labels pointing to explicitly and commonly defined senses rather than personal interpretations of natural language words.

The same mechanisms as used by folksonomies (usage frequencies, URLs per tag and tags per URLs) are now available to facilitate the meaning negotiation process for ontologies. It constitutes an interesting bottom-up complement (or maybe even alternative) to the traditional top-down or middle-out approach of engineering ontologies - see also [42]. Note that we are mainly considering what some call "lightweight ontologies", i.e. first order logic vocabularies without associated inference rules, but with

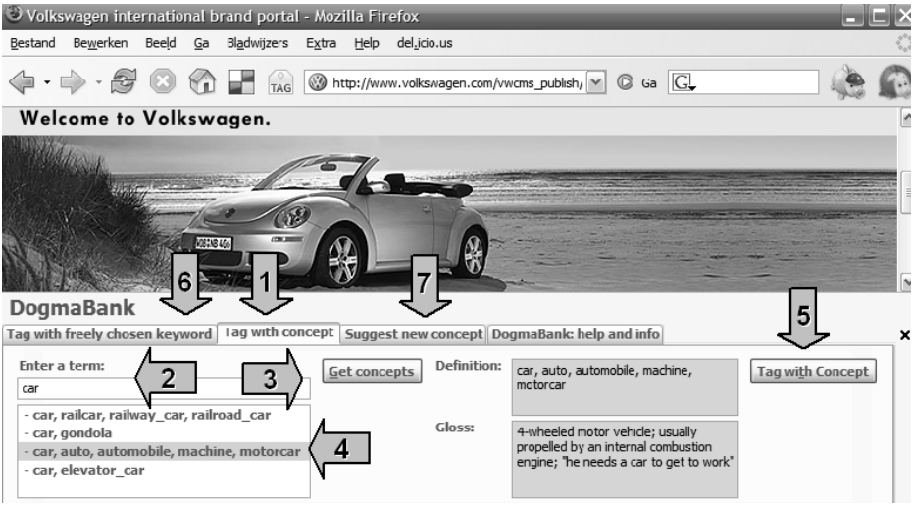


Fig. 2. The DogmaBank Firefox plug-in input and tag window with the senses for the term 'car'

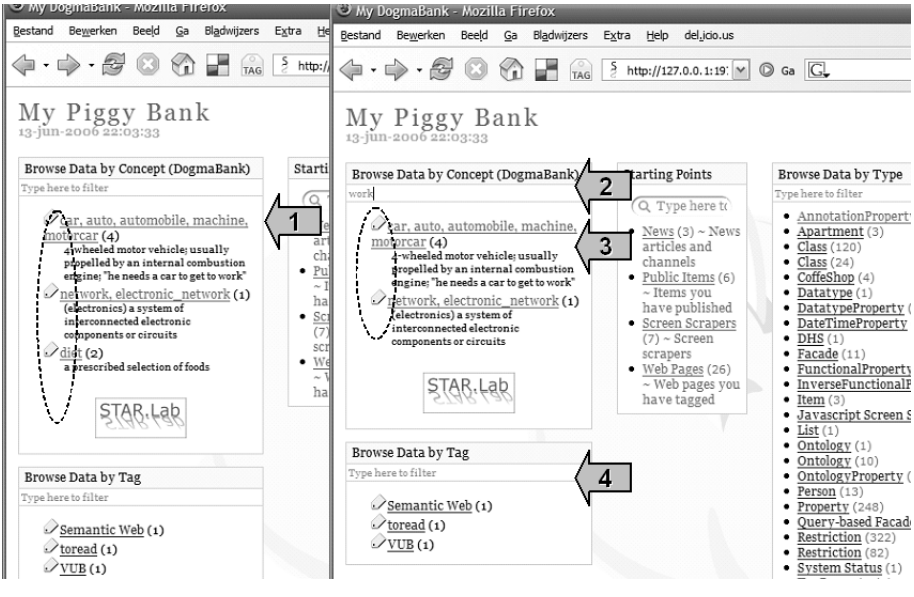


Fig. 3. The slightly adapted Piggy Bank interface showing a user's DOGMA concept tags and descriptions and illustrating the "refine as you type" functionality

explicit definitions for the vocabulary signature[23]. These mechanisms will provide the "ontology stakeholder committee" mentioned above with feedback on how their ontology is being used in practice. So one cannot create new concept tags at will, instead one suggests new tags with their definition (arrow 8 of Figure 2) to a committee of representative stakeholders for acceptance or rejection [24] to avoid an ontology from wildly and anarchically expanding. In order to avoid ending up with a huge collection of sligthly varying definitions, similarity checks have to be performed before a new concept description is entered in the system. Even the existing "folksabularies" are useful for ontology creation. In the same way as natural language processing methods cluster words of a corpus based on their distribution, tags can be "clustered" around the URL tagged. In both cases, it is assumed that the words or tags express somehow a similar meaning. Experience shows that this kind of information is very helpful when building an ontology. Formal definitions still have to be added by ontologists as this requires extra technical skills. The formal definitions mainly serve for machine-machine interoperability purposes, such as semantic web services. Also user-machine interaction (e.g., searches) will be improved by applying taxonomic and other reasoning mechanisms (see e.g. [21,37]). For that purpose, formal relationships between the concepts are needed. In a first phase defining relationships are restricted to ontologists as this activity is more complex. Semantic Web tools [16, p.293 ff.] support this activity.

Also for information browsing and retrieval purposes - see Figure 3, a user will be able to enter keywords (step 1) and choose the appropriate sense as displayed by the system (step 3). To optimise the latter step, a "refine while you type" functionality has been implemented that searches the synsets and related term sets (step 2). It limits the number of potentially relevant senses shown to the user. The DogmaBank system is able cope with synonymy, homonymy and multilinguality - the latter not yet finalised. The Piggy Bank software has been slightly adapted to include the "browse data by concept" pane. Searching by key word tags is still possible (arrow 4). Figure 4 shows the search results for "car".

5 Related and Future Work

In this paper, we examined and illustrated how informal semantics resources (i.e. folksologies) and formal semantics resources (i.e. ontologies) meet productively in the form of an ESS supporting Internet-based communities in their collaborative work. Our analysis and upgrade of a current state-of-the-art system has helped us identify new focal points as well as gaps in current research and development: semantic matching and the development of similarity measures, and the groupwise usage of the information gathered via the (social) bookmarking process (including ways how to operationalise a stakeholder committee [11] and methods how to formalise a decision process leading to meaning consensus [4]) emerge as key issues here.

We will extend the DogmaBank system to transparantly determine at creation time whether or not the intended meaning of a new tag (i.e. its sense) is similar to an already

[23] It might be possible that inheritance and subclass relationships have been defined but are transparent for human taggers. More research needs to be done here.

[24] This part of the DogmaBank system is still under development.

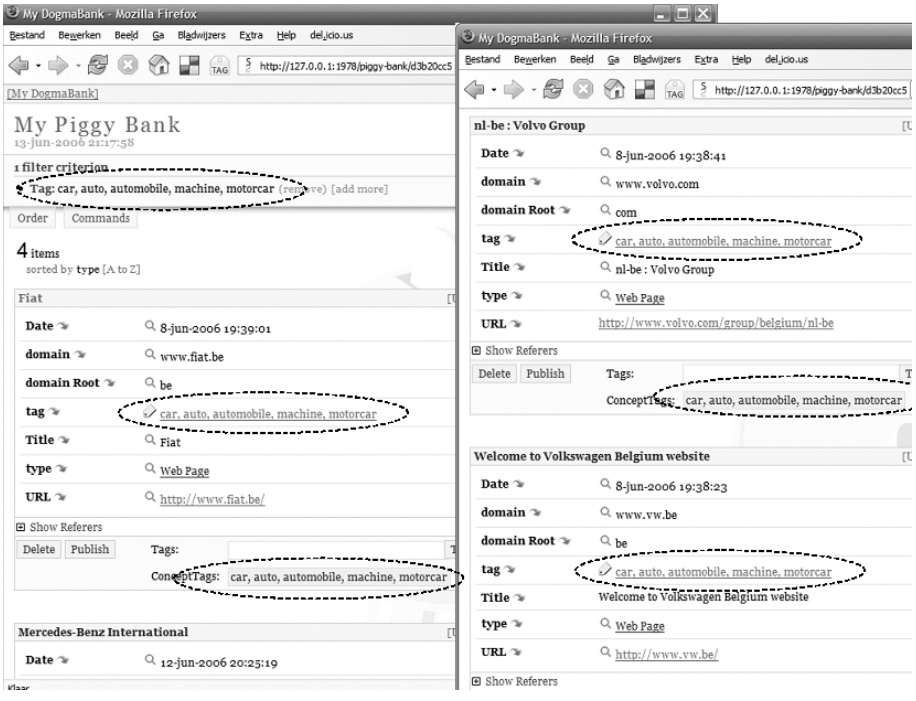


Fig. 4. The Piggy Bank interface showing the search results for the concept "car"

existing one. Therefore, semantic similarity measures (e.g., [15, 33]) will be of prime importance not only for the work mentioned in this paper, but for many Semantic Web applications. Euzenat et al. [13] have gathered a large number of metrics of different origins that measure semantic distance (albeit in the context of ontology aligning and merging) between concepts. The key point will be to identify the most relevant and effective ones to be incorporated in DogmaBank-alike software.

Another area of related work concerns semantic and community portals (e.g., [12, 30, 32, 52]). A community portal stores and allows retrieval of all kinds of information of interest to a certain community. It is a centralised repository in which community members following a work flow upload data and annotate it with pre-defined tags, which might belong to an ontology, for easy retrieval purposes. A semantic portal basically adds semantic technology (i.e., searching on ontological meta-data instead of regular data) to retrieve information more accurately. The big difference with our upgraded social bookmarking tool is that a semantic community portal receives its meta-data from a central repository or from a crawler that gathers meta-data instances and URLs from all over the Web. In this case, the web site creators have to provide for the meta-data schema to be used by the crawler. No new tags can be added. In the context of the DogmaBank social bookmarking tool, a human more or less acts as the crawler (i.e. encounters relevant information and decides to semantically tag a web page and store

the reference). The "regular" Piggy Bank store however offers much less sophisticated retrieval possibilities (cf. Figure 4) than e.g., the OntoWeb semantic portal [44]. From the above, it should be clear that a semantic community portal is the natural complement of a semantically upgraded social bookmarking tool.

Also, many semantic annotating tools exist - cf. [16, p.344 ff.] for an overview[25]. The main difference with DogmaBank is that these tools only load a fixed set of tags of a predefined (and visualised) ontology. Tagging is an individual activity and happens with a finer granularity (single information items (instances) rather than entire web pages). When combined with language technology, it mostly concerns named entity recognition - cf. e.g., [35, 46]. These tools can also be used for automated ontology population.

Semantic annotators, crawlers, social bookmarking tools and community portals are applied in different contexts and settings and complement each other. They all contribute to the same overall goal, i.e. adding meta-data to the web and offering a means to share within a community information on the web. The challenge in ESSs is to find bridges between these automated and human approaches to creating and using formal semantics, thus working towards hybrid, semi-automated approaches, which is exactly what our DogmaBank tool aims at.

6 Conclusion

Semantics play an ever more important role on the Internet. Two major streams in research and development are distinguished: the world of formal semantics, exemplified by ontologies and the Semantic Web, and the world of informal semantics, of which folksologies (as frequently used by the Web2.0 community) are important bearers. The mission of the Semantic Web is to promote interoperability of information resources, so that machine agents become better at information retrieval. The Web2.0 wants to exploit the power of human communities to do the same.

Increasingly, the two worlds meet. In order to tap the full potential of the Web, emergent semantics systems will be required: well-designed socio-technical systems of formal and informal semantics, filled and used by well-calibrated combinations of machine and human agents. To help to analyse the larger picture (including the full range of semantics' functionalities in their context of use [48], which paves the way to a Pragmatic Web [47, 53]), there is a need for the development of a meta-model of emergent semantics systems. We have implicitly already presented some elements of such a meta-model.

The main message of this paper is that there is a need and possibility (proven by our DogmaBank implementation, albeit still a preliminary one) to shift from thinking in terms of individual techniques to more holistic systems supporting communities of practice. In that way, it becomes much easier to identify the linkages and gaps between these techniques, and opportunities for new applications. We hope that this paper will contribute to further catalyzing and focusing the fundamental emergent semantics debate which is currently defining the future of a World Wide Web, be it as a Web2.0, a Semantic or Pragmatic web, or a combination of all.

[25] See also http://annotation.semanticweb.org/tools/.

Acknowledgements

Part of this research has been financed by the Flemish OntoBasis project (IWT GBOU 2001 #10069) [PS] and the DIP project (EU IST - FP6 507483). With many thanks to Luk Vervenne for pointing us to Piggy Bank.

References

1. Karl Aberer, Tiziana Catarci, and Philippe Cudré-Mauroux et al. Emergent semantics systems. In M. Bouzeghoub, C. Goble, V. Kashyap, and S. Spaccapietra, editors, *Semantics for Grid Databases, First International IFIP Conference on Semantics of a Networked World: ICSNW 2004, Revised Selected Papers*, volume 3226 of *LNCS*, pages 14–43. Springer, 2004.
2. Karl Aberer, Philippe Cudré-Mauroux, and Manfred Hauswirth. A framework for semantic gossiping. *SIGMOD Record Special Issue*, 31(4):48 – 53, 2002.
3. C. Batini, M. Lenzerini, and S. Navathe. A comparative analysis of methodologies for database schema integration. *ACM Computing Surveys*, 18(4):323–364, 1986.
4. C. Behrens and V. Kashyap. The "Emergent" Semantic Web: A consensus approach for deriving Semantic Knowledge on the Web. *Real World Semantic Web Applications*. Frontiers in Artificial Intelligence and Applications, Vol 92, IOS Press, 2002.
5. T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, 284(5):34–43, 2001.
6. Andrew Burton-Jones, Veda Storey, and Vijayan Sugumaran. A semiotic metrics suite for assessing the quality of ontologies. *Data and Knowledge Engineering*, 55(1):84 – 102, 2005.
7. S. Buvac. Resolving lexical ambiguity using a formal theory of context. In Van Deemter and Peters, editors, *Semantic Ambiguity and Underspecification*. CSLI Publications, 1996.
8. Philippe Cudré-Mauroux and Karl Aberer et al. Viewpoints on emergent semantics. *Journal of Data Semantics*, 2880. to appear.
9. P. De Leenheer and R. Meersman. Towards a formal foundation of DOGMA ontology Part I: Lexon base and concept definition server. Technical Report STAR-2005-06, STAR Lab, Brussel, 2005.
10. Aldo de Moor. Patterns for the pragmatic web. In *Proc. of the 13th International Conference on Conceptual Structures (ICCS 2005)*, volume 3596 of *LNAI*, pages 1–18. Springer, 2005.
11. Aldo de Moor, Pieter De Leenheer, and Robert Meersman. DOGMA-MESS: A meaning evolution support system for interorganizational ontology engineering. In *Proc. of the 14th International Conference on Conceptual Structures (ICCS 2006)*, LNCS. Springer, 2006.
12. L. Ding, T. Finin, A. Joshi, Y. Peng, R. Scott Cost, J. Sachs, R. Pan R, P. Reddivari, and V. Doshi. Swoogle: A semantic web search and metadata engine. In *Proceedings of the 13th ACM Conference on Information and Knowledge Management*. ACM, 2004.
13. J. Euzenat, T. Le Bach, J. Barrasa, P. Bouquet, and J. De Bo et al. State of the art on ontology alignment. Knowledge Web Deliverable #D2.2.3, INRIA, Saint Ismier, 2004.
14. James Farrugia. Model-theoretic semantics for the web. In *Proceedings of the WWW2003 Conference*, pages 277 – 287, 2003. http://www2003.org/cdrom/.
15. F. Giunchiglia and M. Yatskevich. Efficient semantic matching. In Gomez perez A. and Euzenat J., editors, *Proceedings of the 2nd European semantic web conference (ESWC'05)*, volume 3532 of *Lecture Notes in Computer Science*, 2005.
16. Asunción Gómez-Pérez, Mariano Fernández-López, and Oscar Corcho. *Ontological Engineering*. Springer Verlag, 2004.
17. W. Grosky, D. Sreenath, and F. Fotouhi. Emergent semantics and the multimedia semantic web. *SIGMOD Record Special Issue*, 31(4):54 – 58, 2002.

18. T. R. Gruber. A Translation Approach to Portable Ontology Specifications. *Knowledge Acquisition*, 6(2):199–221, 1993.
19. Tom Gruber. Ontology of folksonomy: A mash-up of apples and oranges. In *First On-line Conference on Meta-Data and Semantics Research (MTSR05)*, 2005. http://tomgruber.org/.
20. N. Guarino and P. Giaretta. Ontologies and knowledge bases: Towards a terminological clarification. In N. Mars, editor, *Towards Very Large Knowledge Bases: Knowledge Building and Knowledge Sharing*, pages 25 – 32, Amsterdam, 1995. IOS Press.
21. N. Guarino, C. Masolo, and G. Vetere. Ontoseek: Content-based access to the web. *IEEE Intelligent Systems*, May-June4-5:70–80, 1999.
22. Nicola Guarino. Formal ontologies and information systems. In Nicola Guarino, editor, *Proceedings of FOIS '98*, pages 3 – 15. IOS Press, 1998.
23. Terry Halpin. *Information Modeling and Relational Databases*. Academic Press, 2001.
24. Jens Hartmann, Peter Spyns, Diane Maynard, Roberta Cuel, Mari Carmen Suarez de Figueroa, and York Sure. Methods for ontology evaluation. KnowledgeWeb Deliverable #D1.2.3, 2005.
25. Dion Hinchcliffe. 10 issues facing web2.0 today. http://web2.wsj2.com/10_issues_facing_web_20_going_into_2006.htm#, 2005.
26. Andreas Hotho, Robert Jäschke, Christoph Schmitz, and Gerd Stumme. Information retrieval in folksonomies: Search and ranking. In *Proceedings of the 3rd European Semantic Web Conference*, LNCS. Springer, 2006.
27. David Huynh, Stefano Mazzocchi, and David Karger. Piggy bank: Experience the semantic web inside your web browser. In *Proceedings of the third International Semantic Web Conference (ISWC05)*, LNCS. Springer Verlag, 2005.
28. M. Jarke, M. Lenzerini, Y. Vassiliou, and Y. Vassiliadis. *Fundamentals of Data Warehouses*. Springer-Verlag, 1999.
29. M. Jarrar and R. Meersman. Formal ontology engineering in the dogma approach. In R. Meersman, Z. Tari, and al., editors, *On the Move to Meaningful Internet Systems 2002: CoopIS, DOA, and ODBASE*, volume 2519 of *LNCS*, pages 1238 – 1254. Springer, 2002.
30. H. Lausen, Y. Ding, M. Stollberg, D. Fensel, R. Lara, , and S.K. Han. Semantic web portals: state-of-the-art survey. *Journal of Knowledge Management*, 9(5), 2005.
31. Ben Lund, Tony Hammond Martin Flack, and Timo Hannay. Social bookmarking tools (ii): a case study - Connotea. *D-Lib Magazine*, 11(4), 2005. http://www.dlib.org/dlib/april05/lund/04lund.html.
32. A. Maedche, S. Staab, R. Studer, Y. Sure, and R. Volz. SEAL — tying up information integration and web site management by ontologies. *IEEE Data Engineering Bulletin*, 25(1):10–17, March 2002.
33. Ana Maguitman, Filippo Menczer, Heather Roinestad, and Alessandro Vespignani. Algorithmic detection of semantic similarity. In *Proc. of the WWW2005 Conference*. ACM, 2005.
34. Adam Mathes. Folksonomies: Cooperative classification and communication through shared metadata. http://www.adammathes.com/academic/computer-mediated-communication/folksonomies.html, 2004.
35. Diane Maynard. Benchmarking ontology-based annotation tools for the semantic web. In *UK e-Science Programme All Hands Meeting (AHM2005) Workshop Text Mining, e-Research and Grid-enabled Language Technology*, 2005.
36. Stefano Mazzocchi. Folksologies: de-idealizing ontologies. Blog http://www.betaversion.org/~stefano/linotype/news/85/, 2005.
37. Deborah McGuinness. Question answering on the semantic web. *IEEE Intelligent Systems*, (january/february):82 – 85, 2004.
38. R. Meersman. Semantic web and ontologies: Playtime or business at the last frontier in computing ? In *NSF-EU Workshop on Database and Information Systems Research for Semantic Web and Enterprises*, pages 61 – 67. NSF-EU, 2002.

39. Robert Meersman. The use of lexicons and other computer-linguistic tools in semantics, design and cooperation of database systems. In Y. Zhang, M. Rusinkiewicz, and Y. Kambayashi, editors, *The Proceedings of the Second International Symposium on Cooperative Database Systems for Advanced Applications (CODAS99)*, pages 1–14. Springer, 1999.
40. Peter Merholz. Metadata for the masses. http://www.adaptivepath.com/publications/essays/archives/000361.php, 2004.
41. Peter Merholz. Clay Shirky's viewpoints are overrated. http://www.peterme.com/archives/000558.html, 2005.
42. P. Mika. Ontologies are us: A unified model of social networks and semantics. In *Proc. of the 4th Internat. Semantic Web Conf. (ISWC05)*, LNCS, pages 522–536. Springer, 2005.
43. S. Nirenburg and V. Raskin. Ontological semantics, formal ontology, and ambiguity. In *Proceedings of the Second International Conference on Formal Ontology in Information Systems*, pages 151 – 161. ACM Press, 2001.
44. Daniel Oberle and Peter Spyns. *Handbook on Ontologies*, chapter OntoWeb - Knowledge Portal, pages 499 – 516. International Handbooks on Information Systems. Springer, 2004.
45. C.K. Ogden and I.A. Richards. *The Meaning of Meaning: A Study of the Influence of Language upon Thought and of the Science of Symbolism*. Routledge & Kegan Paul Ltd., London, 10 edition, 1923.
46. Borislav Popov, Atanas Kiryakov, Damyan Ognyanoff, Dimitar Manov, and Angel Kirilov. KIM - a semantic platform for information extaction and retrieval. *Journal of Natural Language Engineering*, 10(3-4):375 – 392, 2004.
47. Mareike Schoop, Aldo de Moor, and Jan Dietz. The pragmatic web: a manifesto. *Commun. ACM*, 49(5):75–76, 2006.
48. Amit P. Sheth, Cartic Ramakrishnan, and Christopher Thomas. Semantics for the Semantic Web: "The Implicit, the Formal and the Powerful". *International Journal on Semantic Web and Information Systems* 1(1): 1–18, 2005.
49. Clay Shirky. The semantic web, syllogism, and worldview. http://www.shirky.com, 2003.
50. Clay Shirky. Ontology is overrated: Categories, links and tags. http://www.shirky.com, 2005.
51. Peter Spyns, Robert Meersman, and Mustafa Jarrar. Data modelling versus ontology engineering. *SIGMOD Record Special Issue*, 31 (4): 12 - 17, 2002.
52. P. Spyns, D. Oberle, R. Volz, J. Zheng, M. Jarrar, Y. Sure, R. Studer, and R. Meersman. Ontoweb - a semantic web community portal. In D. Karagiannis and U. Reimer, editors, *Proceedings of the Fourth International Conference on Practical Aspects of Knowledge Management (PAKM02)*, volume 2569 of *LNAI*, pages 189–200. Springer Verlag, 2002.
53. Peter Spyns and Robert Meersman. From knowledge to interaction: from the semantic to the pragmatic web. Technical Report 05, STAR Lab, Brussel, 2003.
54. P. Spyns and J. De Bo. Ontologies: a revamped cross-disciplinary buzzword or a truly promising interdisciplinary research topic? *Linguistica Antverpiensia NS*, (3):279–92, 2004.
55. P. Spyns and M.-L. Reinberger. Lexically evaluating ontology triples automatically generated from text. In A. Gómez-Pérez and J. Euzenat, editors, *Proceedings of the second European Semantic Web Conference*, volume 3532 of *LNCS*, pages 563 – 577. Springer Verlag, 2005.
56. Peter Spyns. Object Role Modelling for Ontology Engineering in the DOGMA framework. in R. Meersman and Z. Tari and P. Herrero et al. *On the Move to Meaningful Internet Systems 2005: OTM 2005 Workshops*, volume 3762 of *LNCS*, pages 710 – 719. Springer Verlag, 2005.
57. M. Uschold and M. Gruninger. Ontologies: Principles, methods and applications. *Knowledge Sharing and Review*, 11(2), June 1996.
58. M. Ushold and M. Gruninger. Ontologies: Principles, methods and applications. *The Knowledge Engineering Review*, 11(2):93 – 155, 1996.

Transactional Behavior of a Workflow Instance

Tatiana A.S.C. Vieira and Marco A. Casanova

Pontifical Catholic University of Rio de Janeiro
Rua Marquês de São Vicente, 225
Rio de Janeiro, RJ - Brazil - CEP 22.451-900
Phone: +55-21-3527-1500 ext. 4347; FAX: +55-21-3527-1530
tascvieira@yahoo.com.br,
casanova@inf.puc-rio.br

Abstract. Workflow management systems usually interpret a workflow definition rigidly, allowing no deviations during execution. However, there are real life situations where users should be allowed to deviate from the prescribed static workflow definition for various reasons, including lack of information about parameter values and unavailability of the required resources. To flexibilize workflow execution, this paper proposes an exception handling mechanism that allows the execution to proceed when otherwise it would have been stopped. The proposal is introduced as a set of extensions to OWL-S that capture the information required for the flexibilization mechanism. In particular, this paper focus on the transactional behavior of a workflow instance, in the sense that it guarantees that either all actions executed by the instance terminate correctly or they are all abandoned.

1 Introduction

Workflow management systems, or workflow systems, for short, have an extensive list of requirements, from cooperation and coordination to synchronization. In this work, we focus on a requisite that we call *flexible execution*.

Usually, workflow systems execute a workflow instance rigidly, not allowing any deviation from the workflow definition at runtime. However, this rigid interpretation may sometimes lead to a long waiting time, among other problems.

We therefore propose a model for workflow instance execution based on a notion of workflow flexibilization. For example, the flexibilization mechanism allows a workflow instance to proceed execution without waiting for the availability of a resource or the value of a parameter. The mechanism can be interpreted as a component substitution strategy, applied to workflow execution, where subworkflows and resources act as components.

Our strategy is to consider as an exception any situation not pre-defined in the process structure that can be properly handled at runtime. We define the flexibilization mechanism with the help of an abstract machine that adopts triggers to specify state changes and instance flexibilization. Furthermore, we describe workflows with the help of an extension of OWL-S, that accommodates the necessities of the flexibilization mechanism.

R. Meersman, Z. Tari et al. (Eds.): OTM 2006, LNCS 4275, pp. 756–771, 2006.

This paper is organized as follows. Section 2 summarizes related work. Section 3 presents some preliminary concepts for the understanding of the paper, including a brief description of the flexibilization mechanism. Section 4 presents the ontologies constructed to support the mechanism. Section 5 discusses the transactional behavior of a workflow instance. Finally, Section 6 presents the conclusions.

2 Related Work

The evolution of workflow systems towards less restrictive coordination approaches is discussed in [13]. Most of the earlier proposals for flexibilization mechanisms suggest to dynamically change the workflow structure [3].

Rule-driven frameworks that support structural changes in a workflow, by dictating how tasks can be inserted or removed from the workflow at runtime, are presented in [10]. The flexibilization mechanisms described in [1] [15] also offer the possibility to include, remove or even stop activities during workflow execution. The mechanisms proposed in our paper follow a different strategy by allowing execution to proceed in the presence of incomplete or negative information, with minimal user intervention.

The OPENflow system [9] offers two distinct flexibilization approaches: flexibility by selection and flexibility by adaptation. Flexibility by selection allows several paths to be included in the workflow description, but only one path is chosen at runtime. Flexibility by adaptation leads to workflow adaptation, at the instance level, when changes occur in the workflow structure at runtime (inclusion of one or more execution paths). Flexibility by adaptation, however, is used when, during runtime, a workflow instance is considered incomplete or erroneous. Our approach does not cover flexibility by selection, but it achieves flexibility by adaptation by using semantic information about workflow description to partly guide instance execution. Our proposal differs from the notion of flexibility by adaptation in OPENflow in that we also consider the case where an instance is delayed beyond some pre-specified time limit.

The flexibilization mechanisms in [8] account for the cooperation between users and for the anticipation of task execution. This mechanism offers a special transaction model for cooperative systems that deviates from the conventional start-end synchronization model [2].

The construction of workflow schemas from a standard set of modeling constructs is proposed in [11]. This is partially done at design time, and completed at runtime, according to selection, termination and workflow definition constraints, which dictate how each fragment of work can be included in the workflow, under what conditions the workflow instance can be terminated, and what conditions must hold during workflow definition. Flexibilization is achieved, in this case, by leaving workflow definition to be completed at runtime, according to the specified constraints. To some extent, our mechanism for choosing alternatives achieves a similar effect, as discussed in Section 3.

With respect to the transactional behavior of workflow instances, our work proposes an execution model that guarantees that either all actions executed by the instance terminate correctly or they are all abandoned, according to certain transactional properties. In addition, the transactional behavior takes into account execution consistency in the presence of flexibilization. It is similar to WS-Transaction Framework [6] [7], except that the latter focuses on Web services.

Lastly, the flexibilization mechanisms we propose bear some similarity to the query relaxation strategy adopted in the CoBase system [4] and in the CoSent system [5]. CoBase is a database system that uses the idea of cooperative queries. When queries are submitted, CoBase analyzes a hierarchy of additional information to enhance the query with relevant information. The information added to the query is bounded by a maximum semantic distance from the information present in the original query. This query modification mechanism is similar to the strategy we propose to deal with negative or incomplete information. In our proposal, the workflow description is enhanced with additional information available in the workflow ontology, as discussed in Section 4.

3 Preliminaries

3.1 Basic Concepts

Following OWL-S [12], we will use the terms *process* and *process instance*, instead of the terms workflow and workflow instance, from now on.

A *process* is a representation of a set of activities that are composed to reach a common goal. A *process instance* represents the execution os a process. A process is *atomic* if it is indivisible, and *composite* if it is a composition of two or more processes, called *subprocesses* or *component* processes. A *composite* process is defined with the help of *control constructs*, such as *join*, *split* and *sequence*.

An *abstract process* is a process that cannot be directly executed because it does not have any associated implementation. Hence, an abstract process cannot generate any process instance. By contrast, a *concrete process* has an associated implementation and, consequently, may generate process instances.

A process may require *resources* to be executed. A resource may be *physical* or *logical* and must be allocated to the process instance before it starts executing.

A process also has *pre-* and *post-conditions*. A pre-condition defines under which conditions the process can be executed. We assume that the availability of the necessary resources is an implicit precondition of a process.

3.2 A Brief Description of the Flexibilization Mechanism

The flexibilization mechanism we propose allows:

- the definition of a process to be completed at runtime, depending on execution data;
- the execution of a process instance to proceed even when some resource is unavailable or when the value of a parameter is unknown.

The ontologies explained in the Section 4 guide the flexibilization mechanism to:

- select concretizations for abstract processes and resources;
- compute default values for the parameters;
- find alternative processes and resources, used when the original processes or resources cannot be used;
- find processes to handle certain (types of) exceptions.

The flexibilization mechanism is based on the notion of *exception*, that is, a situation that is not considered in the static process definition and that, once registered with the mechanism, can be properly treated. Exceptions form a hierarchy, as shown in Figure 1, where an exception at level n can only be reached if an exception at level n-1 was reached, but could not be treated.

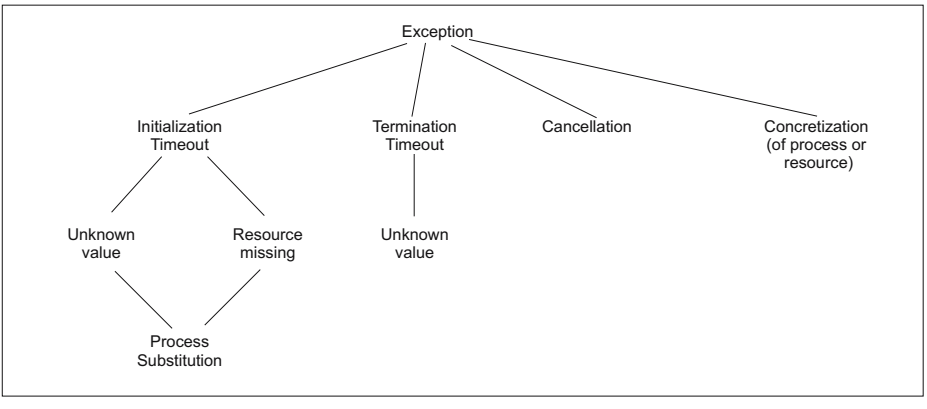


Fig. 1. Hierarchy of exceptions

At the first level of the hierarchy, we have four (types of) exceptions, namely:

initialization timeout exception: the system throws this exception when it cannot initialize an instance because the value of a parameter is unknown or because a required resource is unavailable, and the instance exceeded the allowed initialization waiting time, included in the process definition. The flexibilization mechanism treats this exception by selecting another process for execution, if one is available;

termination timeout exception: the system throws this exception when the instance cannot terminate within the allowed running time, included in the process definition. The flexibilization mechanism treats this exception by selecting another process for execution, if one is available;

concretization exception: the system throws this exception when it finds, during instance execution, a reference to an abstract process or to an abstract resource. The flexibilization mechanism treats this exception by finding a concrete process or a concrete resource which is associated with the abstract one;

cancellation exception: the system throws this exception when an atomic process instance is aborted. The flexibilization mechanism treats this exception by running a process to undo the instance effects, if one is available, before aborting the instance.

Under the initialization timeout exception, we have two second-level exceptions:

unknown input value exception: the system throws this second-level exception when the flexibilization mechanism does not find a process to handle the initialization timeout exception, and the exception was caused by an unknown input parameter value. The flexibilization mechanism treats this exception by trying to find default values for the unknown input parameters;

resource missing exception: the system throws this second-level exception when the flexibilization mechanism does not find a process to handle the initialization timeout exception, and the exception was caused by an unavailable resource. The flexibilization mechanism treats this exception by trying to find alternative resources that are semantically equivalent to the resources that could not be allocated.

Under the termination timeout exception, we have a second-level exception:

unknown output value exception: the system throws this second-level exception when the flexibilization mechanism does not find a process to handle the termination timeout exception. The flexibilization mechanism treats this exception by trying to find default values for the unknown output parameters.

Under the unknown input value and the resource missing exceptions, we have a third level exception:

process substitution exception: the system throws this third-level exception when the flexibilization mechanism does not find default values for the unknown input parameters, or alternative resources for the unavailable resources. The flexibilization mechanism treats this exception by trying to find alternative processes.

In addition to the concept of exception, the flexibilization mechanism depends on the concept of *weighted semantic proximity*, defined as a relationship between pairs of objects, processes or resources, with an attribute representing the weight. For instance, let p and p' be two processes. We indicate that p' is an alternative process for p by defining a semantic proximity relationship between p and p'. The weight of the relationship indicates the similarity of p' and p. The flexibilization mechanism will use such weights to find the best alternative for p. Note that, if p' substitutes p, then there must be a valid mapping between the parameters of p and p'.

Concretizations of abstract processes and resources are similarly modeled.

Default values can be treated statically, by including them in the process definition, or dynamically, through *presupposition rules* that take into account the status of the process instance execution.

We say that an instance selected for execution by the flexibilization mechanism is a *flexibilization instance*.

Finally, the subinstances of a given instance P are not directly defined by the static structure of p. They are rather defined as those instances that are under the control of P, after flexibilization. Instances that undo other instances do not have superinstances and cannot suffer flexibilization. In this sense, the flexibilization of an instance dynamically generates an instance tree.

4 Processes, Resources and Application Ontologies

The flexibilization mechanism we propose depends on additional information represented by mean of two basic ontologies:

- *pr*, the central ontology of processes and resources, which is domain independent. This ontology contains classes and properties that capture the concepts of process and resource;
- *lib*, a library of process definitions and resource descriptions, which applications can reuse. This ontology contains instances of the classes and properties defined at *pr* ontology.

The user may then define application processes, using the pr and the lib ontologies, and organize them as a new namespace. In what follows, we will use:

- *app*, an ontology of application processes, defined using the pr and the lib ontologies.

The *pr* ontology must allow the definition of: abstract and concrete processes; atomic and composite processes; abstract and concrete resources; concrete processes that implement abstract processes and the weight of this relationship; which process(es) treat(s) the nitialization/termination timeout exception of a process p; the execution cost of a process; the utilization cost of a resource; if a resource is or is not available; the default value of process parameters; and the parameter mappings between related processes. Since OWL-S covers many of these aspects, we opted to define the *pr* ontology by extending OWL-S.

Figure 2 shows the composition of the *pr* ontology and how it relates to the library of processes and resources and to the application ontology. It also presents the extensions (classes and resources) added to each ontology. The namespaces used in Figure 2 are (omitting the actual URIrefs):

process: namespace of the OWL-S process ontology
p-ext: namespace of the extended OWL-S process ontology
r: namespace of the OWL-S resource ontology
r-ext: namespace of the extended OWL-S resource ontology
pr: namespace of the *pr* ontology

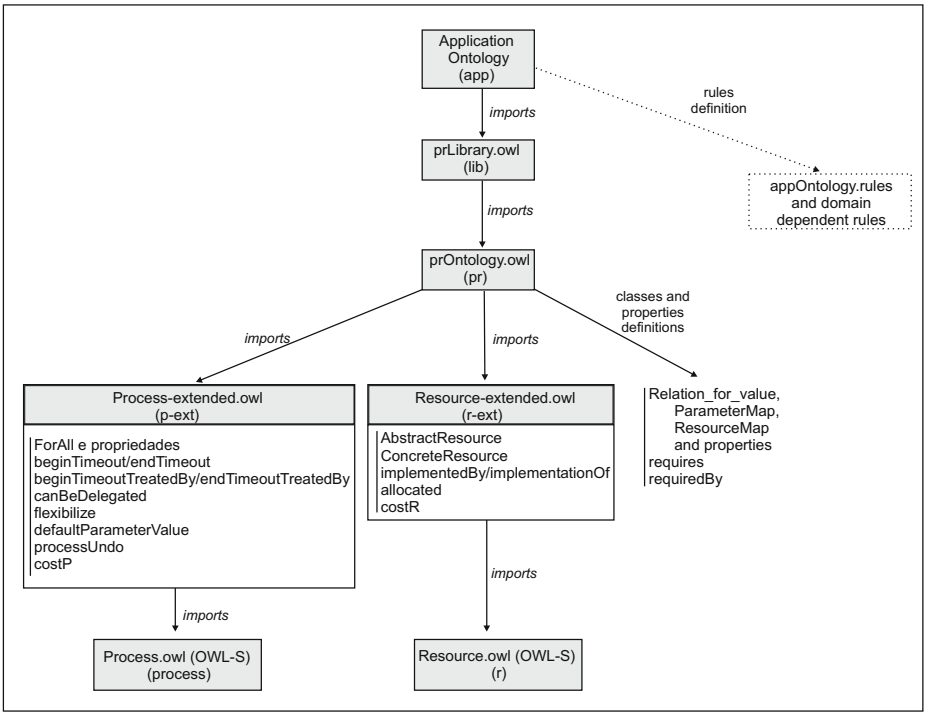


Fig. 2. *pr* ontology composition

lib: namespace of the *lib* library of processes and resources
app: namespace of the application ontology.

We refer the reader to [14] for the details of the *pr* ontology, since it is not the focus of the paper. We just mention that, to define relationship weights, we used ternary relations and created ontology rules to complement the definition with information that could not be readily represented using OWL. We also observe that, to clarify the semantics of OWL-S and the extensions we proposed, we defined an operational semantic for OWL-S, based on the concept of an abstract machine [14], briefly outlined the next section.

5 Transactional Model for Flexible Workflow Execution

This section presents the abstract machine and the transactional model that dictates the way process instances are executed. The specification of an operational semantic for OWL-S and its extensions through the definition of an abstract machine (AM) is justified for two reasons:

– the abstract machine uses very simple concepts, and is therefore easy to understand;

– the definition of the abstract machine can be made incrementally, through successive extensions that incorporate new language constructs.

5.1 Abstract Machine Without the Flexibilization Mechanism

The abstract machine *AM* does not include the flexibilization mechanism and consists of (see Figure 3):

– a *Controller*, responsible for instance management;
– an *Active Triggers Manager*, responsible for firing triggers. The triggers capture the semantics of atomic and composite processes, based on their control constructs, and guide the state transitions during instance execution. The *Active Triggers Manager* and the *Controller* exchange messages to guide instance execution.
– a *workspace area*, responsible for maintaining process instances. Each process instance P contains: the representation of the corresponding process definition p, called *process*; a unique id, called *instanceId*; the identification of its superprocess instance, called *superInstance*; a tuple with the current state and the associated timestamp; and a set of attributed values for the defined parameters, called *instance blackboard*;
– a *log*, responsible for registering the execution steps of the running instances.

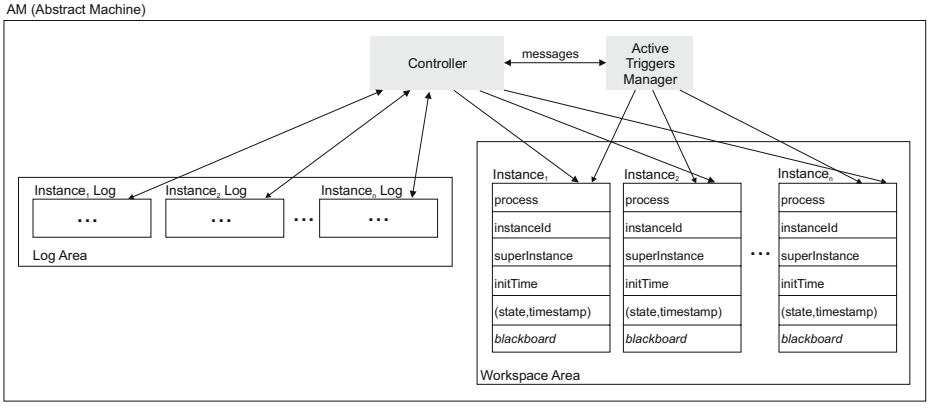


Fig. 3. Abstract machine (without the flexibilization mechanism)

When defining the *AM*, we assumed that a process can be selected for execution only when: (1) all input parameter values are known; (2) all preconditions evaluate to true; (3) all required resources are available. The dataflow in OWL-S is also unclear. This forced us to assume that, if a process p receives input values from the parameters of another process p', then p and p' must have a superprocess in common, i.e., they must belong to the same process hierarchy. Moreover, in the case of multiple instances of the same process, to capture the most recently terminated instance, each instance keeps the timestamp in which

the current state was reached. Finally, since the *AM* does not have strict control over the atomic actions of an instance, it does not automatically undo the effects of the abandoned actions (as in a database management system).

Figure 4 shows the states a process instance traverses. As depicted in the shaded areas, a process instance is *closed* if and only if its state is *Completed* or *Aborted*, and it is *open* if and only if its state is *Initiated*, *Running* or *PreparedToComplete*.

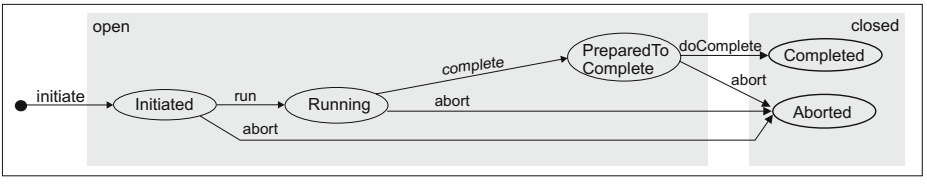


Fig. 4. State transition diagram of an instance P (without extensions)

By transactional behavior we mean that either all actions of an instance terminate correctly or all of them are abandoned and properly identified. This transactional behavior of an instance in the *AM* is formalized as two properties.

Let P be a composite process instance:

Property 1: If P terminates in the *Completed* state, then all direct subinstances of P also terminate in the *Completed* state.

Property 2: If P terminates in the *Aborted* state, then all direct subinstances of P also terminate in the *Aborted* state.

Property 1 immediately implies that, if P terminates in the *Completed* state, then all direct and indirect subinstances of P also terminate in that state (and analogously for Property 2).

5.2 Abstract Machine Extended with the Flexibilization Mechanism

5.2.1 Components of the Extended Abstract Machine

Figure 5 sketches the extended abstract machine (*EAM*) that includes the flexibilization mechanism.

The *EAM* contains a new component, the *Ontology Manager*, that is responsible for implementing the flexibilization mechanism and that exchanges messages with the *Controller*.

In the *EAM*, each instance P keeps additional information: the set of process subinstances of P; the instance type (if it is an instance executed in the predefined order in the superprocess definition, or if it was generated by the flexibilization mechanism, as explained later on); the default values for process parameters; the resource substitutions performed; the resource concretizations performed; the instance that caused the flexibilization of the current instance (if it is the case); the process concretizations performed; and the parameter mappings between the processes of related instances.

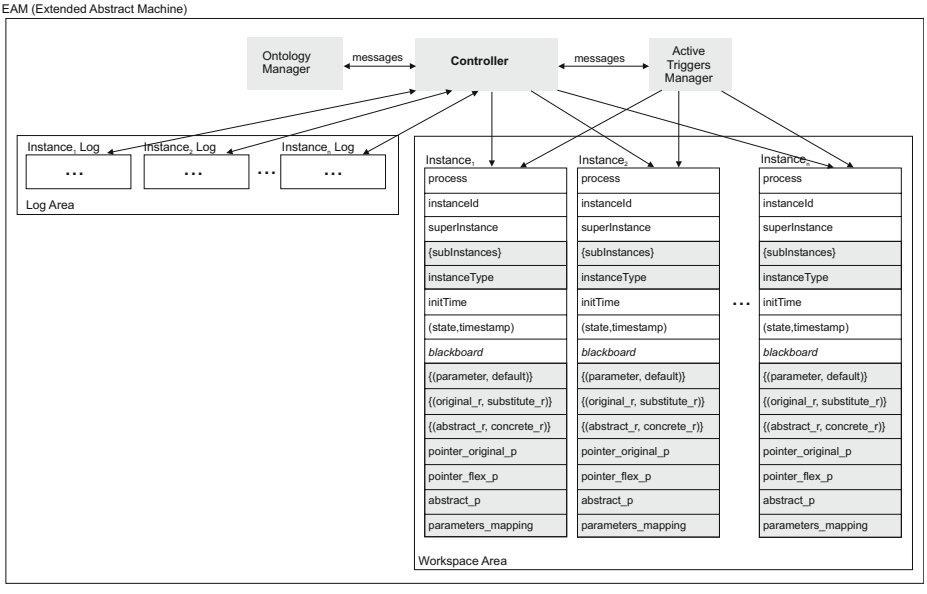


Fig. 5. Extended abstract machine (EAM)

Each instance P' has a type defined as follows:

"temporal": when P' is related to a process which was selected to treat a timeout exception raised by another instance;

"concretization": when P' is related to a concrete process which was selected to treat a concretization exception raised by an abstract process found in the superprocess definition;

"substitution": when P' is related to a process which was selected to treat an exception that called for the substitution of a process;

"undo": when P' is related to a process which was selected to undo the effects of an aborted atomic instance (*PreparedToAbort* state), as a result of a cancellation exception; and

"null": when the instance P' is related to a process p' statically defined by the control flow of the superprocess definition of p'.

In the above cases, we say that P' is a *flexibilization instance*. If the exception called for a process substitution, we say that P' *substituted* P. We suppose that any flexibilization process instance P' can also be flexibilized, except for instances which are undoing the effects of an atomic process instance which was aborted.

The *pointer_original_p* entry of P' identifies the instance P for which the instance P' is executing; the entry *pointer_flex_p* of instance P refers to P'. If the *instanceType* value of P' is "null", the *pointer_original_p* for P' is also null.

Let P be an instance of a process p. In the presence of flexibilization, the subinstances of P are not directly induced by the control structure of p. We consider as subinstances of P those instances listed in the entry *subInstances* of

P. This entry is updated each time the instance suffers flexibilization. In other words, flexibilization induces instance trees that are dynamically generated.

In the *EAM*, the triggers had to be redefined as follows:

- triggers must verify the current state of the subinstances registered in the {*subinstances*} entry of an instance P. Indeed, recall that, in the *EAM*, the subinstances of an instance P are the instances that in fact executed until termination.
- triggers must consider that it is possible to undo atomic instances and, consequently, composite instances.
- triggers responsible for completing the execution of an instance P must also check if any subinstance of P terminated using default parameter values for its output parameters.

The *Controller* of the *EAM* is also more sophisticated. For example, to find parameter values in the same data flow, when instance P' is the receiver of the values of instance P, the *Controller* must: (1) traverse all the instances list formed by the *pointer_flex_p* of instance P, until it reaches the last flexibilization instance; (2) compare its timestamp with the timestamp of all other flexibilization instances of P: (3) choose the most recently finished instance.

Note that, when an instance P is flexibilized by P', the *Controller* must obey the information about parameter mappings saved at instance P'.

The *EAM* also includes a log that registers information about instance execution and instance flexibilization. The log saves the states reached during instance execution, the input and output parameter values of the instance, the concrete resources and processes used as implementations of the abstract resources and processes found during instance execution, the default values used, and the flexibilization instances adopted, together with the resources and parameter mappings between the original instance and the instance run as a result of flexibilization.

5.2.2 State Transitions in the Extended Abstract Machine

Figure 6 shows the state transition diagram adapted to support flexibilization. It has seven new states: *BeginTimed-out*, *EndTimed-out*, *Skipped*, *PreparedToBy-Pass*, *Forced*, *PreparedToAbort* and *Undone*. The dotted arrows indicate that the corresponding messages are processed by the *Controller* during the execution of the superinstance of the instance P that had its state changed.

Let P be an instance of a process p. We explain the state transition diagram in Figure 6 as follows.

The *Controller* moves P to the *Initiated* state when it processes the *initiate* message for P.

The *Controller* may move P from the *Initiated* state to:

- *Running*, when it receives the *run* message for P, that signals that all preconditions of p are satisfied.
- *BeginTimed-out*, when P reaches its initialization timeout limit.

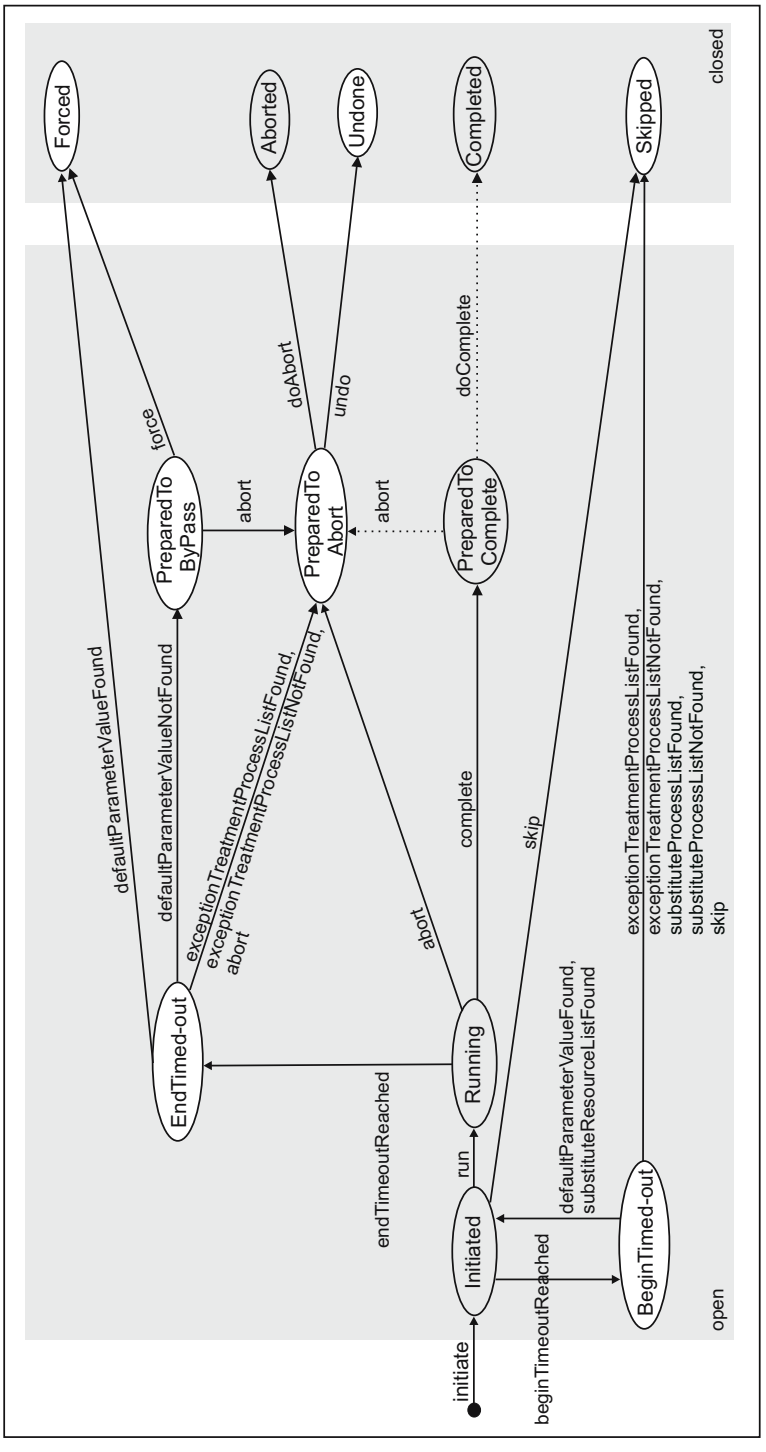


Fig. 6. Modified state transition diagram

The *Controller* may move P from the *BeginTimed-out* state to:

- *Skipped*, when the *Ontology Manager* finds a process p' that handles the initialization timeout exception for p, or if there is no way to execute P and the *Ontology Manager* finds no process p' that handles the initialization timeout exception for p.
- *Initiated*, when the *Ontology Manager* finds default parameter values for the missing value parameters, if this was reason for not starting the execution of P, and when the *Ontology Manager* finds alternative resources that can be allocated to P, if this was reason for not starting the execution of P.

The *Controller* may move P from the *Running* state to:

- *PreparedToComplete*, when it receives the *complete* message for P, that signals that P wishes to terminate, and it detected that no conflict occurred because of a possible use of default parameter value when P was in the *BeginTimed-out* state.
- *PreparedToAbort*, when it receives the *abort* message for P, that signals that P must be aborted, or when it receives the *complete* message for P, but it detects that a conflict occurred because of a possible use of default parameter value when P was in the *BeginTimed-out* state.
- *EndTimed-out*, if P reaches its termination timeout limit.

The *Controller* may move P from the *PreparedToComplete* state to:

- *PreparedToAbort*, when at least one of the subinstances of P reached the *PreparedToAbort* state.
- *Completed*, when all subinstances of P reached the *Completed* state or the *Forced* state.

The *Controller* may move P from the *EndTimed-out* state to:

- *PreparedToAbort*, when it receives the *abort* message for P, when the *Ontology Manager* finds a process p' that handles the termination timeout exception for p, and when the *Ontology Manager* finds no process p' that handles the termination timeout exception for p and P cannot use default values for its output parameters.
- *PreparedToByPass*, when the *Ontology Manager* finds no process p' that handles the termination timeout exception for p and no default values for the missing output parameters of P.
- *Forced*, when the *Ontology Manager* finds default values for the missing output parameters of P.

The *Controller* may move P from the *PreparedToByPass* state to:

- *Forced*, when the *Ontology Manager* finds default values for the missing output parameters of P among the output parameters of the superinstance of P.
- *PreparedToAbort*, otherwise.

The *Controller* may move P from the *PreparedToAbort* state to:

- *Undone*, when P is an atomic instance and the *Ontology Manager* finds at least one process p' that handles the cancellation exception for p, the instance P' of p' thereby created executes successfully and without flexibilization;
- *Aborted*, otherwise.

5.2.3 Transactional Behavior in the Extended Abstract Machine

We can also define properties that capture transactional behavior under flexibilization. Such properties indeed guided the definition of the triggers in the extended abstract machine.

Let P be a composite process instance:

Property 1': If P terminates in the *Completed* state, then all direct subinstances of P terminate in the *Completed* or *Forced* states.

Property 2: If P terminates in the *Aborted* state, then all direct subinstances of P terminate in the *Aborted*, *Undone* or *Forced* states and at least one subinstance, direct or not, of P terminates in the *Aborted* state.

Property 3: If P terminates in the *Undone* state, then all direct subinstances of P terminate in the *Undone* or *Forced* states.

Property 4: If P terminates in the *Forced* state, then all direct subinstances of P terminate in the *Aborted* or *Undone* states.

Note that, if P is an instance of a composite process, it may terminate in the *Abort* or *Undone* state, depending on the termination of its subinstances, as defined in Property 2 and in Property 4, respectively. If all subinstances of P terminated in the *Undone* or *Forced* states, then P also terminates in the *Undone* state.

6 Conclusions

Our approach to workflow flexibilization is based on an exception handling mechanism that allows the execution to proceed when otherwise it would have been stopped. The proposal was cast as a set of extensions to OWL-S, and is based on process and resource ontologies that capture the semantic information needed for the flexibilization mechanism. In this paper, we focused on the transactional behavior of a workflow instance, in the sense that it guarantees that either all actions executed by the instance terminate correctly or they are all abandoned.

As for future work, we may suggest to enhance the flexibilization mechanism with new features that: (1) consider environmental variables that change value independently of instance execution; (2) include the temporal analysis of lower cost process alternatives; (3) allow instance execution to be redone when conflicts are detected as consequence of the use of default parameter values.

Acknowledgment

This work was partially supported by CNPq, under grants 140600/01-9 and 550250/05-0.

References

1. Ilia Bider and Maxim Khomyakov. Is it Possible to Make Workflow Management Systems Flexible? Dynamical Systems Approach to Business Processes. In *Proceedings of the 6th International Workshop on Groupware (CRIWG' 2000)*, pages 138-141, Madiera, Portugal, October 2000.
2. G. Canals, C. Godart, F. Charoy, P. Molli, and H. Skaf. COO Approach to Support Cooperation in Software Developments. In *IEEE Proceedings in Software Engineering*, volume 145, pages 79-84, April/June 1998.
3. Fabio Casati, Stefano Ceri, Barbara Pernici, and Giuseppe Pozzi. Workflow Evolution. In Bernhard Thalheim, editor, *International Conference on Conceptual Modeling / the Entity Relationship Approach (15th ER' 96)*, pages 438-455, Cottbus, Germany, October 1996. Lecture Notes in Computer Science.
4. Wesley W. Chu, Q. Chen, and M. Merzbacher. *Studies in Logic and Computation 3: Nonstandard Queries and Nonstandard Answers*, volume 3, chapter CoBase: a Cooperative Database System, pages 41-72. Oxford University Press, New York, 1994. Edited by R. Demolombe and T. Imielinski.
5. Wesley W. Chu and Wenlei Mao. CoSent: a Cooperative Sentinel for Intelligent Information Systems, March 2000. Computer Science Department - University of California, LA.
6. Tom Freund and Tony Storey. Transactions in the World of Web Services, Part 1: an Overview of WS-Transaction and WS-Coordination. http://www-106.ibm.com/developerworks/library/ws-wstx1/, August 2002. IBM, DeveloperWorks.
7. Tom Freund and Tony Storey. Transactions in the World of Web Services, Part 2: an Overview of WS-Transaction and WS-Coordination. http://www-106.ibm.com/developerworks/library/ws-wstx2/, August 2002. IBM, DeveloperWorks.
8. Daniela Grigori, François Charoy, and Claude Gobart. Flexible Data Management and Execution to Support Cooperative Workflow: the COO Approach. In *Proceedings of the Third International Symposium on Cooperative Database Systems for Advanced Applications (CODAS 2001)*, pages 124-131, April 2001.
9. J. J. Halliday, S. K. Shrivastava, and S. M. Wheater. Flexible Workflow Management in the OPENflow System. In *Proceedings of the Fifth IEEE International Enterprise Distributed Object Computing Conference (EDOC '01)*, pages 82–92. IEEE, September 2001.
10. G. Joeris. Defining Flexible Workflow Execution Behaviors. In *Enterprise-wide and Cross-enterprise Workflow Management - Concepts, Systems, Applications, GI Workshop Proceedings - Informatik '99*, pages 49-55, 1999. Ulmer Informatik Berichte Nr. 99-07.
11. Peter Mangan and Shazia Sadiq. On Building Workflow Models for Flexible Processes. In *ACM International Conference Proceeding Series - Proceedings of the Thirteenth Australasian Conference on Database Technologies (ADC'2002)*, volume 5, pages 103-109, Melbourne, Australia, January/February 2002. Australian Computer Society, Inc. Darlinghurst.
12. David Martin, Mark Burstein, Jerry Hobbs, Ora Lassila, Drew McDermott, Sheila McIlraith, Srini Narayanan, Massimo Paolucci, Bijan Parsia, Terry Payne, Evren Sirin, Naveen Srinivasan, and Katia Sycara. OWL-S: Semantic Markup for Web Services. W3C Member Submission, November 2004. http://www.w3.org/Submission/2004/SUBM-OWL-S20041122/Overview.html.

13. Gary J. Nutt. The Evolution Toward Flexible Workflow Systems. In *Distributed Systems Engineering*, volume 3, pages 276-294, December 1996.
14. Tatiana A. S. C. Vieira. *Execução Flexível de Workflows*. PhD thesis, Department of Informatics - Pontifical Catholic University of Rio de Janeiro, Brazil, Rio de Janeiro, RJ - Brazil, August 2005. In Portuguese.
15. Mathias Weske. Flexible Modeling and Execution of Workflow Activities. In *Proceedings of the Thirty-First Hawaii International Conference on System Sciences*, volume 7, pages 713-722, January 1998.

An Open Architecture for Ontology-Enabled Content Management Systems: A Case Study in Managing Learning Objects

Duc Minh Le[1] and Lydia Lau[2]

[1] Department of Computing, Imperial College London, 180 Queen's Gate London SW7 2AZ, U.K.
dmle@doc.ic.ac.uk
[2] School of Computing, University of Leeds, Leeds LS2 9JT, U.K.
llau@comp.leeds.ac.uk

Abstract. An important goal of a content management system (CMS) is to acquire and organise content from different data sources in order to answer intelligently any ad-hoc requests from users as well as from peer systems. Existing commercial CMSs address this issue by deploying structured metadata (e.g. XML) to categorise content and produce search indices. Unfortunately, these metadata are not expressive enough to represent content for sophisticated searching. This paper presents an open architecture framework and a Java-based reference implementation for Ontology-enabled Content Management System. The reference implementation uses an open-source CMS called OpenCMS, the Protégé's OWL library, and RacerPro reasoning engine. The implemented system is a web-based management system for learning objects which were derived from the course and instructional materials used in several postgraduate taught courses. We believe that our OeCMS architecture and implementation would provide a strong platform for developing semantic web protals in general.

1 Introduction

A content management system (CMS) provides an integrated environment for an organisation to develop and manage presentable content in a wide variety of formats. A primary goal of a CMS is to automatically acquire and organise the content from different sources in order to answer intelligently any ad-hoc requests from users as well as from other peer systems. To achieve this, the following challenges would need to be addressed [13] [7]:

- To provide an extensible technology framework for connecting to and manipulating heterogeneous data sources; such as relational databases, web contents, documents as well as legacy contents
- To analyse, categorise and integrate (i.e. merge or map) contents in their corresponding domain contexts to reveal semantic structures

R. Meersman, Z. Tari et al. (Eds.): OTM 2006, LNCS 4275, pp. 772–790, 2006.

– To formalise these semantic structures using some form of descriptive rule set and serialise them in hierarchical concept repositories
– To provide intelligent access interfaces appropriate for people and/or computer software (agents) to search and make use of the underlying ontologies.

In this paper, we propose an open architecture for an ontology-enabled CMS (OeCMS) which is an attempt to address the challenges outlined above. Since ontology helps unambiguously define the meaning of *things* in terms of formal concepts with clearly defined relationships [10], it would serve well as a common integrator for the different types of content in a CMS. This framework could also be used for developing semantic web portals in general as their underlying concept is very similar to the combination of semantic web technologies and CMS. We will clearly show in this architecture how the required semantic web components could be developed and/or integrated to the content management infrastructure. The reference implementation of this architecture uses mostly Java open-source technologies at the core layer. This implementation was based on a case study for managing learning objects in a number of selected taught postgraduate courses at the University of Leeds. The rest of the paper is structured as follows. Section 2 reviews the related work in the areas of content management system and semantic web portals. Section 3 characterises an OeCMS in terms of a set of requirements and design criteria. This is then followed by Section 4 which discusses the general OeCMS architecture. Section 5 presents a reference implementation in Java with a set of essential functions. Finally, Section 6 concludes the paper.

2 Related Work

This section will highlight the current development in three main areas which are relavant to our research.

Content Management System. A content management system [5] is designed to support a content management cycle which includes the *creation* and *collection* of content, the *publication* of content for *access* by users and/or other systems and the *management* of these content. An important glue for these different CMS components is content metadata which are extracted (semi-)automatically from the content. However, a major limitation of the existing CMSs is inadequate support for expressing this metadata to allow for accurate retrieval of content. Recent research work have shown that ontology would provide a viable solution to this problem because it can help define the formal specification of a problem domain [10]. The most recent work which bear some similarities to ours is Infoflex [8]. This system also used semantic web technologies in content management systems but mainly for integrating existing CMSs while answering content search requests. This differs from our approach in that we propose that each CMS system should adopt an open architecture to tackle ontology-enabled content management.

Semantic Web Portals. Another related area is the development of a Semantic Web Portal (SWP). In essence, a semantic web portal [19] is a content management system to the extent that it also needs to acquire, organise and distribute information to interested users. A SWP, however, differs in that it deploys semantic web technologies such as ontology and related tools in order to improve the representation of content metadata and, therefore, information retrieval. Two recent work in this area have proposed a system infrastructure [24] and an architecture [19] for SWPs. In [19], a SWP is viewed as an enrichment of a three-layer web-based information portal by adding Semantic Web technologies as a sub-layer of its "Grounding Technologies" layer. However, it is not clear from this proposal how this addition would gel in with the existing portal technologies. Also, this paper did not discuss if any changes to its information processing (i.e. middle) layer would be needed in order to utilise the semantic web technologies. In [24], a detailed technical infrastructure for SWPs, called Semantic Web application server, was discussed. The main advantage of this architecture lied in its modularity which considered all software modules as self-contained components with a standard system interface. This KAON SERVER [25] application server was implemented in Java based on JBoss [9] open-source application server. However, this system only focussed on functionalities at the infrastructure layer and did not address the content management cycle. Although the proposed system in [24] represents an alternative application server component to that of our architecture, its all-in-one design makes it less flexible compared to our approach (see Section 5.1).

Web-Based Learning Systems. These represent the class of e-learning systems that cover the scope of our case study. These systems may tackle a range of issues including the delivery of instructional materials and the management of learning objects about students and assessment. Research in this area have begun exploring the use of ontology for describing these learning objects. It was argued that ontology can help improve the representation and organisation of teaching resources and student profiles [20] [23] [26]. However, these research lacked openness in their underlying technical platforms.

In brief, no research work reported so far in the literature have explored the design and development of semantic web portals from the perspective of a content management system. We argue that it is advantageous to follow this approach given the similarities between these portals and CMSs and that it is possible to develop the semantic web extensions for the content management cycle in a CMS.

3 Requirements and Design Criteria of an Ontology-Enabled CMS

As suggested earlier, the OeCMS aims to improve the search results by extending the capability of a CMS with semantic web technologies. Hence, it should

meet the requirements of the traditional CMS and those of the Semantic Web technologies.

3.1 Requirements

The main requirements of an OeCMS are:

- **Multiple representation** [24] to support multiple ontology languages (e.g. RDF and OWL Lite/DL/Full [2]).
- **Semantic content syndication** to (semi-)automatically acquire contents from different data sources, extract the key concepts and establish the relationships between these concepts.
- **Ontology Mapping** [24] which is a pre-requisite of semantic content syndication. This is a fast-evolving and challenging research area of ontology which aims to achieve some degree of automation in matching and merging ontologies from potentially different but overlapping domains.
- **Integrated ontology engineering method** which consists of techniques and tools that allow the CMS development team to identify, design and build a knowledgebase for a problem domain and to seemlessly deploy and maintain this knowledgebase in the system.
- **Integrated access** to harness the expressiveness of semantic content for the purpose of producing system interfaces (e.g. for navigation or searching) that can greatly increase the accuracy and relevancy of the requested information from both users and peer systems.
- **Ease of use** [24] to provide at least a compatible level of ease-of-use of the ontology interface for existing users of an CMS.
- **Inferencing and Verification** [24] which are two primary functionalities of the ontology reasoning engine in the OeCMS. Through reasoning, the system is able to search the knowledgebase in a finite time for *instances* matching a query. Also through reasoning, the system is able to compute any inconsistencies or anomalies that exist in the ontology.
- **Access Control and Versioning** [24] to manage multiple user access; to detect and monitor user's access to content documents and the knowledgebase. While access control to content documents is a built-in feature of the underlying CMS, it still remains an open quesion as to how to do the same for the knowledge-base. Ontology versioning is a background process for supporting the evolution of an ontology knowledgebase. Different versions of the knowledgebase are maintained persistently to allow for easy roll-back to a previous version when needed.

3.2 Design Criteria

Since ontology and related tools are continuously evolving, it is important to have an adaptable OeCMS architecture which can scale well with changes. The following list summarises the main criteria ([15] [24]) of such a design:

- **Modularity.** The system is component-based with self-contained software modules to allow for maximum development and maintenance flexibility.
- **Interoperability.** The ability to communicate with other systems via open-standard protocols such as XML [2], Web Service Definition Language [6], OWL-S [21], and Description Logic Implementation Group (DIG) for ontology reasoning [4].
- **Manageability.** The ability to streamline the management of content, ontology and system functionalities.
- **Scalability.** The ability to cope with an ever-increasing number of requests and an increasing demand of resources (e.g. processing, storage) without major changes to the architecture.
- **Integrability.** The ability to upgrade the system with new functionalities without major changes to the overall architecture.

4 An Open Architecture for Ontology-Enabled CMS

The proposed OeCMS, see Figure 1, is based on a layered approach to clearly define the responsibilities and services of each distinct class of components. The clear advantages of this layering are modularity, interoperability and easy integration. The following sub-sections will discuss each layer in greater detail.

4.1 The Web Interface Layer

This layer provides a uniform web-based interface for users to access and manage both content documents and their ontology-enabled semantics. The web technologies involved, e.g. HTML/XML and client-side scripting, should conform to W3C standards [33]. The major technical challenge at this layer is in the building of a dynamic and user-friendly view of an ontology using the rather restrictive set of browser technologies (e.g. HTML and Javascript). The difficulty lies in the ability to draw an incremental picture of the ontology in order to hide from the user the complexity inherent in the many-to-many relationships between ontological concepts. A diagramatic view of a modest ontology is sometimes hard for a novice user to understand. In addition to the use of HTML/Javascript, several emerging technologies such as TouchGraph [30] and Formal Concept Analysis [11] have been proposed as alternatives for the ontology view component of this layer. However, these tools require browser plug-ins (e.g. Java Applet) which are not always accessible to the users.

4.2 The Semantic Content Management Layer

This layer consists of two main components: content management and ontology management. The content management component provides functionalities to support the four main phases of a typical content management cycle [5]: (1) create content (2) publish content (3) access content (from user or other systems) and (4) management content. Each phase in this cycle is supported by one or more primitive ontology management functions. The primitive functions shown

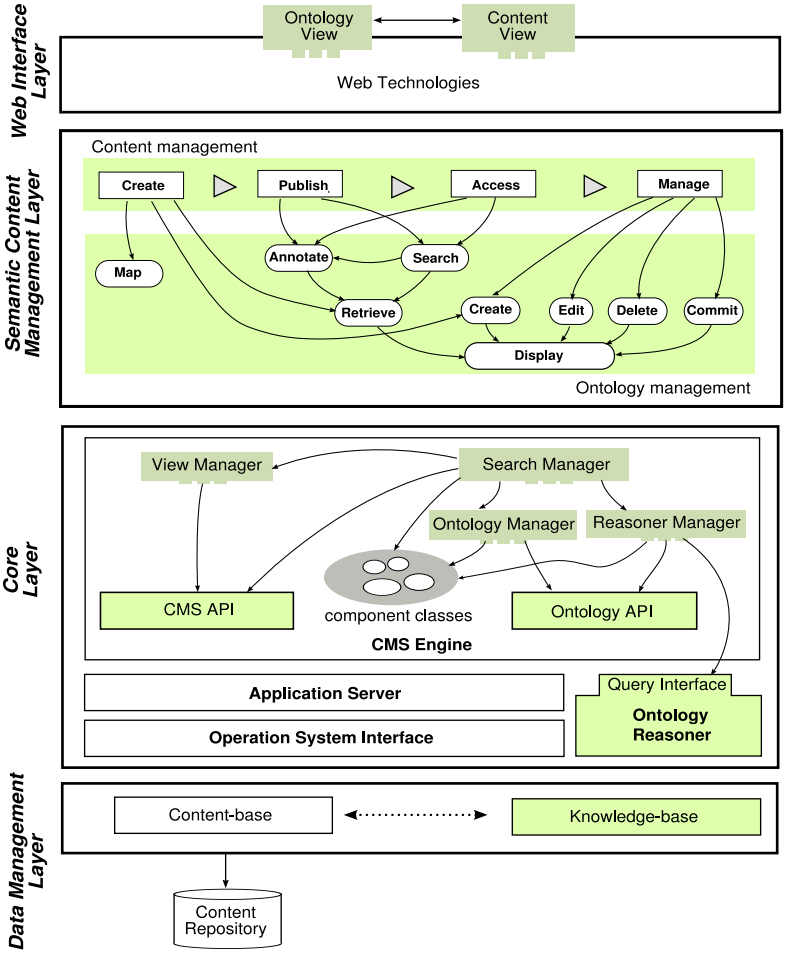


Fig. 1. The OeCMS Architecture

in Figure 1 are not meant to be exhaustive but they form the core of the ontology management:

– *map:* when a content document is defined or uploaded to the system, one or more ontology instances are created and mapped to elements of this document. A typical mapping technique is to store the *url* of a content document with each ontology instance defined for that document so that the system can later *retrieve* all associated content documents for a given set of ontology instances. It is worth noting that here we define a generic map function for all types of source documents including raw (typically binary) and (semi-/un)structured. Although the techniques for (semi-/un)structured types are well documented (e.g. [1] [17] [18] [22]), those for raw type remain largely unexplored.

- *retrieve*: retrieves the ontology instances related to a content document so that the system can automatically *annotate* the document's view with additional information.
- *annotate:* this function produces a component view of related ontology instances to a content document. This function depends on the *retrieve* function to access information about the properties of the ontology instances.
- *search:* this important function searches the ontology knowledgebase in the "Data Management" layer for instances matching a specific criterion. This function naturally depends on *retrieve* and *annotate* to generate the results.
- *create/edit/delete/commit:* these are the standard functions for maintaining the ontology knowledgebase. They can be applied to all types of ontological *objects* including concept, property and instance.

To increase modularity, these primitive functions rely on the manager components of the core layer for detailed algorithmic implementations. The next subsection will discuss the structure of the core layer which enables this modularity.

4.3 The Core Layer

This core layer has three basic sub-layers:

- **operating system interface:** this sub-layer implements the basic system functionalities (e.g. creating network sockets, accessing local/networked files, etc.) and make them available as service to upper layers.
- **application server:** this sub-layer represents the environment in which an OeCMS can easily be deployed and managed. This layer leverages the existing application servers many of which are open-source (e.g Tomcat [28] and JBoss [9]). In practice, the operating system interface may be implemented as part of this sub-layer.
- **content management engine:** supports both content and ontology management functionalities of the semantic content management layer. This engine is typically implemented as a bare-bone CMS system which provides a CMS API for developers access to content management functionalities. More importantly, it must support the deployment of an ontology API, an ontology reasoner and a set of custom-built ontology management functions:

  ontology API: implements the state-of-the-art ontology language specifications (e.g. RDF/OWL) and provides developers with the functionalities needed to design, develop and test ontologies. This API also provides a standard programming interface for interacting with the *ontology reasoner*.

  ontology reasoner: implements the logics behind ontology languages and to provide the facility needed for answering such complex queries as '*Find all lecturers who teach subject Information System Enginering in 2005*'. Note from Figure 1 that the separation of ontology reasoner from other core sublayers is only logical to reflect the fact that there are two possible methods of integrating a reasoner engine to the core layer: (1) as an independent application accessible via a network programming interface (e.g. RacerPro [12])

and (2) as an embedded software component accessible via a programming interface (possible with future reasoners).

ontology management components: are implemented as a set of managers and component classes. These software components are not provided by the CM engine and, thus, developed as plugged-in modules of this engine. Figure 1 shows the dependencies of these components on the core sub-layers.

- Search Manager: this is a self-contained search component that encapsulates the different ontology-related search algorithms. The main objective of this component is to perform an integrated search which leverages the powerful text expression provided by existing text-based search and the reasoning capability provided by ontology language.
- Ontology Manager: manages the run-time cycle of an ontology knowledgebase. For example, this component is responsible for loading the knowledgebase and controlling run-time application access to ensure the the knowledgebase integrity.
- Reasoner Manager: is responsible for communication with the ontology reasoner via a standard query interface such as DIG. In particular, the reasoner manager defines standard function calls for instructing the reasoner engine to load a given ontology and for subsequently executing an ontology query on the reasoner.
- View Manager: is responsible for dynamically generating the web interfaces for ontology-based and content-based tasks.
- Component Classes: these are atomic classes that are generated (either through coding or code engineering) to encapsulate the state and behaviours of the primitive domain entities. An example of a component class would be *Department* which represents the department concept in the knowledge-base.

4.4 The Data Management Layer

This is the bottom layer of the architecture which is responsible for database and/or file access. This layer provides a common access interface for both content base and ontology knowledgebase. The content-base is typically stored in a relational database for easy management and querying, whilst the ontology knowledgebase can be serialised into a database or to a file for convenient re-use by other systems.

5 A Reference Implementation in Java

The OeCMS architecture discussed in Section 4 was used in developing a web-based system, called WOICMS, for managing learning objects metadata at the University of Leeds. This section will discuss this prototype implementation which demonstrates the performance of the essential semantic content management functions of the OeCMS architecture (see Section 4.2). The prototype requirements are:

- the creation of a learning-object ontology for teaching materials used in a number of selected courses
- the development a web-based navigation interface assisted by the learning object ontology
- the development of two ontology-integrated search methods based on the learning object ontology

The choice of an implementation platform will be explained first because this dictates the development environment and deployment strategy for the WOICMS prototype. This is then followed by separate sub-sections on the implementation for each of the above requirements.

5.1 The Java-Based Implementation Platform

The implementation platform consists of the following software components which form the core layer of the OeCMS architecture: the application server with its operating system interface, the CMS API, the ontology API and the reasoner engine. Again, the design criteria for this platform is *openness* so that it can be used not only for adding new functions to the prototype in later stages but also for developing semantic web protals in general. We decided to use Java as the base platform which supports the following:

- Tomcat as an open-source application server and operating system interface
- OpenCMS [32] as an open-source CMS engine (deployed in Tomcat)
- Protégé/OWL [14] as an open-source ontology development API
- RacerPro [12] as the DIG-compliant reasoner engine

The combination of OpenCMS and Tomcat provides a favourable technical infrastructure (compared to others such as Zope [31]) because it naturally supports the deployment of the ontology development tool. In addition, the deployment of OpenCMS as a web application in Tomcat provides an extra, infrastructure-level modularity. We used Protégé-OWL plug-in which allows the developers to develop and test concepts and instances in OWL-DL. This also provides an API for writing Java code to manage ontology and to interact with RacerPro through a TCP-based programming interface called JRacer. The RacerPro engine provides OWL-based inference and verfication services through its DIG-compliant programming interface.

5.2 The WOICMS Implementation

Having concluded an implementation platform, let us now briefly discuss the development of the essential functions of the WOICMS system prototype using this platform. Firstly a virtual site was created in OpenCMS for the WOICMS's application space which represents the semantic content management layer of the OeCMS architecture. Next the manager and component classes of the core layer and the management functions of the semantic content management layer were developed using the OpenCMS and Protégé-OWL APIs. The manager and

component classes were deployed to OpenCMS, whilst the semantic content management functions were developed as JSP pages and deployed to the WOICMS virtual site.

Note also that each web page presented to the user at the web interface layer is generated dynamically by a functional JSP page at the semantic content management layer. At its final processing stage, this JSP page invokes the appropriate ViewManager's methods to produce the component views for the required web page from a set of pre-defined HTML/Javascript templates. A template can either be specifically tailored to one web page or be a generic view component such as URL link, list box, text box, table and the like. The decomposition of our templates down to the HTML-tag level allows us to conveniently annotate individual view components of a result web page with ontology-enabled content. Refer to Section 5.4 for an examplary use of templates.

5.3 Developing the Learning Object Ontology

The domain for the learning object ontology covered information used in a number of selected postgraduate taught courses. For each course, we were interested in knowing about its syllabus, teaching staff, instructional materials and references to other relavant resources. Course materials included web pages and binary files such as Powerpoint slides and Word documents. The preliminary design of the learning object ontology was carried out in the Protégé ontology editor tool [16] using the OWL-DL language as it provides more expressive power over its predecessors. Figure 2 shows the conceptual view of this ontology. For brevity, the super-parent concept, $ModuleElement$, is not shown in this figure. Let us take the relationship set between $Reference$ and its sub-concepts as an example. This set is characterised by the following rules:

- A $Reference$ instance must be of type $PrintedPublication$, $OnlinePublication$ or $HybridPublication$ (i.e. both printed and online)
- A $Reference$ instance may contain links to a number of other reference instances
- $Book$ and $Journal$ are disjoint sub-types of $Publication$, i.e. a $Book$ instance is distinctively different from all other $Journal$ instances.

Using the DL-based syntax [3], this set of inter- and subsumed relationships can be defined in the Protégé-OWL editor as follows ($\sqsubseteq$ implies *necessary* and $\equiv$ implies *necessary and sufficient*):

$$
\begin{aligned}
Reference &\sqsubseteq ModuleElement \\
&\quad \sqcap (\exists\, belongsTo\, ModuleSkeleton) \\
&\quad \sqcap (\exists\, authoredBy\, Author) \\
&\quad \sqcap (\exists\, hasReference\, Reference) \\
Publication &\sqsubseteq Reference \\
Publication &\equiv OnlinePublication \sqcup PrintedPublication \\
&\quad \sqcup HybridPublication
\end{aligned}
$$

$$OnlinePublication \sqsubseteq Publication \sqcap (\exists\, webAddress\, WebAddresss)$$
$$PrintedPublication \sqsubseteq Publication \sqcap (\exists\, publisher\, PublishingHouse)$$
$$HybridPublication \equiv OnlinePublication \sqcap PrintedPublication$$
$$Book \sqsubseteq Publication \sqcap (\neg\, Journal)$$
$$Journal \sqsubseteq Publication \sqcap (\neg\, Book)$$

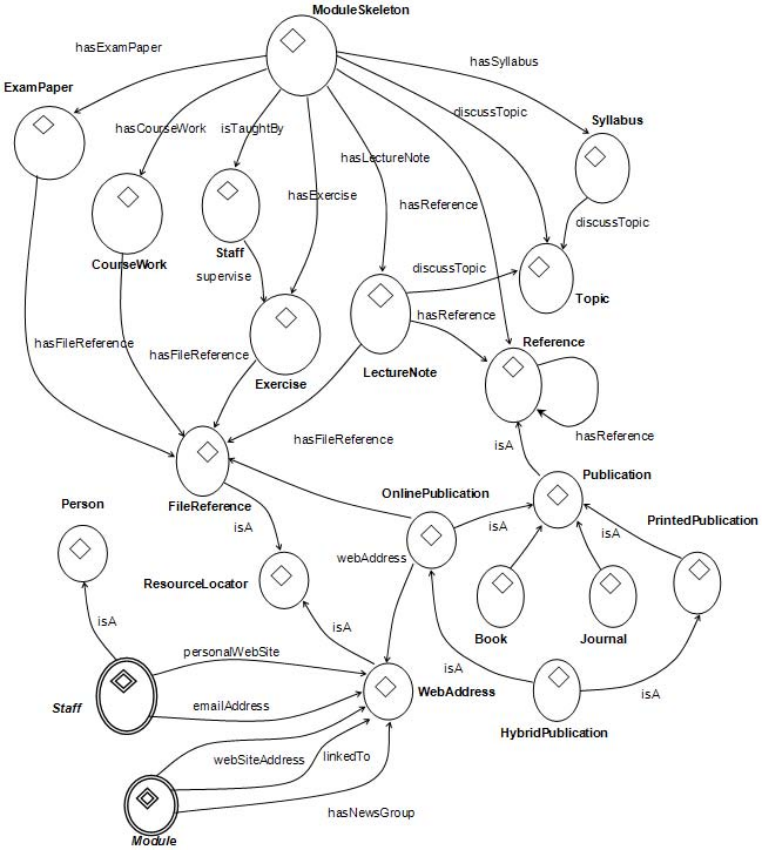


Fig. 2. The Conceptual View of Learning Object Ontology

After the initial design, the LearningObject ontology was serialised into a file using the RDF/XML syntax and later deployed in OpenCMS. At application start-up, the ontology file would be pre-loaded into memory and also onto the RacerPro server. Ontology instance(s) were created and mapped to each content document through the ontology editor interface in Figure 3. Each editor's view is dynamically generated by the ViewManager for an ontology instance of a given

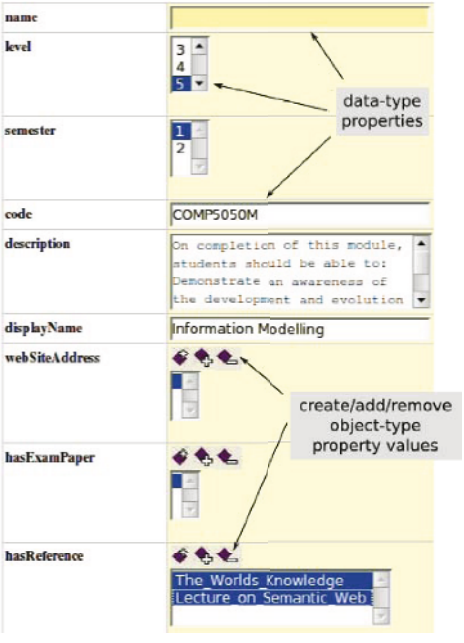


Fig. 3. The Web-Based Ontology Editor

concept. In this example, the ontology instance is titled 'Information Modelling' and the concept is ModuleSkeleton. A useful feature of the editor is that the view component of an object-type property is generated with a set of Javascript-enabled controls which are used to manage the property's *range*.

5.4 Ontology-Assisted Navigation

A practical and widely used method to visualise an ontology using the current web browsers technologies is to render the ontology subsumption hierarchy as a tree and uses it to assist with content navigation. Figure 4 illustrates such an interface developed in this project. Technically, this interface is consisted of two frames working in coordination. The left-frame contains two navigation controls: (1) a concept search box which helps retrieve a concept's definition from the knowledgebase and (2) an ontology tree to guide user's input. Actions performed on the left-frame are processed by the corresponding JSP pages at the server who invoke the appropriate methods provided by the manager classes at the core layer to produce the result view. This result view is then pushed back to the right-frame for display. The main advantage of using a frame-based design for this interface is the significant reduction in response time by avoiding the re-contruction of the ontology tree everytime a new web page is displayed. The sequence diagram illustrated in Figure 5

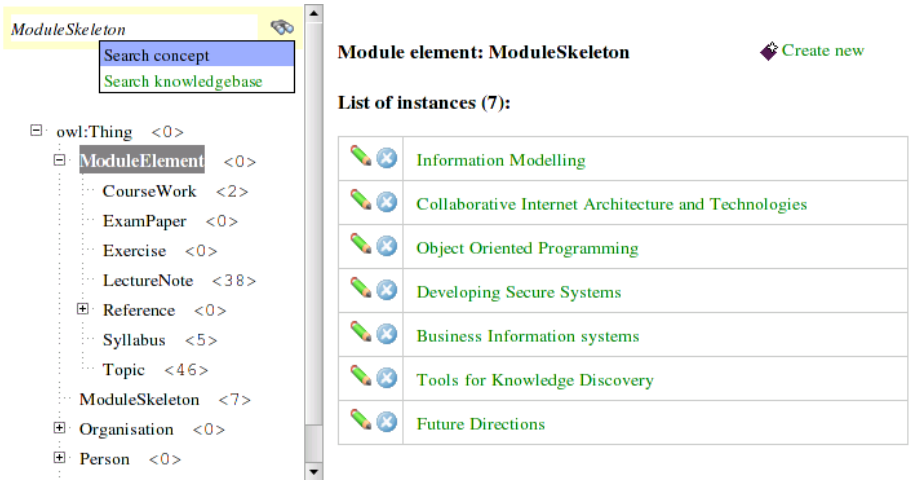


Fig. 4. The Ontolog-Assisted Navigation Interface

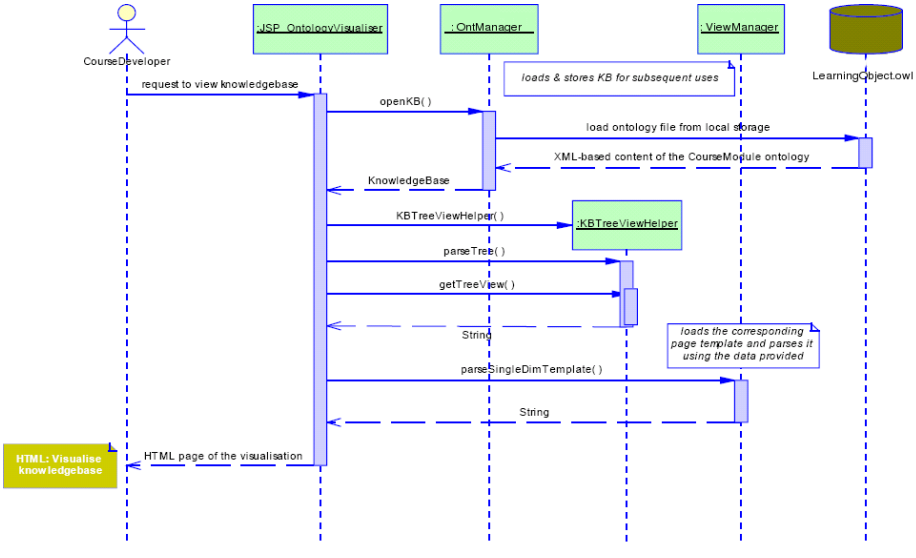


Fig. 5. The Sequence Diagram for Ontology-Assisted Navigation

explains the interaction between different components in creating the ontology tree:

– upon receiving a request to view the ontology, the `OntologyVisualiser.jsp` page invokes the `OntManager.openKB()` method to load the OWL-DL ontology (`LearningObject.owl`) into memory (this is performed once)

– the `OntologyVisualiser.jsp` then instantiates a singleton object of the class `KBTreeViewHelper` with the loaded `LearningObject` ontology. This is followed by an invocation of the `KBTreeViewHelper.parseTree()` method which recursively traverses the ontology's sub-sumption hierarchy and constructs the corresponding tree view as shown in the left-frame of Figure 4. This tree is rooted at the `owl:Thing` concept which is the default parent of all OWL concepts.
– the `OntologyVisualiser.jsp` retrieves the ontology tree view and calls the `ViewManager.parseSingleDimTemplate()` method which operates on the the page template of the left-frame to produce the final view before pushing it back to the client's browser for display

5.5 Ontology-Integrated Search

The motivation behind ontology-integrated search are the capability gaps between the native ontology search and reasoning and text-based search. The first gap is between the rather restrictive text-matching facility currently supported by ontology search and the rich regular expression provided by text-based search methods (e.g Lucene [27]). The second gap is between the lack of support for a *formal* (i.e. unambiguous) definition of search queries in text-based search and the reasoning capability of the ontology language. Therefore, ontology-integrated search methods were devised to bridge these gaps by leveraging:

– the expressiveness of the ontology language for designing formal search queries
– the reasoning capability of the ontology language for defining a semantic search scope for content documents
– the rich regular expression of text-based search to retrieve the relavant content documents from the semantic search scope

In our project, two ontology-integrated search methods were implemented (refer to Table 2 for their psuedocodes):

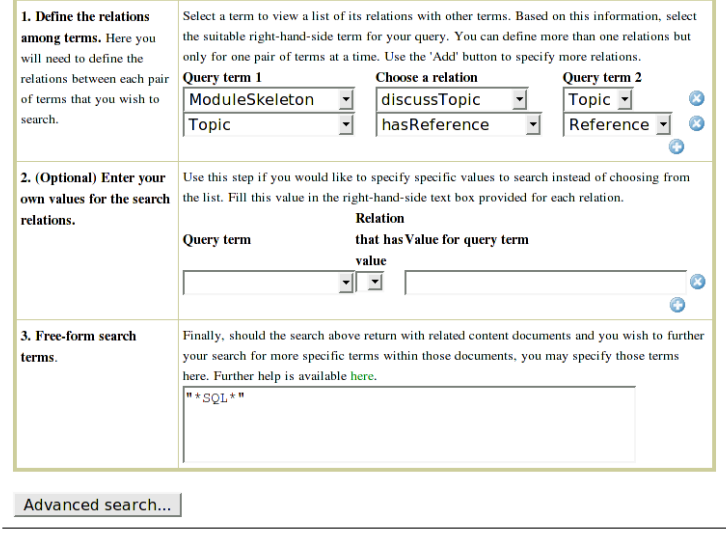
– **Guided search:** has an incremental design interface for generating on-the-fly RacerPro's queries in nRQL [29] language (refer to Table 1 for an example). The result of executing this query on the reasoning engine produces a semantic-search scope in which text-based search is performed. This method uses RacerPro and the Lucene search API (included in the OpenCMS API)
– **Parameterised search:** the query interface uses the simple text-based search provided by the ontology API to locate a branch (i.e. a concept and its sub-concepts) of the ontology sub-sumption tree to scope the text-based search that follows. This method uses the Protégé and Lucene search API

We evaluated the usefulness of these two search methods by uploading the contents of three courses web sites on to the system. Based on the overall knowledge about these courses, we then designed a set of 18 query pairs for 18 different query requirements. Each pair had a query in nRQL language and a query in Lucene query format both seeking answer for the same requirement. The query

Table 1. An Example of Evaluation Queries

Integrated Search	**Content Search**

- define the search scope using nRQL query:
 	
`(retrieve (?ModuleSkeleton ?Topic ?Reference )`
`(and (?ModuleSkeleton |ModuleSkeleton|)`
`(?ModuleSkeleton ?Topic |discussTopic|)`
`(?Topic ?Reference |hasReference|)))`
 	
- search content documents using text expression "*SQL*"
- The Guided Search Interface
	Use the following text expression to search:
	
	`'module' AND ('topic' AND "*SQL*")`



requirements were chosen to have different levels of complexity. For example, the query pair shown in Table 1 answers to the requirement *Find all course modules that discuss topics about SQL?*. Also illustrated in this table is the guided search interface which defines the scope of the semantic search. This interface has two parts: (1) the top part, which covers steps 1 and 2, is a wizard-like dialog which helps the user design an nRQL query for the search scope and (2) the bottom part (step 3) has a text field for entering an optional text expression for the subsequent text-based search. The evaluation result showed that on average the relavancy of content documents returned from the integrated search methods was above 90% compared to around 50% relavancy of those returned from the normal text-based search.

Table 2. The Psuedocodes of Two Search Methods

Guided Search

```
OWL_M = load OWL model from
file 'LearningObject.owl'
Initialise RacerPro with owlModel
as follows:
  initialise reasoner_manager with
  OWL_M
  Connect reasoner_manager to Racer
  server's URL via HTTP port 8080
  Send OWL_M
Construct matched instance set
M_I_S as follows:
  C_S = concepts selected by user
  P_S = properties selected by user
  V_S = property values specified
  by user
  rql_query = generate RQL query
  from (C_S, P_S, V_S)
  Execute rql_query as follows:
    Connect to Racer via TCP
    port 8088
    result_string = send rql_query
    Parse result_string into M_I_S
Retrieve file reference set
F_R_S from M_I_S as follows:
  For each instance I in M_I_S
    O_P_S = set of OWL object
    properties of I
    For each property P in O_P_S
      R = range of P
      If R equals 'FileReference'
        P_V_S = set of property
        values of P
        Put P_V_S to F_R_S
Retrieve content document set
C_D_S as follows:
  Q = content query specified
  by user
  C_D_S = Execute Lucene search
  of Q
Filter content result set
C_R_S from C_D_S using F_R_S
as follows:
  For each file url F in C_D_S
    If F exists in F_R_S
      F_O = file object (F) in F_R_S
      F_M = Construct file_metadata
      of F_O
      Put F_M to C_R_S

Display C_R_S
```

Parameterised Search

```
OWL_M = load OWL model from file
'LearningObject.owl'
Initialise Racer with OWL_M as follows:
  initialise reasoner_manager with
  OWL_M
  Connect reasoner_manager to
  Racer server's URL via HTTP port 8080
  Send OWL_M
C = concept name specified by user
Pt = knowledge-base search pattern
specified by user
Q = content query specified by user
C_S = set of concepts in the concept tree
of C in OWL_M
Construct instance set I_S containing
instances matching Pt in OWL_M as follows:
  P_S = set of all user-defined RDF
  properties from OWL_M
  For each property P in P_S
    If value V of P matches Pt
      P_I_S = set of instances in OWL_M
      that owns P
      For each instance I in P_I_S
        If I not exists in I_S
          Put I to I_S
Filter matched instance set M_I_S from
I_S using C_S as follows:
  For each instance I in I_S
    D_C_S = set of concepts which are the
    direct parents of concept of I
    If D_C_S intersects with C_S
      Put I to M_I_S
Retrieve file reference set F_R_S from M_I_S
as in 'Guided Search' method
Retrieve content document set C_D_S
and filter C_R_S from C_D_S
using F_R_S as in 'Guided Search' method
Display C_D_S
```

6 Summary and Future Work

This paper presented an open architectural framework for the class of ontology-enabled content management system. This architecture has a semantic content management layer which provides the functionalities for developing both content documents and their semantic descriptions on the system. This is supported by a core layer which has a modular design to leverage the traditional content management engine and ontology development tools. A reference implementation based on a Java technical platform consisting of proven open-source components was also discussed. This implementation showed that it is possible to extend the support for a traditional content management cycle in a CMS with a set of primitive ontology management functions. These ontology management functions help construct the content semantics in terms of the formal ontological concepts. When expressed in a powerful language such as OWL, more intelligent interface could be designed for user to navigate and access the content. More importantly, our implementation showed that a number of search methods could be developed to leverage the benefits of the content ontology to deliver more accurate results to users. The proposed OeCMS architecture and its implementation together would provide a strong technical platform for developing semantic web portals in general. An extension to this work would be the integration of (semantic) web service technologies to the core layer to support an open model of distributed collaboration with other systems.

References

1. K. Ahmad and L. Gillam. Automatic ontology extraction from unstructured texts. In *Proceedings of the ODBASE 2005*, pages 1330–1346, 2005.
2. G. Antoniou and F. van Harmelen. *A Semantic Web Primer*. The MIT Press, Cambridge, Massachusetts, 2004.
3. F. Baader, D. Calvanese, D. McGuineness, D. Nardi, and P. Patel-Schneider. *The Description Logic Handbook*. Cambridge University Press, 2003.
4. S. Bechhofer, R. Moller, and P. Crowther. The DIG description logic interface. In *Proc. of International Workshop on Description Logics (DL2003)*, San Diego, California, USA, 2003.
5. B. Boiko. *Content Management Bible*. Wiley Publishing, New York, 1st edition, 2002.
6. E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana. Web Services Description Language (WSDL) 1.1. W3C, 2001. Available from http://www.w3.org/TR/wsdl.
7. D. Fensel. Semantic Web application areas. In *Proceedings of the 7th International Applications of Natural Language to Information Systems*, Stockholm, Sweden, 2002.
8. N. Fernandez-Garcia, L. Sanchez-Fernandez, and J. Villamor-Lugo. Next generation web technologies in content management. In *Proceedings of the WWW2004 Conference*, New York, USA, 2004.
9. M. Fleury and F. Reverbel. The JBoss extensible server. In *Proceedings of the International Middleware Conference*, 2003.

10. T. R. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5:199–220, 1993.
11. J. Guoqian and R. S. Harold. FCA view tab, 2004. Available from http://info.med.hokudai.ac.jp/fca/fcaviewtab/fcaviewtab.html.
12. V. Haarslev and R. Möller. Racer: An OWL reasoning agent for the Semantic Web. In *Proceedings of the International Workshop on Applications, Products and Services of Web-based Support Systems, in conjunction with the 2003 IEEE/WIC International Conference on Web Intelligence*, pages 91–95, Halifax, Canada, October 2003.
13. J. Hartmann and Y. Sure. Semantic Web challenge: An infrastructure for scalable, reliable, Semantic Portals. *IEEE Intelligent Systems*, 19(3):58–65, May 2004.
14. K. Holger *et al*. The Protégé OWL plugin: An open development environment for Semantic Web applications. In *Third International Semantic Web Conference - ISWC 2004*, Hiroshima, Japan, 2004.
15. R. Kazman *et al*. ATAM: Method for architecture evaluation. Technical report, Carnegie Mellon University, 2000.
16. H. Knublauch, M. A. Musen, and A. L. Rector. Editing description logic ontologies with the Protégé OWL plugin. In *International Workshop on Description Logics - DL2004*, Whistler, BC, Canada, 2004.
17. L. Kof. An application of natural language processing to domain modelling – Two case studies. *International Journal on Computer Systems Science Engineering*, 20(1):37–52, 2005.
18. N. Kozlova. Automatic ontology extraction for document classification. Master's thesis, Computer Science Department, Saarland University, February 2005.
19. H. Lausen *et al*. Semantic Web Portals - state of the art survey. Technical report, DERI, 2004. Available from http://www.deri.ie/publications/techpapers/documents/DERI-TR-2004-04-03.pdf.
20. C. W. Lo, K. T. Ng, and Q. Lu. CJK knowledge management in multi-agent m-learning system. In *Proceedings of the First International Conference on Machine Learning and Cybernetics, IEEE*, 2002.
21. D. Martin *et al*. Bringing semantics to web services: The OWL-S approach. In *Proceedings of the First International Workshop on Semantic Web Services and Web Process Composition (SWSWPC 2004)*, San Diego, California, USA, 2004.
22. G. Modica. A framework for automatic ontology generation from autonomous web applications. Master's thesis, Department of Computer Science, Mississippi State University, December 2002.
23. D. L. Musa *et al*. Sharing learner profile through an ontology and web services. In *Proceedings of the 15th International Workshop on Database and Expert Systems Applications, IEEE*, 2004.
24. D. Oberle, S. Staab, R. Studer, and R. Volz. Supporting application development in the Semantic Web. *ACM Transactions on Internet Technology, TOIT*, 5(2), 2005.
25. D. Oberle, S. Staab, and R. Volz. An application server for the Semantic Web. In *Proceedings of the 13th International WWW Conference*, 2004.
26. D. Woelk and P. Lefrere. Technology for performance-based lifelong learning. In *Proceedings of the International Conference on Computers in Education, IEEE Computer Society*, 2002.
27. Lucene performance benchmarks, 2005. Available from http://lucene.apache.org.
28. Apache Tomcat, 2005. Available from http://jakarta.apache.org/tomcat/index.html.

29. *RacerPro User Guide Version 1.8*, 2005. Available from http://www.racer-systems.com.
30. Touchgraph, 2005. Available from http://www.touchgraph.com.
31. The Zope Book 2.6 Edition, 2005. Available from http://zope.org/Documentation/Books/ZopeBook/2_6Edition/.
32. OpenCMS 6.0 interactive documentation, 2005. Available from http://www.opencms.org/opencms/en/download/documentation.html.
33. HTML 4.01 specification. W3C Recommendation, 1999. Available from http://www.w3.org/TR/REC-html40/.

Ontology Supported Automatic Generation of High-Quality Semantic Metadata*

Ümit Yoldas[1] and Gábor Nagypál[2]

[1] Conemis AG, Karlsruhe, Germany
yoldas@conemis.com
[2] FZI Research Center for Information Technologies
at the University of Karlsruhe, Karlsruhe, Germany
nagypal@fzi.de

Abstract. Large amounts of data in modern information systems, such as the World Wide Web, require innovative information retrieval techniques to effectively satisfy users' information need. A promising approach is to exploit document semantics in the IR process. For this purpose, high-quality semantic metadata is needed. This paper introduces a method to automatically create semantic metadata by using ontologically enhanced versions of common information extraction methods, such as named entity recognition and coreference resolution. Furthermore, this work also proposes the application of ontology-specific heuristic rules to further improve the quality of generated metadata. The results of our method was evaluated using a small test collection.

1 Introduction

To find relevant documents to a user query, most existing information retrieval (IR) systems merely perform a syntactical comparison between the term-based document representations and the keywords in the query. E.g., if a user initiates a full-text query by typing in the phrase "Semantic Web", a typical IR system on the Web returns a list of hyperlinks pointing to documents that also contain this string syntactically. Although this method may be sufficient for some applications[1], it definitely lacks the linguistic and semantic awareness, which becomes increasingly valuable in large information systems and for complex search requests.

In common IR systems, documents are often represented merely as a set of terms — i.e., strings — together with their corresponding frequency measures. In this model, some crucial aspects of natural languages such as synonyms[2] and homonyms[3] are not considered. E.g., from the system's perspective, the terms

* This work was partially funded by the VICODI (EU-IST-2001-37534), DIP (FP6-507483), and IMAGINATION (FP6-034626) EU IST projects.

[1] For so-called navigational searches, where the exact keywords in the document are already known for the user.

[2] Different words that may denote the same things.

[3] Words with more than one meaning.

R. Meersman, Z. Tari et al. (Eds.): OTM 2006, LNCS 4275, pp. 791–806, 2006.

"doctor" and "physician" differ completely, although in many documents they have the same meaning.

Apart from this trivial deficiency, several other problems lead to decreased IR performance. According to [1], the major reasons why purely text-based search fails to find some of the relevant documents are the following:

- Abstract concepts: Some high-level, vaguely defined abstract concepts like "World War Two" or "Industrial Revolution" are often not mentioned explicitly in relevant documents. Therefore, most text-based search engines do not consider those documents relevant for queries containing these terms.
- Semantic and temporal relations: No connections can be discovered between the terms "Germany" and "Berlin" or the terms "1990s" and "1994" because non-linguistic relations among concepts are not exploited.

Statistical algorithms can detect coexisting terms in texts, which sometimes (but not always) coincides with semantic relations among terms. Thesaurus-based approaches, such as systems using *WordNet* [2], can exploit some basic linguistic relations. Such approaches cannot, however, handle the problem of indirectly relevant abstract concepts or exploit semantic and temporal relations to find relevant documents.

Ontologies provide an "explicit specification of a conceptualization" [3], and make it possible to define knowledge in a machine-processable form using a formal language such as OWL [4]. Using ontologies, it is possible to exploit semantic and temporal relations to improve IR effectiveness.

Semantic metadata link documents with their relevant ontology instances from the knowledge base (KB). High-quality semantic metadata is a major requirement for any ontology-based information system, including the Semantic Web [5]. Because of the large amount of data in modern information systems, manual or semi-automatic approaches for metadata generation, which rely on significant human input during the annotation process [6,7,8,9], are not feasible for most of the applications. It is therefore a very important question how to generate high-quality semantic metadata with as little human effort as possible.

Inspired by the ontology-based IR project VICODI [10], we are currently developing a new ontology-based IR system [1]. This paper describes the metadata generation aspect of this complex system. We present an approach that facilitates the automatic creation of semantic metadata, and provides a framework for the definition of certain ontology-based, domain-specific heuristics. Such heuristics help extend and improve the quality of semantic metadata. To test the claim that such an approach increases the quality of the generated metadata, we evaluated the results of the system using a small test collection.

The structure of the paper is the following. Section 2 gives an overview of our ontology-based metadata generation approach. Section 3 describes our evaluation methodology, and analyzes evaluation results. Section 4 discusses related work, Section 5 concludes the paper and provides some outlook.

2 Approach

2.1 Ontology Formalism

First, some basic requirements are formulated, which have to be met by the ontology formalism in our system. The application domain of our IR system is history and news articles. Therefore, time plays an important role. The ontological structure must provide for temporal restrictions for relation or attribute definitions. These properties are called *temporal relations* and *temporal attributes*, respectively. E.g., it should be possible to define a certain time interval as a validity constraint for the relation `isMemberOf(Steve-Ballmer, Microsoft)`. Moreover, the usual ontological features, such as symmetry and transitivity of relations, and inverse relations, should be supported for temporal relations, too. E.g., we would like to use the temporal transitive `part-of` relation between locations.

As temporal transitive relations are not supported by the current W3C OWL standard [4], an appropriate ontology framework and API is provided by the IR project presented in [1]. Apart from the temporal relations and attributes, our ontology formalism supports the usual ontology modeling constructs, including concepts, instances, relations and attributes[4]. The formalism is implemented using the KAON2 reasoning engine [11], where KAON2 is used as an efficient Datalog engine[5].

2.2 Semantic Metadata Model

Because of performance reasons, we use a metadata model, which is inspired by the common vector space model [12] used in most traditional full-text search engines. This allows us to exploit full-text search engines during the search phase, similarly to [13,14,15]. The classical vector space model represents documents as a weighted set of terms[6]. Therefore, our model is also based on a weighted set of various model elements.

As was mentioned, semantic metadata links documents with ontology elements. Therefore, our metadata model has a *conceptual part*, which consists of a weighted set of ontology instances[7] (OI). In our system, elements of the conceptual part are termed *weighted ontology instances* (WOI). A WOI contains the URI of the ontology instance, together with its weight, denoting its semantic relevance to the document content.

As was also mentioned, time plays a very important role in our application domain. Moreover, from the IR point of view, time has different characteristics

[4] We will sometimes refer to relations and attributes together as "properties", which is the usual Semantic Web terminology.

[5] In addition, KAON2 also supports OWL-DL reasoning and DL-safe rules, but these features are not used in this work.

[6] Also called as the "bag of words" model.

[7] Although the model does not prohibit using other ontology elements, such as concepts, we use the term ontology instances because in our application scenario the conceptual part includes only instances.

from terms or ontology entities. While in the case of terms (and ontology entities) (non-)equality is the only interesting relation, in the case of time, users are mostly interested in other, more complex relations. E.g., if we search for documents about the XX. century we are interested in documents, which are *between* 1901 and 2000. Therefore, our model includes a *temporal part*, which consists of a weighted set of temporal intervals. Actually, for the application domain of history we use fuzzy temporal intervals instead of usual temporal intervals. In this paper, however, we assume for the sake of simplicity that the intervals in the model are all usual time intervals. Details on the fuzzy temporal intervals were reported in [16]. Elements of the temporal part are termed *weighted temporal intervals* (WTI), containing the (fuzzy) temporal interval and its relevance weight.

Finally, it is important to see that for the majority of information systems it is practically impossible to guarantee 100% ontology coverage. I.e., there will be cases, when a relevant concept in the document does not have its respective counterpart in the ontology. For this purpose, we also have a *textual part* in our model, which contains usual strings as its elements. The only difference from the classical full-text vector space model is that our "terms" are not necessary single words. They can be complex phrases, too, such as "Karlsruhe, the German city". Phrases that are semantically important for the document content, but do not have their counterparts in the ontology, are included in this part of the metadata. Elements of the textual part are termed *weighted terms* (WT), containing the term string and its relevancy weight.

The relevance weight values are between 0.0 (no relevance) and 1.0 (maximum relevance) for all metadata parts.

An example of a possible (partial) metadata of a document describing the causes and consequences of the Russian Revolution is shown in Fig. 1.

```
textual: {"Vladimir Ilich Lenin":1.0, "Attack on the Winter Palace":0.7}
conceptual: { #Lenin:1.0, #Russia:0.8, #Russian-Revolution:1.0 }
temporal: {1917-1920:1.0}
```

Fig. 1. Example metadata about the Russian Revolution

2.3 Semantic Annotation Steps

We agree with [14] that common *information extraction* (IE) techniques can substantially support the automatized process of metadata generation. We also accept the statement made there that *named entities*[8] (NE) occurring in text documents constitute a major part of their semantics. Therefore, we start our metadata generation with an IE step, which extracts named entities from the document text. Based on this information, it is possible to generate an initial version of semantic metadata, using the metadata model introduced before. Finally, various ontology-based heuristic rules can be exploited, to extend the

[8] Named entities are terms referring to people, organizations or locations; the definition often includes tokens representing dates, percentage numbers etc.

initial metadata with relevant ontology entities that are not mentioned in the document text explicitly.

The whole metadata generation process is shown in Fig. 2. In the following, we describe the individual steps in more detail.

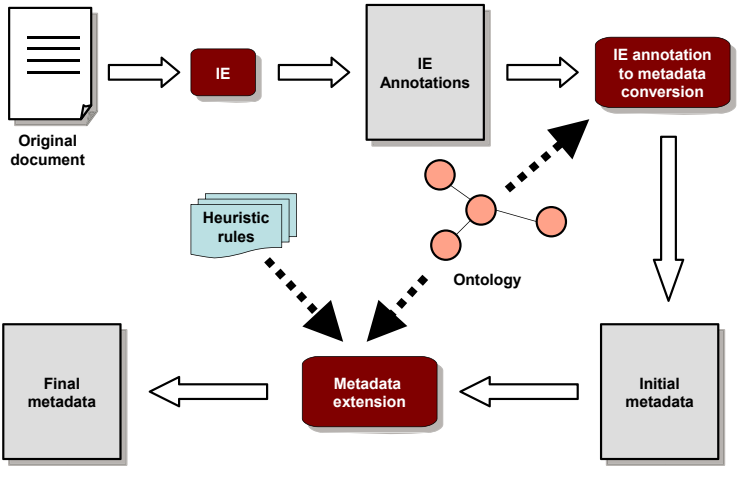


Fig. 2. Steps of the annotation process

2.4 Information Extraction

Classical IE methods comprise some techniques from *natural language processing* (NLP), such as token and sentence splitting or part-of-speech detection. This procedure is sometimes referred to as *shallow parsing* because unlike pure NLP applications, it usually does not include a full (costly) linguistic analysis of the text. The linguistic information obtained by the shallow parsing serves as input to the *named entity recognition* (NER) step.

Detected NEs may also have referring phrases in the text with different notions. These are called *coreferences* and can be distinguished between nominal and pronominal type. An example for a nominal coreference is the term "chief executive officer" referring to the NE "Steve Ballmer" in a document. The "he" reference pointing to "Steve Ballmer" is a pronominal coreference.

To our best knowledge, currently there is no existing IR system which exploits coreference information to create semantic metadata. However, we consider this an important step, since coreference resolution[9] can improve term relevance estimation, as our tests have shown.

With respect to the IE process, we used the established text engineering framework GATE[10], which includes modules for the NLP, NER and coreference resolution tasks. We used the standard ANNIE components, which are included in the standard GATE installation.

[9] Modern implementations may achieve an F-measure of up to 70 percent.

[10] General Architecture for Text Engineering, http://gate.ac.uk/

The result of the described IE operations are GATE annotations following a special annotation scheme. They contain detailed linguistic information about each identified term, such as its position within the sentence, part-of-speech information, and a list of its coreferences. These annotations are automatically stored for each document in a relational database for later use during ontology-supported post-processing.

This separation of expensive IE operations from subsequent ontology dependent tasks has some significant advantages. First, linguistic annotations can be generated independently from ontology-lookup operations and thus are independent from any changes in the ontology[11]. Second, different ontology-based heuristics can be applied and tested without complete regeneration of GATE annotations.

2.5 Initial Semantic Metadata Generation

In our ontology-based approach, the system must be able to identify appropriate NEs as ontology instances[12]. It is a hard task, because a term in the document can syntactically match many ontology instances. To reduce this ambiguity to a minimum, our implementation follows the "longest match principle", which is also used by other approaches [17]. According to this principle, always the longest possible text snippet is matched with the ontology instance labels. I.e., we prefer "Bill Gates Foundation" to "Bill Gates".

Next, linguistic annotations covering an ontology instance are transformed to *ontology annotations* (OIAnnotation). Every OIAnnotation consists of URIs of possibly matching ontology instances (OI), and the number of occurrences of its candidate OIs. The resolved coreferences are taken into account simply by increasing the occurrence counter of the OIAnnotation. E.g., if a pronoun is detected as a coreference to a certain entity, and that entity is known to be an OI, the occurrence counter of the OIAnnotation is increased by one.

The remaining entity annotations are categorized as *term annotations* (TermAnnotation) and *date annotations* (DateAnnotation). Term annotations contain the (normalized) terms from the text, together with their occurrence counters; whereas date annotations are special term annotations, where the term text represents a valid date (or time) specification, such as "May 12, 2006" or "today".

After this step, all annotations are transformed to the initial document metadata, using the model introduced in Section 2.2.

The mapping from linguistic annotations to metadata elements is straightforward. WTIs are generated from date annotations, WTs from term annotations and WOIs from ontology annotations. If an OIAnnotation is ambiguous, i.e., contains more than one possible URIs, WOIs are created for each OI candidate.

[11] For better coreference recognition, it is sometimes necessary to update the gazetteer lists of GATE based on the ontology labels.

[12] Actually we consider all tokens in the text during this step, not only the text snippets that were identified by GATE as NEs. This is needed because GATE sometimes fails to correctly identify text parts as NEs.

WOI and WT weights are calculated according to the following logarithmic function:

$$w(x) = \left(\frac{log(x+1)}{log(r_{max}+1)} \right)^2 \quad (1)$$

where $w(x)$ denotes the resulting weight of a new metadata element; x denotes the occurrence counter of the corresponding annotation element and r_{max} the largest occurrence counter of all annotation elements for the document. Because the co-occurence pattern cannot be applied to date specifications, recognized WTIs always get a weight of 1.0 in the current implementation. A more sophisticated weighting scheme for temporal intervals is subject of future work.

Table 1 illustrates the transformation from annotations to an initial document representation (with $r_{max} = 30$).

Table 1. Initial metadata generation

Entity	*Ann. type*	*Occurence*	*Metadata type*	*Metadata weight*
`#Bill-Gates`	OIAnnotation	30	WOI	1.0
`#Microsoft`	OIAnnotation	15	WOI	0.65
"Oracle Corporation"	TermAnnotation	7	WT	0.37
"Steve Ballmer"	TermAnnotation	2	WT	0.10
`2000-2005`	DateAnotation	N/A	WTI	1.0

2.6 Metadata Extension Using Heuristic Rules

The main idea of our approach is that after the initial generation of the document metadata, certain ontology-specific heuristics are used to adapt this metadata to the document's semantics.

Similar to human readers' cognitive processing, some basic conclusions are drawn automatically by the system. We achieve this by providing a framework for defining and applying specific rules within the document metadata generation process. A special algorithm iteratively applies these heuristic rules and terminates when no further adaptations can be made to the document metadata.

Our rules follow the pattern shown in Fig. 3. To put it simple, the pattern allows domain experts to specify rules, which add new OIs to the conceptual part of the metadata, if they exist in the temporal context[13] of the document, and if they are connected with other instances in the metadata through relations which are valid in the temporal context of the document. If according to the rule an instance should be added to the metadata, which is already there, only the weight of the instance is adjusted.

Concrete examples of rules following this pattern are shown in Fig. 5(a) and 5(b).

[13] The temporal context of the document is defined by the temporal part of the metadata.

Add x to M if all of the following conditions hold

- $C(x)$
- $a^T(x)$
- $\exists x_{1,1} \cdots x_{1,n_1} : n_1 \geq N_1 \wedge x_{1,1} \in M \wedge \cdots \wedge x_{1,n_1} \in M \wedge C_1(x_{1,1}) \wedge \cdots \wedge C_1(x_{1,n_1}) \wedge p_1^T(x_{1,1}, x) \wedge \cdots \wedge p_1^T(x_{1,n_1}, x)$

  $\vdots$
- $\exists x_{k,1} \cdots x_{k,n_k} : n_k \geq N_k \wedge x_{k,1} \in M \wedge \cdots \wedge x_{k,n_k} \in M \wedge C_k(x_{k,1}) \wedge \cdots \wedge C_k(x_{k,n_k}) \wedge p_k^T(x_{k,1}, x) \wedge \cdots \wedge p_k^T(x_{k,n_k}, x)$

where M denotes the current document metadata; $C(x)$ denotes that x is instance of the concept C; $r^T(x, y)$ denotes that the instances x and y are connected with a relation r, whose validity time is in the temporal context of the document; and $a^T(x)$ denotes that the value of the attribute a on instance x is in the temporal context of the document.
The input parameters of the pattern are M, the $C_1 \cdots C_k$ concepts, the $N_1 \cdots N_k$ minimal cardinalities, the C concept, the $r_1 \cdots r_k$ relations, and the a attribute. The specification of a is optional, and $k \geq 1$ must hold.

Fig. 3. Rule pattern

Our weight calculation scheme for the WOIs introduced by the rules is the following:

$$\overline{w} = \left(\sum_{i=1}^{p} \frac{w_i}{p} \right) \cdot \prod_{j=p+1}^{n} \left(1 + \frac{w_j}{2+p} \right) \tag{2}$$

where $\overline{w}$ denotes the resulting weight of the new WOI; $w_1, w_2, ..., w_p, w_{p+1}, ..., w_n$ are the weights of all n WOIs in decreasing order, which are used as input values of the rule with a minimum cardinality of $p = \sum N_i$. This means, at least p WOIs of the current metadata have to meet the rule requirements. The more additional elements are contained in the metadata, the higher the resulting weight of the WOI gets (until a maximum of 1.0).

E.g., if a rule needs at least two metadata elements, which must fulfill the rule conditions, but the current metadata contains three such elements with weights 0.7, 0.6 and 0.4, the resulting weight is calculated as

$$\overline{w} = \left(\frac{0.7 + 0.6}{2} \right) \cdot \left(1 + \frac{0.4}{2+2} \right) = 0.65 \cdot 1.1 = 0.715$$

In our system, rules following the pattern from Fig. 3 can be comfortably defined in an XML file. Thus, the rules can be easily adapted without any programming expertise. Using the XML file, additional parameters can be defined, as well. One parameter determines the minimum weight a WOI must have, so that the rule application algorithm uses it. Another parameter is the *weakening factor*, which determines how the weight of the resulting WOI is weakened[14].

[14] The value of the weakening factor is always less than 1.0.

This is an important feature, because it ensures the termination of the algorithm (weight changes will monotonically converge to zero).

2.7 Metadata Extension Algorithm

Starting with an initial document metadata and some heuristic rules, the metadata extension process is executed. The actual semantic metadata is altered by the following algorithm:

1. Read in the initial document metadata.
2. Read all defined rules from the XML file.
3. Set the current document metadata as the initial metadata.
4. Apply the following steps iteratively
   (a) Execute all applicable rules on the current metadata.
   (b) Extend the current metadata with WOIs added by the rules (or adjust the weights of existing WOIs, if they are already there).
   (c) If the metadata has been modified, restart the iteration.
5. Return the current metadata as the final metadata.

The resulting metadata is the final semantic metadata for our IR system, and stored in the document database for later indexing.

3 Evaluation

We compared the quality of metadata generated by our approach with manually generated semantic metadata. This section discusses the evaluation methodology we used and the results of the evaluation.

3.1 Document Collection

First, we needed to build a small document collection for the purposes of the evaluation. We chose the domain *IT news* and created a small collection of fifteen documents selected from the *ZDNet* news portal[15]. The selected documents are related to one of the three topics *acquisitions*, *product launches* and *IT fairs*.

3.2 Domain Ontology

Using the documents in our document collection, we designed an ontology for the areas *acquisitions*, *product launches* and *IT fairs*. The high-level structure of the resulting ontology is shown in Fig. 4. It is important to see that some of the relations and attributes were defined as temporal (indicated by "T").

Here we will only briefly introduce some concepts and properties, which we consider helpful to comprehend the example heuristics we will provide later. The ontology itself contains 331 instances, 147 relation instances, 91 temporal relation instances, and 24 temporal attribute values.

[15] http://news.zdnet.com/

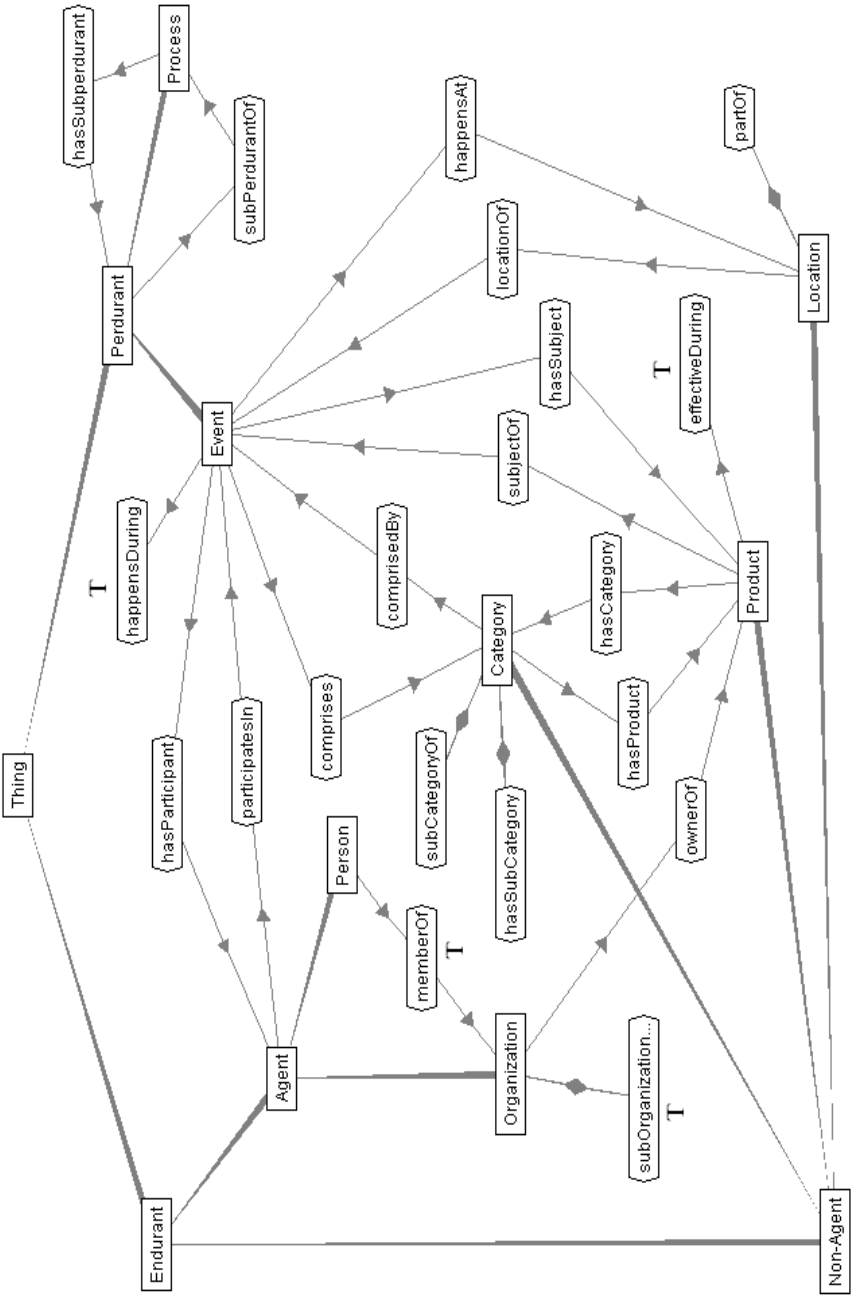


Fig. 4. Evaluation ontology for the IT news domain

Every ontology instance is modeled as direct or indirect instance of the `Thing` concept. An important subconcept of `Perdurant` (a subconcept of `Thing`) is the `Event` class. It is important for the heuristics-based approach because it has many relations to different concepts. Among others, it is related to `Organization` instances (via the relation `hasParticipant`). An example for this relation is:
`hasParticipant(Microsoft-launches-BizTalkServer-2004, Microsoft)`.

The temporal attribute `happensDuring` defining the temporal extension of an event, can be exemplified by:
`happensDuring(PeopleSoft-replaces-CEO, '2004-10-01;2004-10-31')`.

The time interval given by the start date `2004-10-01` and the end date `2004-10-31` marks the interval when the event happened.

3.3 Heuristic Rules

Generally, heuristic rules must be carefully aligned with the domain ontology. Otherwise, they may unintentionally disturb the document's semantics, or may not find the expected relevant concepts.

For our evaluation, we defined fourteen heuristic rules, of which we now introduce two for demonstration purposes. These rules are formulated and briefly explained in Fig. 5(a) and 5(b).

Add x to M if all of the following conditions hold

- $Event(x)$
- $happensDuring^T(x)$
- $\exists x_1 \cdots x_n : n \geq 2 \wedge x_1 \in M \wedge \cdots \wedge x_n \in M \wedge Agent(x_1) \wedge \cdots \wedge Agent(x_n) \wedge participatesIn(x_1, x) \wedge \cdots \wedge participatesIn(x_n, x)$

Idea: If at least two agents – i.e., instances of the concepts `Person` or `Organization` – are contained in the current semantic metadata, these agents are related via `participatesIn` to the same event instance, and the temporal context of the document is compatible with the time interval given by the temporal attribute `happensDuring` then the targeted ontology instance of `Event` is considered relevant to document.

(a) Event rule

Add x to M if all of the following conditions hold

- $Process(x)$
- $\exists x_1 \cdots x_n : n \geq 1 \wedge x_1 \in M \wedge \cdots \wedge x_n \in M \wedge Event(x_1) \wedge \cdots \wedge Event(x_n) \wedge subPerdurantOf(x_1, x) \wedge \cdots \wedge subPerdurantOf(x_n, x)$

Idea: At least one ontology instance of the concept `Event` in the metadata leads to the addition of all related process instances (via relation `subPerdurantOf`).

(b) Process rule

Fig. 5. Example rules

E.g., if a the ontology contains information on the `Peoplesoft-replaces-CEO` event, and the initial semantic metadata contains `Peoplesoft`, `Craig-Conway`

and `Dave-Duffield`, `Peoplesoft-replaces-CEO` is added to the metadata according to the *Event rule* (Fig. 5(a)).

Using this extended metadata, the *Process rule* (Fig. 5(b)) adds the information about the high-level concept *Process* `Oracle's-takeover-of-Peoplesoft`.

As this example illustrates, heuristic rules can often build on the result of previous rules. This way, semantically related OIs are added stepwise to the initial document representation.

3.4 Evaluation Methodology

The evaluation consisted of three steps. First, domain experts were asked to manually define reference document representations for each document in our evaluation corpus. I.e., they were told to select ontology instances from the knowledge base, and assign them to one of five equidistant intervals between 0.0 and 1.0.

Next, we executed two runs of the system. One run without using any of our heuristics (i.e., only generating the initial metadata representation), and another run with all heuristics activated.

Finally, we compared the generated semantic metadata with the reference metadata defined by the experts, using the following evaluation measure. If the calculated weight of a metadata element lies within the reference interval of that element, a conformity value of 1.0 is assigned to this element. If it is within one interval above or below, a conformity value of 0.5 is assigned. If a calculated element is not contained in the reference metadata, or its weight is not one interval above or below reference weight, the assigned conformity value is 0.0. Finally, for elements that appear only in the reference metadata, but do not appear in the generated metadata, a conformity value of 0.0 is assigned.

By averaging the conformity values for the metadata elements, we obtained average conformity values for each document, which indicate how similar the generated metadata are to the manual (perfect) metadata.

3.5 Results

Fig. 6 compares the measured conformity values without heuristics to those where the rules were applied.

As we can see, almost every document representation was improved by applying the heuristic rules. The average conformity value of the initial document metadata was 0.67, whereas applying the rules lead to an average of 0.81. In none of the cases a decrease in annotation quality was observed. Moreover, no improvement of the semantic metadata could be observed at only two documents (number 3 and 15). In these cases no rules could be applied on the initial metadata. This can be either due to insufficient ontological knowledge[16], or because the documents would need heuristics which are not described by the existing rule set.

[16] WOIs that would be preconditions in rules are not included in the initial metadata.

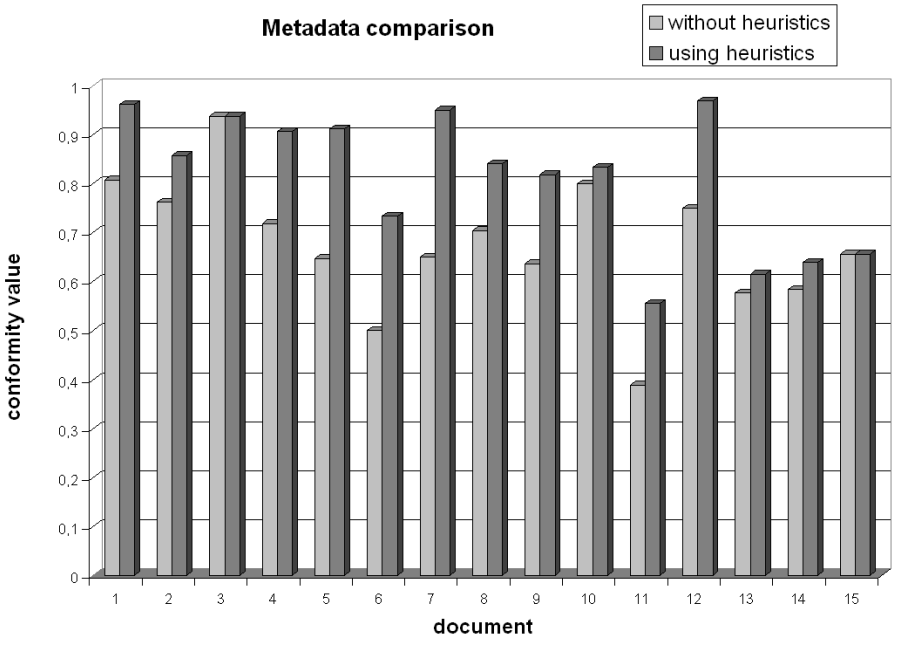


Fig. 6. Evaluation results

4 Related Work

Generating high-quality semantic metadata with the least possible human effort is agreed to be an important step toward the Semantic Web. Thus, there are several projects that aim to support or replace human experts in the metadata generation task.

Approaches such as [6,8,18,9] propose frameworks for manual annotation of metadata. E.g., a GUI[17] application which facilitates the annotation of semantic tags is provided by [8]. However, fully manual approaches do not scale well for large information systems.

There are a couple of automatic annotation systems, as well. These include the KIM system [14], the SemTag system [19], and the system of Vallet et al [17]. Unfortunately, they do not exploit the full ontology structure, but use only labels of ontology elements (the KIM system) or exploit only the concept taxonomy in addition to ontology labels (Vallet et al. and SemTag).

S-CREAM [7] uses a semi-automatic annotation approach, which includes a machine learning algorithm. They use the ontology to refine the structure of the semantic metadata, i.e., to find the exact ontological relations between metadata elements. Our approach is different because we concentrate on finding new, only implicitly relevant ontology instances — a task what S-CREAM does not address.

[17] Graphical User Interface.

The authors of C-PANKOW [20] propose an advanced annotation and disambiguation system without any machine learning technique. Their approach is to identify correct conceptual entities by measuring statistical information from Google search results. The system uses, however, only syntactical information, i.e., cannot find indirectly mentioned instances.

5 Conclusion and Outlook

In this work, we presented a system for automatic ontology-supported generation of semantic metadata, as part of our ontology-based IR system. We showed how common IE methods can be used in combination with a simple lookup on ontology labels to create an initial document representation. On top of that, we propose our framework for heuristic rule definition for semantically extending document metadata.

Our evaluation results verified the thesis that suitably parameterized heuristic rules can indeed significantly improve the quality of semantic metadata. With our selected evaluation corpus, the average conformity value could be increased by 20.9 percent.

As the manual definition and parameterization of adequate heuristic rules for larger ontologies is a quite laborious task, the integration of some automatic techniques may be reasonable. A promising step in this direction is done by [21]. We will consider to integrate the proposed spread-activation approach to our heuristics-based system.

Currently, we are working on a more elaborated temporal information extraction module. Another scheduled improvement is to include a disambiguation step into the metadata generation process. In the current system only new ontology elements can be added, or weights can be adjusted. There is no way, however, to remove apparently irrelevant ontology entities from the semantic metadata — a deficiency, which we would like to address.

References

1. Nagypál, G.: Improving information retrieval effectiveness by using domain knowledge stored in ontologies. In: On the Move to Meaningful Internet Systems 2005: OTM 2005 Workshops. Volume 3762 of Lecture Notes in Computer Science. (2005) 780–789
2. Voorhees, E.M.: Using WordNet to disambiguate word sense for text retrieval. In: Proceedings of SIGIR-93, 16th ACM International Conference on Research and Development in Information Retrieval, Pittsburgh, US (1993) 171–180
3. Gruber, T.: A translation approach to portable ontology specifications. Knowledge Acquisition **5** (1993) 199–220 the definition of the word "ontology".
4. Dean, M., Schreiber, G.: OWL web ontology language reference. Recommendation, W3C (2004)

5. Berners-Lee, T., Hendler, J., Lassila, O.: The Semantic Web. Scientific American **284** (2001) 34–43
6. Decker, S., Erdmann, M., Fensel, D., Studer, R.: Ontobroker: Ontology based access to distributed and semi-structured information. In Meersman, R., Tari, Z., Stevens, S.M., eds.: Database Semantics - Semantic Issues in Multimedia Systems, IFIP TC2/WG2.6 Eighth Working Conference on Database Semantics (DS-8). Volume 138 of IFIP Conference Proceedings., Kluwer (1999) 351–369
7. Handschuh, S., Staab, S., Ciravegna, F.: S-CREAM - semi-automatic CREAtion of metadata. In Gómez-Pérez, A., Benjamins, V.R., eds.: Knowledge Engineering and Knowledge Management. Ontologies and the Semantic Web, 13th International Conference, EKAW 2002. Volume 2473., Springer (2002) 358–372
8. Hendler, J., Heflin, J.: Searching the web with SHOE. In: Artificial Intelligence for Web Search. Papers from the AAAI Workshop., AAAI Press (2000) 35–40
9. Martin, P., Eklund, P.: Embedding knowledge in web documents. In: Proceedings of the Eighth International World Wide Web Conference, Toronto, Canada, Elsevier (1999) 325–341
10. Nagypál, G., Deswarte, R., Oosthoek, J.: Applying the Semantic Web – the VICODI experience in creating visual contextualization for history. Literary and Linguistic Computing **20** (2005) 327–349
11. Hustadt, U., Motik, B., Sattler, U.: Reducing SHIQ-description logic to disjunctive datalog programs. In: Principles of Knowledge Representation and Reasoning: Proceedings of the Ninth International Conference (KR2004). (2004) 152–162
12. Salton, G., Wong, A., Yang, C.S.: A vector space model for automatic indexing. Communications of the ACM **18** (1975) 613–620
13. Finin, T., Mayfield, J., Joshi, A., Cost, R.S., Fink, C.: Information retrieval and the Semantic Web. In: Proceedings of the 38th Annual Hawaii International Conference on System Sciences (HICSS'05). (2005)
14. Kiryakov, A., Popov, B., Terziev, I., Manov, D., Ognyanoff, D.: Semantic annotation, indexing, and retrieval. Journal of Web Semantics **2** (2005) 49–79
15. Davies, J., Weeks, R.: QuizRDF: Search technology for the Semantic Web. In: Proceedings of the 37th Hawaii International Conference on System Sciences (HICSS-37 2004),. (2004)
16. Nagypál, G., Motik, B.: A fuzzy model for representing uncertain, subjective, and vague temporal knowledge in ontologies. In Meersman, R., Tari, Z., Schmidt, D.C., eds.: On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE. Volume 2888 / 2003 of Lecture Notes in Computer Science., Springer-Verlag (2003) 906 – 923
17. Vallet, D., Fernández, M., Castells, P.: An ontology-based information retrieval model. In: The Semantic Web: Research and Applications: Second European Semantic Web Conference, ESWC 2005. Volume 3532 of Lecture Notes in Computer Science., Heraklion, Crete, Greece, Springer (2005) 455–470
18. Kahan, J., Koivunen, M.R., Prud'Hommeaux, E., Swick, R.R.: Annotea: An open RDF infrastructure for shared web annotations. Computer Networks **39** (2002) 589–608
19. Dill, S., Eiron, N., Gibson, D., Gruhl, D., Guha, R., Jhingran, A., Kanungo, T., Rajagopalan, S., Tomkins, A., Tomlin, J.A., Zien, J.Y.: SemTag and Seeker: Bootstrapping the semantic web via automated semantic annotation. In: Proceedings of the Twelfth International World Wide Web Conference, WWW 2003, Budapest, Hungary (2003) 178–186

20. Cimiano, P., Ladwig, G., Staab, S.: Gimme' the context: context-driven automatic semantic annotation with C-PANKOW. In Ellis, A., Hagino, T., eds.: Proceedings of the 14th international conference on World Wide Web, WWW 2005, Chiba, Japan, ACM (2005) 332–341
21. Rocha, C., Schwabe, D., Aragao, M.P.: A hybrid approach for searching in the semantic web. In: Proceedings of the 13th international conference on World Wide Web (WWW '04), New York, NY, USA, ACM Press (2004) 374–383

Brokering Multisource Data with Quality Constraints

Danilo Ardagna, Cinzia Cappiello, Chiara Francalanci, and Annalisa Groppi

Politecnico di Milano, Department of Electronics and Information
{ardagna, cappiell, francala, groppi}@elet.polimi.it

Abstract. Access to multisource heterogeneous data is a fundamental research issue in a variety of contexts, including syndicated data retrieval, Web service selection and cooperative information systems. In these variable contexts, the brokering approach to multisource data access provides greater flexibility with respect to the more traditional data integration. The general brokering model assumes that the broker is submitted a query and has the responsibility to optimize the response along specified parameters such as time efficiency, completeness, and consistency. This paper takes a data quality perspective on data brokering and considers data accuracy. Furthermore, the data quality literature assumes that metadata are associated with data to describe their quality. Metadata support data selection without viewing and assessing data directly. On the contrary, previous brokering approaches view data. This paper compares previous results with those of a brokering approach based on metadata which assumes that actual data are transparent to the broker. Testing results comparing the delta between the data visibility and transparency approaches to data brokering are presented.

1 Introduction

Access to multisource data is a fundamental research issue in a variety of contexts, including syndicated data retrieval, Web service selection and cooperative information systems. These contexts are characterized by high heterogeneity and variability. The number of data sources can vary over time. Furthermore, data sources can overlap with each other and, in turn, overlapping data can cause integration conflicts. These conflicts are usually solved at a schema level, according to the Local As View (LAV) or Global As View (GAV) approaches [6]. However, they may persist at a data level. For example, syndicated data providers typically offer overlapping data sets that are updated at different points in time. In cooperative information systems, schemas themselves may be only partially integrated and typically undergo rapid change over time [12]. In these highly variable contexts, the brokering approach to multisource data access provides greater flexibility with respect to the more traditional data integration.

The general brokering model assumes that the broker is submitted a query and has the responsibility to optimize the answer by providing the knowledge necessary to make a choice among multiple query answers [12] [2]. In the literature, optimization parameters are typically time efficiency in query optimization

R. Meersman, Z. Tari et al. (Eds.): OTM 2006, LNCS 4275, pp. 807–817, 2006.

[3], completeness in the distributed database domain [9] and consistency in data integration [13]. This paper takes a data quality perspective on data brokering and considers as an important dimension, data accuracy. Available data values are considered accurate if they coincide with their actual values. The accuracy dimension of data quality has rarely been considered as a brokering dimension. The correspondence between available and actual data values is clearly difficult to measure and accuracy is usually provided a statistical estimate based on sampling [11]. These estimates represent an instance of metadata, which represent a description of data properties.

The data quality literature hypothesizes that metadata are associated with data along all quality dimensions. The choice among alternative answers to a user query can be based on metadata, which allow selecting the answer that maximizes overall data quality. This allows the broker to select an answer without viewing and assessing data directly. On the contrary, the majority of previous brokering approaches are based on the assumption that data are visible [12, 2]. This assumption raises a number of limitations to the applicability of brokering in a business context, where data are often strictly private.

The paper presents testing results comparing the delta between the data transparency and visibility approaches (DTA and DVA respectively) to data brokering. In the first case, the answer returned to the user is selected by using metadata. In the second case, the broker selects an answer by using data and calculating the actual improvement that can be obtained for each data source. The purpose of the paper is to verify the extent to which the greater computational complexity of the DVA is counterbalanced by a higher quality of the answer and to study the properties of data sources affecting this trade-off. In particular, the degree of overlap among data sources and the relevance of data quality dimensions are properties that are considered in the analyses.

The presentation is organized as follows. Section 2 describes the broker architecture and the data model in the DTA and DVA. Considering the DVA, the model for the evaluation of accuracy after merging and data cleaning operations is also presented. Section 3 discusses the brokering methodology. Section 4 presents the experimental results in which the two approaches are compared. Section 5 discusses the innovative aspects of the paper compared to previous work presented in the literature. Conclusions are drawn in Section 6.

2 A Quality-Oriented Architecture to Query Multi-source Data

The broker is supposed to receive a query from a customer specifying both the data request and the quality requirements that must be satisfied by the answer. Considering the providers' data available in the system, it calculates the set of sources that better satisfies user requirements. The broker is modelled according to the Local-As-View (LAV) perspective [6], representing the schema of a source as a view of a global schema (GS). The broker knows both the GS and the local views of all sources. The GS is defined as a set of relations, called Global

Relations, which are tied to each other by join attributes. The Universal Relation (UR) is defined as the join of all global relations. Assuming that A is the set of attributes of the UR, a user query Q is a conjunction of selection and projection operations on attributes $a_k \in A_Q$, and of quality requirements. Selection operations and quality constraints are expressed in the form "a_i [or QD_r] θ constant", where $\theta \in \{<, >, \leq, \geq, =\}$ and QD_r is a generic quality dimensions defined in the range [0, 1]. Users specify the minimum level of quality that they consider acceptable, referred to as QD_r^*. Users can also rank the importance of quality dimensions by specifying weights w_r, with $1 \leq r \leq R$, such that $\sum_{r=1}^{R} w_r = 1$. The overall quality q required by the customer is expressed as $\sum_{r=1}^{R} w_r \cdot QD_r$. A constraint q* on the overall value of quality q is also specified by the user.

The data quality literature has defined a set of quality dimensions. This paper focuses on accuracy, since it has rarely been considered as a brokering dimension. The difference between DTA and DVA mainly impacts on the accuracy dimension. In DVA, all the cleaning operations that can improve such a dimension are evaluated and mathematically modelled as described in Section 2.2.

2.1 The DTA: The Broker Cannot View Data Values

The main assumption of the DTA is that the broker cannot view data values, but it can view metadata and knows the quality of local data sets. It also knows the extent to which local data sets overlap with each other. In details, local data sets are divided into fragments and the broker knows the cardinality of all data fragments (i.e. the number of data values contained in the fragments). The response Q* to query Q is a set of fragments. The local data sets that can contribute to build response Q* are referred to as d_i. Each data set d_i represents the smallest subset of data with which a provider can contribute to satisfying Q. We assume that data quality is homogeneous within each data set ***(HP1)***. A data set d_i is defined as a set of fragments.

In general, the response to query Q, can be built by using multiple combinations of d_i, i.e query plans. The broker identifies all possible query plans that can be used to build Q*. Note that data sets can overlap with each other and the broker also knows the cardinality of overlap among data sets. The average value of quality of each d_i is indicated as $QD_r(d_i)$. We assume that providers are responsible for the evaluation of the quality of their data. To support the selection of the most suitable set of suppliers, the broker must calculate the overall quality of each query plan. Mathematical details of DTA are omitted in this paper. The entire model is thoroughly described in [1].

2.2 The DVA: The Broker Can View Data Values

According to the DVA, the broker can view the data values supplied by providers. Therefore, it is possible to hypothesize that quality values are associated with values of attributes included in the data sets as opposed to fragments. We suppose that quality values have a Gaussian distribution over data values. With this additional information, the broker can perform data cleaning activities to

improve the quality of local data and perform quality-based merging of local data. These improvement actions are discussed in the next sections.

Data cleaning activities. The data cleaning theory provides the following classification of data cleaning actions [4]: (a) normalization of data format and/or representation; (b) resolution of acronyms and abbreviations; (c) elimination of duplicated records; (d) reconciliation of contradicting records; (e) control of external references; (f) extraction of embedded values through data parsing.

In our approach, the broker does not control external references, since we assume that providers guarantee the referential integrity of their sources. Furthermore, the extraction of embedded values is not required, since a global schema is supposed to exist according to the LAV approach. Thus, the data cleaning actions performed by the broker are the normalization of data format, the resolution of acronyms and abbreviations, and the elimination of duplicated and contradicting records. The normalization of data format is a data cleaning action suitable for particular attributes such as date, currency, and measurement unit. Resolution of acronyms and abbreviations requires a dictionary providing the full words corresponding to different acronyms and abbreviations. Finally, for the elimination of duplicated and contradicting records, the literature provides several techniques to identify records referring to the same object [5]. In general, similar records are more likely to refer to the same object and to be either duplicated or contain attributes with contradicting values. If two such records are identified, a new tuple t* is created replacing the original records and storing correct values. Correct values are identified as follows. Null values are considered first. If a field contains a null value in a record and a non-null value in the other record, the new tuple t* will include the non-null value. If both records have an admissible value for a specific field, the "distance" between the two values is calculated to understand whether records should be considered different. This degree of similarity is calculated by using the EditDistance technique. The algorithm that solves this problem can be found in [8]. If similarity is greater than a certain value Sim_{min} the two values are considered equal (duplicated records) and the value with greater accuracy is selected. If the two values are not similar, but the difference between their values of accuracy is greater than a specified value Δ_{min}, the value with the higher value of accuracy is considered correct. Otherwise, business rules are applied, if possible. Business rules have commonly the following schema: "if *Condition* then *Action*" and verify if existing constraints associated with each attribute are violated.

Improvement of accuracy. In this section, the improvement of accuracy for each data cleaning technique considered is modeled and calculated.

1 - Normalization of data format - The normalization of data format impacts the data representation. For example, the same data measured with different units may appear as different data, while storing the same content with a different representation. It is necessary to transform data in the right format by maintaining the same content. Note that if the content is not accurate, the format transformation does not improve data quality. It is possible to state that the accuracy of data attribute a_k in a tuple t_i is calculated as a weighted average

of two components: content accuracy and format accuracy. Note that weights have to be defined by considering that content is more relevant than format. If the format of some attribute needs to be changed, errors can be modelled by the Poisson distribution. If N is the number of tuples to be checked, p is the probability that each data have a wrong format, and $\lambda = Np$ the parameter of the Poisson distribution. Since the average number of format errors in the database is λ, it is possible to estimate the accuracy increase related to a single attribute as: $\Delta_{Acc}(t_i[a_k]) = [(\lambda \cdot \alpha)/n] \cdot Acc(t_i[a_k]$. The average accuracy improvement $\Delta_{Acc}([a_k])$ of an attribute a_k on all N tuples of the data set can be expressed as $\Delta_{Acc}([a_k]) = \sum_{i=1}^{N} \Delta_{Acc}(t_i[a_k])/N$.

2 - Resolution of acronyms and abbreviations - Abbreviations and acronyms are considered a format mismatch and are modelled as described in the previous section.

3 - Deletion of duplicated and contradictory records - When two records refer to the same object, it is necessary to choose which value has to be considered in the final solution. It is possible to identify different situations:
− both values are Null: null value is inserted in the final solution and the accuracy improvement is equal to 0.
− one of the two values is null: the not null value is considered for the final solution. If $\mathrm{Acc}(\mathrm{t}_i[\mathrm{a}_k])$ and $\mathrm{Acc}(\mathrm{t}_j[\mathrm{a}_k])$ are the accuracy values associated with the data values, the accuracy improvement is: $\Delta_{Acc}(t^*[a_k]) = |Acc(t_i[a_k]) - Acc(t_j[a_k])|/2$ and since, in this case, one data value is equal to Null $\Delta_{Acc}(t^*[a_k]) \in (0, 0.5]$.
− both values are not null and are similar: the value associated with the greater value of accuracy is considered. The accuracy improvement is defined in the interval $\Delta_{Acc}(t^*[a_k]) \in (0, 0.5]$.
− both values are not null, are not similar and the difference of their accuracy values is greater than a specified Δ_{min}: the data value associated with highest accuracy is chosen. The accuracy improvement is defined in the interval $\Delta_{Acc}(t^*[a_k]) \in [\Delta_{min}/2, 0.5]$.
− both values are not null, the values are not similar and the difference of their accuracy values is lower than a specified Δ_{min}: in this case, it is not possible to choose a value by considering the accuracy value. It is necessary to apply business rules related to the specific attribute. As explained above, a business rule verifies the data value along specified conditions: if the rule is not satisfied, accuracy is equal to 0, otherwise accuracy is equal to 1. Then, the data value characterized by the higher value of accuracy is chosen. Three cases can be verified: first, no business rule is associated with the attribute. In this case, the result is the value associated with the greater value of accuracy. The accuracy improvement is defined in the interval $\in (0, \Delta_{min}/2]$. In the second case, business rules exist. If they are not verified for both values, the final accuracy is equal to 0. Otherwise, if they are verified for both values the solution includes the data value associated with the greater value of accuracy and the evaluation of the accuracy improvement is similar to the previous case. In the third case,

business rules are verified for one value. The accuracy improvement is defined in the interval $\in [\Delta_{min}/2, 1 - \Delta_{min}/2]$.

The seven cases above are considered as events e_h with probability p_h. $E[e_h]$ is the expected value of the accuracy improvement related to the h-th event. The value of the accuracy improvement associated with generic attribute k of the tuple t_i that is: $\Delta_{Acc}(t^*[a_k]) = \sum_h p_h \cdot E[e_h]$ where $0 \leq p_h \leq 1$ and $\sum p_h$=1.

The average accuracy improvement associated with the record is: $\Delta_{Acc}(t^*) = \sum_{k=1}^{K} \Delta_{Acc}(t^*[a_k])/K$ where K is the number of attributes in the database schema.

Finally, it is necessary to estimate the probability of duplicated or contradictory data. We associate a value β to this probability and estimate the total accuracy improvement as: $\Delta_{AccTot}(t^*) = \beta \cdot \Delta_{Acc}(t^*)$.

Merging different sources. The broker selects the data sets that build the most complete and accurate answer. Data sets are provided by different sources and can overlap. We assume that each source has a unique ID and that two tuples refer to the same object if they have the same ID. In order to solve conflicts among data values it is necessary to define a resolution function. Let us consider the domain D of a given attribute, $D^+ = D \bigcup$ Null, and [0,1] is the domain of the accuracy value associated with the attribute. A resolution function is an associative function $f:(D^+ \times [0,1]) \times (D^+ \times [0,1]) \rightarrow D^+$ that works on the same logic presented for the deletion of duplicated and contradictory records.

When data sources overlap, the standard relational operators are not suitable for integration. The literature provides the following merge operators:

- *Join-Merge Operator j-m(t[a_m])*: can be applied in two situations: (a) the join attribute $t_i[a_m]$ is the ID for both sources; (b)the join attribute $t_i[a_m]$ is the ID for one source, while it represents a foreign key in the other source.
- *Left (Right) Outerjoin-Merge-Operator (l-m(t[a_m]))*: guarantees that all the tuples of the data set at the left (right) of the operator belong to the final solution and the join with the other attributes of the data set is performed wherever possible, otherwise corresponding data values are set to Null.
- *Full Outerjoin-Merge Operator (f-m(t[a_m]))*: guarantees that all the tuples of both data sets are included in the final solution.

In [9], these operators are defined for the completeness dimension. We have defined these operators for the accuracy dimension and the total value of accuracy calculated using these operators can be represented as the ratio between the sum of the accuracy value associated with each data values and the total number of attributes is:

$$\sum_{i=1}^{I} \sum_{k=1}^{K} Acc(t_i^*[a_k])/(I \cdot K)$$

where I is the number of tuples that are included in the final result, K is the number of attributes included in the final result and $Acc(t_i^*[a_k])$ is the accuracy of the generic data value $t_i^*[a_k]$ belonging to the final solution.

3 The Data Quality Brokering Methodology

In this section we summarize the data quality brokering methodology we presented in [1]. A query plan is considered *feasible* if it satisfies both quality and a price constraint $Price^*$. A query plan is optimum if it is feasible and maximizes quality. The goal of the broker is to select the optimum query plan $\overrightarrow{x}^*$. If no feasible plan exists, the broker can negotiate data quality characteristics with data providers which can improve the quality of their data with an additional cost. Negotiation identifies a new set of candidate data sets whose combination may provide a solution satisfying constraints (see Figure 1). In [1] we have modeled the identification of the optimum query plan as a NP-hard mixed-integer non linear problem which has been solved by the tabu search algorithm. The negotiation process is based on multi-party, multi-attribute, single-encounter negotiation.

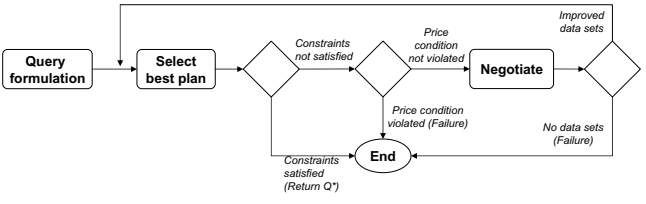


Fig. 1. Data quality brokering methodology

In this paper, in order to evaluate the enhancement of the data accuracy quality dimension which can be obtained by implementing the data cleaning operations discussed in Section 2.2, we relax the price constraint and we assume that a feasible solution of the problem always exists. In the following the negotiation approach will not be investigated and we will limit our analyses only to the data accuracy and data completeness quality dimensions.

4 Experimental Results

The effectiveness of our approach has been tested on a wide set of queries. A distributed data citizen information database adopted by the Italian Public Administration has been considered as UR. Data fields accuracy values have been randomly generated according to a Gaussian distribution for the probability density function. Analyses have been performed by varying as in [1] the following parameters: (i) the data field Null probability, (ii) the expected value of the Gaussian probability density function, (iii) its corresponding variance. Queries with up to 16 data sets and 23 attributes have been considered. The comparison of DVA and DTA has been performed by considering the same number of iterations and the maximum number of iterations for the tabu search algorithm has been set equal to 100. On average, the DVA implies a 30% execution time overhead. In all analyses, for a given value of parameters several tests have been performed. In the following, results of representative test cases are reported.

4.1 Data Accuracy Heterogeneity vs. Data Cleaning Analysis

The aim of this test is to estimate how the improvement of accuracy changes with the heterogeneity of the local data sets. Analyses consider four test cases:

- case 1: 16 data sets have an expected value of accuracy equal to 0.7;
- case 2: 8 data sets have an expected value of accuracy equal to 0.7, and 8 data sets have an expected value of accuracy equal to 0.8;
- case 3: 4 data sets have an expected value of accuracy equal to 0.7, 6 data sets have an expected value of accuracy equal to 0.8, and 6 data sets have an expected value of accuracy equal to 0.9;
- case 4: 4 data sets have an expected value of accuracy equal to 0.7, 4 data sets have an expected value of accuracy equal to 0.8, 4 data sets have an expected value of accuracy equal to 0.9 and 4 data sets have an expected value of accuracy equal to 0.99.

The accuracy of the initial data sets increases from case 1 to case 4. The Null data probability value and the variance of the Gaussian density function have been set equal to 0.01 and 0.02, respectively. Completeness greater than 0.8 and accuracy greater than 0.65 are required. The query includes all attributes. Results are reported in Figure 2. Note that, the accuracy of the final query plan increases both for the DTA and DVA as the accuracy of the data fragments is increased. The percent improvement decreases and the effectiveness of the broker tends to decrease as the accuracy of the data sets increases. This is due to two factors: (i) if the accuracy of data sets increases, the improvement from data cleaning operations become less effective since there is a lower number of errors to correct; (ii) if the number of data sets with a high value of accuracy increases, the merging procedure becomes less efficient since overlapping data often coincide with the data characterized by the highest average accuracy: the DVA and DTA select tuples from the same data set.

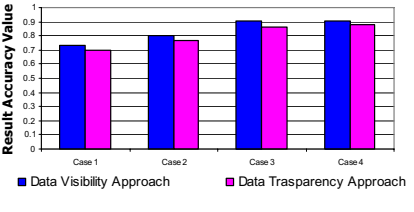


Fig. 2. Acc. heter. and Data Cleaning

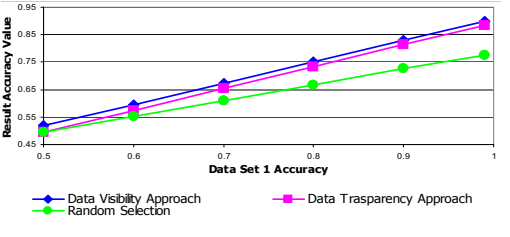


Fig. 3. Acc. heter. and Data Merge

4.2 Data Accuracy Heterogeneity and Merging

The goal of the analysis is to determine how the quality heterogeneity of the local data sets influences the effectiveness of merging. Intuitively, the greater the difference of accuracy for overlapping data, the greater the improvement obtained from merging. Two data sets are considered. Initially they have the same value of accuracy, equal to 0.5; then, the accuracy of the first data set is

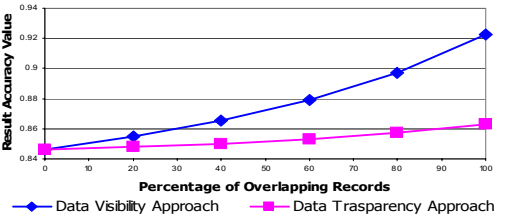


Fig. 4. Accuracy trend as a function of the percentage of overlapping records

increased by 0.1 at each step, until accuracy is equal to 1. The data field Null probability and the variance of the Gaussian density function have been set equal to 0.01 and 0.02, respectively. The query includes the overlapping data fields that are not involved in data cleaning operations in order to evaluate the enhancement of data quality due to the merging algorithm only. The completeness constraint is equal to 0.8, while the accuracy constraint is set equal to 0.45. We consider also the random selection of tuples from the two data sets. Results are shown in Figure 3. Note that the accuracy of the query plan of the DVA is always greater than that determined by the other two solutions. The improvement with respect to the random selection increases, since if one of the two data sets becomes more accurate, the selection of the best data performed by the merging algorithm becomes more effective. On the other hand, the improvement with respect to the DTA decreases, since if one of the two fragments becomes more accurate, the DVA and DTA select tuples from the same data set.

4.3 Fraction of Records and Merging

The aim of this analysis is to evaluate the dependency of the accuracy on the fraction of overlapping records. Two data sets are considered. The query includes the overlapping data fields that are not involved in data cleaning operations in order to evaluate the enhancement of data quality due to the merging algorithm only. Accuracy and completeness constraints have been set greater than 0.8. The data fields Null probability and the variance of the Gaussian density function have been set equal to 0.01 and 0.02 respectively, while the expected value has been randomly generated in the range [0.8, 0.9]. The fraction of overlapping records has been varied in the range [0, 100%] with step 20%. The results of a representative test case are reported in Figure 4. As the percentage of overlapping records increases, the accuracy of the final query plan grows for both the DTA and DVA. The improvement is more significant for the DVA since the merging algorithm always selects the best of the overlapping data. Note that the accuracy of the DTA is always greater than that of the DTA. The percent improvement of accuracy obtained by the DVA with respect to the DTA increases up to 7%.

5 Related Work

The problem of accessing multisource heterogeneous data has attracted vast attention by the research community. If the same data can be obtained by multiple providers, the data quality becomes a selection driver. Several dimensions have been proposed in the data quality literature. For example, accuracy and completeness assess data along their numerical extension and correctness. Timeliness evaluates the validity of data along time. Several architectures for the data quality management and data provider selection based on quality criteria have also been proposed in the literature. In the particular context of Cooperative Information Systems (CIS), a Data Quality Broker has been proposed for the selection of the best data sources satisfying quality requirements [12]. The broker receives a user request and sends corresponding data requests to the organizations belonging to the CIS. The broker is based on a GAV (Global As View) approach, since it is responsible for data retrieval and reconciliation. Reconciliation is performed by choosing the data values characterized by highest quality. Anyway, the paper does not provide a mathematical model for the calculation of overall quality. The optimization of the query plan has been addressed in [2,7,10]. Authors in [2] have considered a linear formulation of the optimization problem which is obtained by considering a priori all possible intersections of overlapping data sets and by pre-computing corresponding quality values. The problem is solved by state of the art integer linear solvers and the optimum solution is identified. Authors in [7] and [10] proposed a similar formulation of our optimization problem which is solved by a branch and bound algorithm. The branch and bound can identify the optimum solution of the problem, but the worst case execution time grows exponentially with the number of nodes of the underlying decision tree [14], which is obtained when no feasible solution exists.

6 Conclusions and Future Work

In this paper we have presented a comparison between the DVA and DTA to data brokering. Results show that the accuracy improvement obtained by using the DVA justifies the additional computational complexity. In particular, the accuracy improvement is more significant when data sets have low accuracy and data cleaning techniques are adopted. Future work will extend the optimization algorithm and implement column generation techniques in order to identify the global optimum. Negotiation phase will be also considered with the introduction of the evaluation of price constraints in the optimization problem. Finally, the methodology will be extended by considering other data quality dimensions.

Acknowledgements

This work has been partially supported by the MAIS FIRB Italian Project. Thanks are expressed to Marco Comuzzi and Barbara Pernici for many fruitful discussions.

References

1. D. Ardagna, C. Cappiello, M. Comuzzi, C. Francalanci, and B. Pernici. A Broker For Selecting And Provisioning High Quality Syndicated Data. In *ICIQ 2005 Proc.*, 2005.
2. A. Avenali, P. Bertolazzi, C. Batini, and P. Missier. A formulation of the data quality optimization problem in cooperative information systems. In *CAiSE Workshops (2)*, 2004.
3. R. Braumandl. Quality of service and optimization in data integration systems. In *BTW Proc.*, 2003.
4. L. P. English. *Improving data warehouse and business information quality: methods for reducing costs and increasing profits.* John Wiley & Sons, Inc., 1999.
5. M. A. Hernández and S. J. Stolfo. Real-world data is dirty: Data cleansing and the merge/purge problem. *Data Min. Knowl. Discov.*, 2(1):9–37, 1998.
6. M. Lenzerini. Data integration: A theoretical perspective. *PODS 2002 Proc.*, 2002.
7. U. Leser and F. Naumann. Query Planning with Information Quality Bounds. In *FQAS 2000 Proc.*, 2000.
8. U. Manber. *Introduction to Algorithms.* Addison Wesley, 1989.
9. F. Naumann, J. C. Freytag, and U. Leser. Completeness of integrated information sources. *Inf. Syst.*, 29(7):583–615, 2004.
10. F. Naumann, U. Leser, and J. C. Freytag. Quality-driven Integration of Heterogeneous Information Systems. In *VLDB Proc.*, 1999.
11. J. E. Olson. *Data Quality: The Accuracy Dimension.* Morgan Kaufmann, 2003.
12. M. Scannapieco, A. Virgillito, C. Marchetti, M. Mecella, and R. Baldoni. The daquincis architecture: a platform for exchanging and improving data quality in cooperative information systems. *Inf. Syst.*, 29(7):551–582, 2004.
13. M. Turner, F. Zhu, I. A. Kotsiopoulos, M. Russell, D. Budgen, K. H. Bennett, P. Brereton, J. Keane, P. J. Layzell, and M. Rigby. Using web service technologies to create an information broker: An experience report. In *ICSE04 Proc.*, 2004.
14. L. Wolsey. *Integer Programming.* Wiley, 1998.

Enhancing the Business Analysis Function with Semantics

Sean O'Riain[1] and Peter Spyns[2]

[1] Semantic Infrastructure Research Group
European Software Centre, Hewlett-Packard, Ballybrit Business Park, Galway, Ireland
Sean.ORiain@hp.com
http://h40055.www4.hp.com/galway/
[2] Vrije Universiteit Brussel, STAR Lab
Pleinlaan 2, Gebouw G-10, B-1050 Brussel, Belgium
Peter.Spyns@vub.ac.be
http://www.starlab.vub.ac.be

Abstract. This paper outlines a prototypical work bench which offers semantically enhanced analytical capabilities to the business analyst. The business case for such an environment is outlined and user scenario development used to illustrate system requirements. Based upon ideas from meta-discourse and exploiting advances within the fields of ontology engineering, annotation, natural language processing and personal knowledge management, the Analyst Work Bench offers the automated identification of, and between business discourse items with possible propositional content. The semantically annotated results are visually presented allowing personalised report path traversal marked up against the original source.

1 Introduction

1.1 Background

Business analysis is largely performed as a Business Intelligence[1] (BI) activity with data mining and data warehousing acting as the driving force in the monitoring, identification and gathering of information on topics of interest. On-line analytical processing performed on historical data allows report generation and data views, from which further BI analysis is typically performed. Data not formally mapped as part of the extract, transform and load phases, passes through the process unaltered. Current efforts to mine this unstructured data rely heavily upon problematic document level technologies such as string-based searching resulting in data being overlooked and its information value being lost. Enterprises performing customer analysis as a means to identify new business opportunities by necessity have to work their way through large volumes of free text to identify information of interest and check whether there are other informational items which are relevant to them. Current BI technologies present

[1] Term coined by Gartner in the late 1980s that refers to the user-centred process of data gathering and analysis for the purpose of developing insights and understanding leading to improved and informed decision making.

R. Meersman, Z. Tari et al. (Eds.): OTM 2006, LNCS 4275, pp. 818 – 835, 2006.

limitations in facilitating this identification and association activity when processing information. Introducing semantic technology offers the potential to address these limitations and contribute towards a more accurate and sophisticated analytical function for the analyst.

For informational analysis purposes the Form 10-Q[2] may be considered as comprising financial accounts and statements from the CEO[3]. CEO statements concern a company's performance, are seen as a promotional medium designed to present a corporate image and are hugely important in building credibility and investor confidence. They also serve to present the quantitative aspects of the financial accounts. Despite the fact that analysts have a clear expectation of information content that the statements may contain, it remains a huge task to search for, identify and filter actual relevant information due to the writing style being purposely rhetorical, argumentative and subjective. The writing style attempts to restrict the readers in developing alternative interpretations of the information presented and draws upon meta-discourse (cf. section 3.1 below) to achieve these goals, i.e. the CEO in making financial commentary would attempt to guide the reader/analyst into accepting and agreeing with the company position or view point. Despite this, an analyst engaged in the analytic process intuitively filters, refines and ultimately infers relevant information from what is being presented. Analysts are largely assisted by experience coupled with an inherent awareness and understanding of the meta-discourse employed whether consciously aware of it or not.

With information identification as the context, this paper's aim is to present the idea of a semantically enabled analysts work bench which would allow the identification of business discourse items and their relationships to other discourse items if present. The business case, technical requirements and design for such an application are presented. Development work to date is outlined along with the experimental scenario and its evaluation framework.

1.2 Business Case

HP[4] for a number of years has provided outsourced services to Independent Software Vendors (ISV's). Due to changing business practices the Business Process Outsourcing (BPO) business team was tasked with exploring the possibility of extending the current service offerings to different areas of the product development cycle where HP has considerable competencies. Which ISVs to pursue for business was determined on the basis of findings from company health checks[5]. Forms 10-Q consisting of consolidated financial information and management statements were identified as a major source for this type of information.[6]

---

[2] Quarterly report filed to the Securities and Exchange Commission (SEC) in the US. It includes un-audited financial statements and provides a continuing view of the company's financial position. Investors and financial professionals rely upon these 10-Q forms when evaluating investment opportunities.

[3] Chief Executive Officer, the highest ranking officer of a company, who oversees the company's finances and strategic planning.

[4] European Software Centre, Hewlett-Packard Galway Limited.

[5] Analysis of a company's performance and its strategic plans.

[6] Typically downloadable from a company's web site.

An initial 70 candidates were selected and subjected to a lengthy analysis process resulting in the identification of five which were then approached for business discussion. The elapsed time period for this activity was nine months. The majority of it was taken up with the identification and extraction of information of potential interest hidden within the Forms 10-Q sections containing the management's discussion and analysis of financial conditions. Contributory factors impeding this manual intelligence gathering activity were the volume of available information, its growth rate and the fact that much of the information is available in free text form only, for which no automated processing procedures have been applied until now.

There is clearly a requirement for an automated intelligence monitoring solution that would offer an opportunity to make more manageable the identification, analysis and facilitate extraction for re-use of this information within the specific domain area. This paper discusses how the use of semantic technologies and natural language processing techniques along with business specific input is currently being applied to develop such a workbench.

The remainder of the paper is organized as follows. Section 2 presents the functional requirements in terms of a user scenario development and introduces the case study. Section 3 introduces the linguistic analysis approach (subsections 3.1 & 3.2) and ontology methodology (subsection 3.3 & 3.4) adhered to. Section 4 outlines the proposed solution, its high level components and grounding technologies accompanied by a worked example covering all implementation steps from natural language identification to eventual semantic annotation. Section 5 discusses preliminary evaluation results. Section 6 presents related work areas, and finally Section 7 concludes the paper.

2 Requirements

Business requirements are presented in two steps. The first (cf. section 2.1) involves gathering high level requirements before moving to general usability and functionality considerations of which the former is only outlined here. The second (cf. section 2.2) presents the results of translating the user requirements to application design. To facilitate understanding detailed requirements functionality are presented in terms of user scenario development. This approach was also utilised to present and refine the application design with the user community.

2.1 General

The business requirement put succinctly is for an application that contributes to the analytic function by reducing the time taken and subsequently the resources necessary to accurately process and evaluate Form 10-Qs for the purpose of performing company health checks. The two main areas identified where analytic processing resource savings could be achieved were:

1. Identification and association of information items:
   The ability to perform automated analysis on the free text areas of a Form 10-Q would assist analysts in the identification of information of importance to them.

Importance was defined on two levels. Firstly as relevant information containing some element of propositional information[7] (here termed informational items). Secondly as defined relationships between these information items, allowing them to be considered collectively, which would additionally contribute towards the further identification of propositional content and hence the analytic function.

2. Personal Knowledge Management:
   Having the ability to view the level of relevant informational items and their relationships instantiations (termed semantic paths) would assist in providing the analyst with an overview of the documents information content. Path traversal to information items of interest along with the ability to view these paths in entirety, accompanied with supporting functionality allowing for information extraction and/or exclusion, would assist an analyst in structuring, managing and personalising the analysis approach.

2.2 Scenario Development

Fig. 1 represents an application mock up for the work bench based upon business requirements by the analysts involved. It comprises the three viewing areas of:

- Navigator:
  A tree view hierarchical ontology navigator allowing report traversal and selection of terms, their roles[8] and instances for viewing.
- Browser Display:
  Displays and allows dynamic traversal of the semantically annotated report with instance mark up
- Relationship Viewer:
  A tree view hierarchical ontology navigator which displays the roles between terms along with the text associated with either the role or term instance.

An analyst in performing a company health check analysis will typically scan the introductory section to gain an insight as to what sections within the report may contain useful information. The remainder of the report is then systematically gone through using these identified areas to guide analysis function on. The major difficulty faced is the search for, identity of and filtering of actual relevant information involving substantial document traversal through large volumes of free text. The viewing areas presented will provide the functionality to perform these activities. In a prototypical case, an analyst having loaded the Form 10-Q will have the report automatically processed and annotated (i.e. enriched with meta-data related to the typical content of a Form 10-Q) based upon the argumentation categories and term/roles like those listed in section 4.1.

Using the **Navigator** to traverse a report allows context specification (here '*Sales*') and provides a listing of all associated terms and roles. As illustrated in Fig. 1, further

[7] This is information about thoughts, actors or state of affairs outside of the text [15]. Here interpreted as information that may contribute towards an understanding of the actual information content in the text.

[8] DOM^2 terminology, refer to section 3.4.

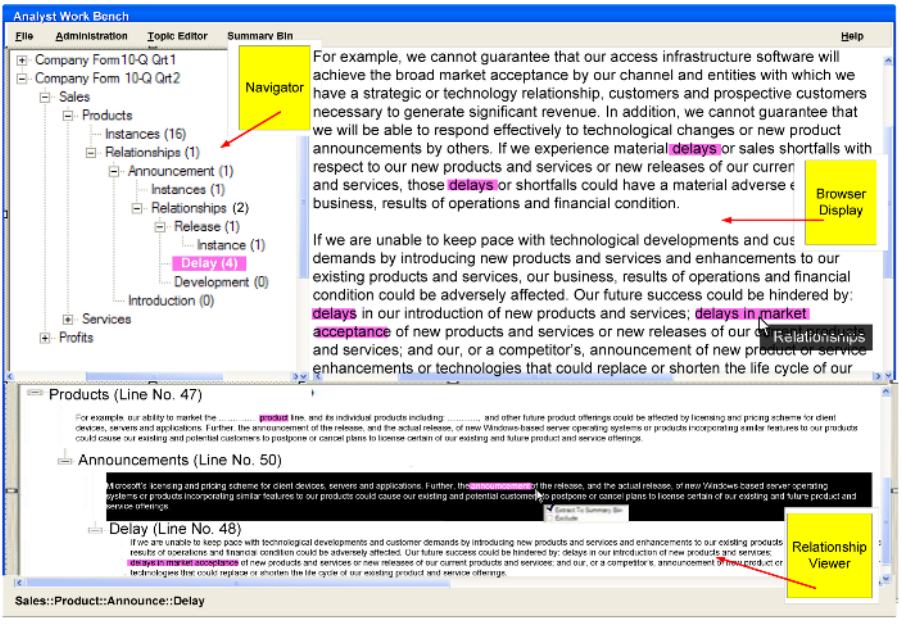


Fig. 1. AWB Main in-text Visualization Area

traversal provides indicators as to which information items such as '*Products*' have multiple instances within the report. The indicators also provide a summary of which anticipated partial semantic paths binary relationships are actually instantiated (here '*Products*' has one instantiated relationship '*Announcements*', '*Announcements*' in turn has two instantiated relationships, '*Release*' and '*Delay')*. In effect the analyst has the capability to select and instantiate semantic paths[9] through the document.

Individual or multiple term or role annotation instances selected for viewing (here '*Delay*') will be highlighted with a background colour in the report displayed within the **Browser Display**. Within this area the analyst can easily identify and select a particular annotation instance (here '*market delay*') to view its instantiated in-text binary relationships within the Relationship Viewer.

The **Relationship Viewer** taking the original context ('*Sales*') as its root presents the semantic traversal path between terms in the binary relationship (cf. section 4.1). Based upon the ontology structure used by the Navigator, it additionally includes the annotation instance information item as part of the tree view allowing the analyst to build up a complete picture of in-context business informational items, without the overhead of having to actually consider the context. The overall in-text viewing capability allows quick identification of informational items and their binary relationships which when considered collectively (i.e. the semantic path) offer the best opportunity to access their actual propositional content value. Once accessed semantic paths terms, binary relationships or specific information item instances can

[9] Currently the functionality for the partial semantic path recognition is under development.

be selectively excluded and filtered out from further consideration. Exclusion used in this manner provides the analyst with varying degrees of flexibility in managing the information overload as part of the analytic process.

3 Methods

The critical research question is whether it is possible to express what the analyst is looking for in terms of meta-discourse functions, and if so, to what extent can it be automated. Expressing analyst information needs therefore requires an understanding of how meta-discourse and its function intrude into the activity. It is the identification of this propositional information (but stripped from the rhetorical and subjective elements) that is the overall goal of the Analysts Work Bench (AWB). To achieve this aim, a multidisciplinary approach combining natural language processing and ontology engineering is proposed. The following sections describe the theories, methodologies and tools used in this experiment.

3.1 Linguistic Analysis

Hyland defines meta-discourse as a linguistic tool that creates a textual structure that goes beyond the statement of the subject matter and gives clues as to the purpose and attitude of the writer [18]. In effect meta-discourse consists of text tokens that do not contribute to the propositional development of the text but serves to guide the reader in interpretation and response to the text [17]. In describing the functional categories of meta-discourse, Hyland incorporates Thompson's classification to model meta-discourse as comprising the functional categories of interactive and interactionable [17,24]. *Interactive* refers to the writers attempts at constraining the text to their preferred interpretation and goals. Resource usage is this category is concerned with how discourse is organized and the extent to which the text is structured with knowledge of the reader and their needs in mind. Interactive resources are used by the writer to organize propositional information in a manner that the reader is likely to find coherent and convincing, effectively managing the information flow. *Interactionable* refers to the level of writer intrusion into the text by way of comment, opinion and evaluation [18]. Interactionable resources are employed to anticipate, acknowledge, challenge or negate alternate interpretations being drawn, in effect restricting opportunities for alternate views in the first instance. Table 1 below details the meta-discourse categories and their functional descriptions.

Hyland uses the proposition meta-discourse distinction as a starting position for academic meta-discourse exploration, but others have included propositional content as part of meta-discourse [5,18]. To further blur the issue it is often the case that meta-discourse and propositional elements occur within one sentence and what is considered propositional in one context is meta-discourse in another.

Results from the study of meta-discourse within CEOs' letters indicate that the functional devices of *transitions* and *hedges* (see Table 1) together account for 66% of all discourse items [17]. Within the area of business postgraduate studies, this figure

Table 1. Model of Meta-discourse in Academic Texts (adapted from [18])

Interactive Resource		
Category	**Function**	**Device lexicalisation**
Transitions	Express semantic relation between main clauses	And/or/but /in addition/thus
Frame Markers	Draw attention to discourse goals or indicate topic or argument shifts	Finally/to conclude/my purpose here is to/ I argue here/ Well now
Endophoric Markers	Refers to information in other parts of the text	See section X/noted above/see Fig X/In section X
Evidentials	Refer to source of information in other parts of the text	According to X/ 2005, X/X states
Code Glosses	Assist reader in interpretation of ideational material	Such as/for instance/in other words/namely/e.g.
Interactional Devices		
Hedges	Withhold writers full commitment to proposition	Possible/might/perhaps/about
Boosters Emphatics	Emphasises force or writers certainty in proposition	In fact/it is obvious/definitely/ clearly/it is clear that
Attitude markers	Express writers attitude to proposition	Unfortunately/I agree/agreement/ surprise/surprisingly
Engagement markers	Explicitly refer to or build relationship with reader	Consider/note that/you can see
Self mentions	Explicit reference to author(s)	I/we/my/our

rises to 90% [5]. While we were unable to use the meta-discourse devices themselves as a means to assist in propositional content identification, we were however able to use the idea of the transitions and hedges functional categories when constructing the ontology and subsequent grammar rules (expanded upon in section 3.3).

3.2 The GATE Linguistic Engineering Framework

The General Annotation for Text Engineering (GATE)[10] is a component-based general architecture and graphical development environment for natural language engineering. Provided APIs allow selective inclusion of languages, processing and visual resources components such as tokenisation, semantic tagging, verb phrase chunking and sentence splitting. GATE's Java Annotation Patterns Engine (JAPE) providing finites state transducers over annotations, allows grammar rule specification and recognition of regular expressions within these annotations. The annotations organised as a graph are modelled as java sets of annotations facilitating manipulation. The last decade has seen the acceptance of shallow analysis techniques involving pattern analysis and regular expressions. GATE provides flexibility

[10] May be downloaded from http://gate.ac.uk. Further details on GATE may be found in [1].

allowing adaptation to activities such as these and to follow on activities such as ontology based information extraction – e.g., for technology watch [20]. It is for these reasons that we selected GATE.

3.3 Ontology Modelling

Reports written in the language of business discourse ensure that analysts face the problem of first identifying information items of interest and then attempting to interpret their actual business message. In tackling identification within this context we adopted the idea of the meta-discourse *hedges* category and used it predominately as an assist to the analyst in targeting sentence clauses that were making a contribution to actual information content. Complicating the problem was the fact that any single information item provides only part of the overall proposition being made and the level of company commitment to it. Interpretation therefore had to be based upon considering relevant information together, requiring a level of semantic association between items that adhered to business logic. Consequently we drew upon the meta-discourse *transition* category for semantic association and hedges category for assisting the analyst in introducing business logic. Combining both approaches offered the possibility to draw out from the manner in which something is being said, what in fact is being said. It provides the analyst with the ability to actually determine what proposition is being made. (cf. section 4.1 for results of these influences).

Adhering to this novel approach, domain experts were used to manually construct a taxonomy for business argumentation categories and their lexicalisation. Due to the domain of application requiring deep background knowledge of the discourse used, the inclusion of large vocabularies which would eventually require gazetteers was purposely avoided and domain specific terms and phrases considered only. This helped to address a secondary problem introduced by the nature of discourse itself, namely that uniform use of synonyms was not possible or practicable in all circumstances as they can and do vary in their meaning. E.g. While the phrase ‘delays some_text introduction’ or ‘delays in market acceptance’ could be rephrased with a delay synonym of postpone, neither could be rephrased with other “delay” synonyms such as “wait/stay/check” and still retain their intended business meaning.

Wishing to build upon database modelling and development experience as a means of bridging the knowledge gap between database and ontology design, we required an ontology development methodology that would facilitate the transition. Resultantly the DOGMA Ontology Modelling Methodology (DOM^2) [30,31], inspired by the principles of Object Role Modelling (ORM) [14] [11] and aN Information Analysis Method (NIAM) [33], was selected. The ontology engineering process resulted in the manual construction of a high level domain ontology (see Table 3 and Table 4 for an example extract).

3.4 The DOGMA Ontology Engineering Framework

Developing Ontology Guided Mediation for Agents” or DOGMA [23] has as its core the notion of *double articulation*, which decomposes an ontology into an *ontology base* (intuitive binary and plausible conceptualisations of a domain) and a separate

[11] Familiarity with ORM and its use in Relational Database Design is assumed.

commitment layer, holding a set of instances of explicit ontological commitments (or domain rules) for an application [29,31].

The ontology base
An ontology base consists of intuitively plausible conceptualisations of a real world domain, i.e. specific binary fact types, called *meta-lexons*, formally noted as a triple <$concept_1$ – relationship – $concept_2$>. They are abstracted (see below) from *lexons*, written as sextuples <(γ,ζ): $term_1$, role, co-role, $term_2$>. Informally we say that a *lexon* is a fact that may hold for some domain, expressing that within the context γ and for the natural language ζ the $term_1$ may plausibly have $term_2$ occur in *role* with it (and inversely $term_2$ maintains a *co-role* relation with $term_1$) [30] (example in subsection 4.1). Lexons and meta-lexons are meant to reach a common and agreed understanding about the domain conceptualisation (important/relevant notions and how they are expressed in one or more natural languages [22]) and assist human understanding.

Lexons are independent of specific applications and should cover relatively broad domains (linguistic level). They form a lexon base, which is constituted by lexons grouped by context and language [29]. Meta-lexons are language-neutral and context-independent (conceptual level). Natural language terms are associated, via the language and context combination, to a unique word sense represented by a concept label (e.g. the WordNet [9] identifier *person#2*). With each word sense, a gloss or explanatory information is associated that describes that notion. To account for synonymy, homonymy and translation equivalents there is an m:n relationship between natural language terms and word senses [30]. Going from the language level to the concept level corresponds with converting the lexons into meta-lexons.

The commitment layer
The layer of ontological commitments mediates between the ontology base and its applications. Each commitment is a consistent set of rules (or axioms) that add specific semantics to a selection of meta-lexons of the ontology base [19]. The commitment layer, with its formal constraints, is meant for interoperability issues between information systems, software agents and web services, as is currently promoted in the Semantic Web area. The constraints are mathematically founded and concern rather typical DB schema constraints e.g. cardinality, optionality etc. While we note the purpose and function of the commitment idea and indeed use it (cf. section 4.1), our implementation does not require the formal definition of rules. The selection of application relevant meta-lexons to form a commitment rule while corresponding to the notion of a semantic path differs as it requires formal constraints to instantiate the particular commitment. The semantic path does not. Simply stated, the domain model as represented in the ontology base can be richer than the actual content of the semantic paths.

4 Proposed Solution

The workbench primary functions are firstly information identification and secondly information extraction. Fig. 2 provides a high level overview of the workbench conceptual architecture and its natural language processing components.

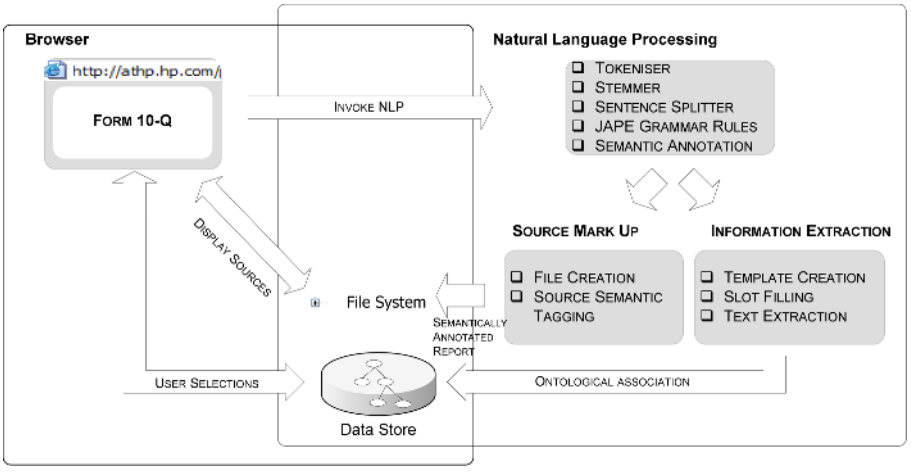


Fig. 2. The AWB High Level Component Modules

The work bench browser component is responsible for display and user related interaction while the Natural Language Processing (NLP) component is responsible for NLP, source mark up and information extraction (IE). The NLP module tokenizes the report, stems the tokens and performs sentence splitting. JAPE grammar rules identify and categorize recognized patterns to concept categories. The source mark up module semantically marks up the patterns found in the source report and renders the annotated result to the AWB browser for display. The IE module using the ontological concept structure is responsible for template slot filling (cf. Table 5) and ontology population. The applications data store can be a relational database, RDF data store (e.g., [3,4]) or both.

4.1 Prototype Development

In conjunction with domain experts, (syntactic) analysis of information sources (e.g. the report text displayed in Fig. 1) was embarked upon to identify domain specific categories, terms and binary relationships. Applying the DOGMA philosophy as explained in section 3.4 to the analysis results, the ontology engineer manually constructed a series of lexons such as the one presented in Table 2. The approach taken was that searching for a combination of these patterns would provide the best opportunity for propositional content identification. Table 2 represents the lexon set for the semantic path from *'Product*' to '*Announcement*' to '*Delays*'. From an analyst's view point the lexon extract (expressed as elementary notions) would be interpreted as: When constructing a picture of the sales area, references[12] to *products* are of interest. To understand what is occurring in the '*Product'* space, references to '*Announcements'* are of interest. Lastly for '*Announcements*', references to items that bring to attention '*Developments*', '*Releases'* or '*Delays*' are important.

[12] Term purposely used to simplify dealings with the business community that refers to the tacit binary relationship.

Table 2. Sales Lexon Extract

Context (γ) = Sales, Language (λ) = UK English			
Head term (t_1)	**Role (r_1)**	**Co-role (r_2)**	**Tail term (t_2)**
Product	Follows	Precedes	Announcement
Product	Is_described_by	Describes	Announcement
Announcement	Publicises	Is_announced_in	Delay*
Announcement	Publicises	Is_announced_in	Release
Announcement	Publicises	Is_announced_in	development

Next, synsets (as used in WordNet [9]) and natural language explanations and glosses were introduced allowing the grounding of concepts and relationships (cf. Table 3) and the creation of meta-lexons $<C_1, R_1, C_2>$ afterwards. Conversion of lexons into meta-lexons copes with ideas expression in several ways, e.g. by morphology (whether inflection or conjugation allowing different forms of the same word), by synonymic wording (lexicology / terminology) or by syntax such as organising a sentence using a noun instead of a verb. As there was no requirement[13] to have multi–lingual capability the principle of having ontologies transcend in so far as possible specific linguistic influences[14] has not been strictly adhered to. Resultantly schema mapping issues with different language usage where words in one language do not have a direct equivalent lexical translation in another necessitating paraphrasing were not of concern[15] and allowed us combine the role and co-role into a single semantic relationship. The labels assigned to the concepts and relationships do not mimic any existing language expressions [25] – cf. Table 3.

Due to issues with synonyms introduction (see section 3.3) and the lack of requirement to cater for multi-lingual aspects, the combination of language and context normally associated with term disambiguation was only used as a means of moving from the language to concept level. The resulting meta-lexons based upon the Sales Lexon Extract in Table 2 are shown in terms of their constituent concept labels (as established in Table 3) in Table 4, along with invoked sample grammar rules.

Automated concept identification was implemented as a set of JAPE grammar rules using the GATE tool (see section 3.2). The grammar rules provided the implementation vehicle for the meta-lexons that correspond to the semantic path building blocks. Referring to the row indicated by an astrix in Table 2 and Table 4, Rule 1 provides the JAPE rules to recognise the concept "*announcement*" while Rule 2 detects the presence of *"delays in market acceptance"*. The former rule uses the macro *(ANNOUNCE)* based upon stemming[16] and synonym expansion to identify the inflected word form for annotation. Once found the right hand side of the rule

[13] Form 10-Qs are a requirement for US companies only

[14] Due to the difficulty of such an exercise, some authors argue for language neutral representations rather than language independent ones [160].

[15] See [80] for details on how multilinguality is handled within DOGMA.

[16] GATE's provided stemmer plug-in is based upon the Porter stemmer for English.

Table 3. Grounding of concepts (excerpt)

Context (γ) = Sales, Language (ζ) = UK English	
Label	**{Explanation; gloss; synonyms etc.}**
C1002	Item manufactured/made and sold to general public; Saleable item; item, goods
C1005	Public statement made for public consumption; Scheduled event; Inform, proclaim, advise
C1014	Delay in market indicating reduction in average order in take; Reduction in order intake; No synonyms
R1001	Set of Program to expand; Intention; Postponement
R1003	Cancel or reduce; Order; Timely
R1005	Cycle; Resources; Market; Longer product

Table 4. Sales Meta-lexon

C_1	R	C_2		
C1002	R1001	C1005	Rule: C1005 ((ANNOUNCE)) : C1005--> : C1005.C1005	**Rule 1**
C1002	R1002	C1005		
C1005	R1003	C1014*	Rule: C1014 ((DELAY) (SPACE_WORD_SPACE)* ((MARKET) (ACCEPT)) : C1014--> : 1014.C1014	**Rule 2**
C1005	R1004	C1016		
C1005	R1005	C1017		

(denoted by '→') fires to annotate the pattern with the ontology concept label '*C1005'*. The latter rule working in a similar manner first recognising *(DELAY)* followed by multiple white spaces and noise words before recognising occurrences of *(MARKET)* and *(ACCEPT)*. The entire concept is then given the label *'C1014'*.

The JAPE rules cope with morphological variations. General synonym expansion using WordNet [9] produced "noise" in intermediary results due to issues of general language knowledge applicability. Resultantly synonym expansion introduction was only possible with analyst selection and agreement[17]. Subsequently, a combination of scenario and relational templates are used to reflect abstract representations of the domain and present an instantiated part of the ontology. Table 5 provides such a reduced Information Template used for ontology population based upon the meta-lexons introduced earlier in Table 4.

Instantiated templates can be built per report source from which specific IE actions and expert system reasoning (here business analysis) can be performed e.g., checking

[17] After this annotation stage more complex JAPE rules operating on the concept level could be applied to add specific tags for meta-lexon instances indicating partial semantic paths. We opted however to not search for implicit relationship at this point but leaving them to the ontological association stage where they are based upon both semantic and business logic.

Table 5. Information Template (Reduced)

Announcement	
InstanceID	Auto generated
InfoItem	Further, the announcement of the release … and the actual release, of new Windows-based server operating systems or products incorporating similar .. could cause our existing ….
LinkedConcept1	C1005
LinkedRelationship	R1002
LinkedConcept2	C1014
Delay in market acceptance	
InstanceID	Auto generated
InfoItem	If we are unable to keep pace with technological developments … hindered by: delays in our introduction of new products… delays in market acceptance of new products and services or new releases …
LinkedConcept1	C1014
LinkedRelationship	R1003
LinkedConcept2	C1005

of whether or not a company's products, already announced, suffer from a delay in being brought to market.

Template slots essentially act as database records lending themselves easily to SQL operations within an RDB context or RDF triples if using an RDF store (e.g., [3,4[18]]). Construction of closed semantic paths with specific meta-lexons in this manner ensures inherent semantics and a bounded view. In this regard the templates themselves and the semantic path correspond closely to the DOGMA notion of a *commitment* bridging the gap between the ontology base and application layers and representing a particular view of the ontology. Here the ontology concepts themselves as defined by a semantic path perform the constraint function, removing the necessity for formalised rules as only a selection of relevant meta-lexons are needed.

5 Preliminary Evaluation

The requirements identify the two areas of identification and association of information items and Personal Knowledge Management as areas where business resource savings are possible. As the application is still under development the discussion will be limited to a preliminary (and reduced in scope) validation of the initial research hypothesis, i.e. that a combination of NLP, IE and ontology technology contributes to a tangible resource reduction for the business analysts in gathering business intelligence from company reports. The purpose of the evaluation

[18] May be down loaded from http://jena.sourceforge.net/index.html.

exercise was to gain an initial indication as to the tool's usefulness and whether overhead associated with the initial setup was justifiable. Therefore, we only involved a single senior annotator expert. Thus, for reasons of methodological soundness, precision and recall have not been calculated as such. In the future, a thorough evaluation according to the method described in [11] will be conducted.

In performing the evaluation (see. Table 6 for results) we had a senior domain expert analyse manually 10-Q forms[19], with the instruction to identify and annotate information items based upon partial semantic path recognition. The semantic path used was the full sales meta-lexon extract of which Table 4 mentioned previously is only an excerpt. From these the analyst performed further expert validation to identify only those items of *actual* relevance. The same activity sequence was then performed on reports that were automatically annotated to identify partial semantic paths. The outcomes have been evaluated in a three-fold manner. The first phase involved the performance comparison of manual vs. automated annotation, while the second measured the overall impact of the tool on the business intelligence activities (manual vs. automated relevance). Finally, the time needed by a business analyst to perform his/her task with and without tool support has been compared (cf. Table 7).

The tool supported analysis outperformed the manual by 34% translating in actually relevant terms (8 vs. 12) to an increase of 50% in the number of additional informational items annotated by the analyst (11 vs. 22). Further expert analysis on these additional relevant items indicate that they directly contributed to reinforcing analysis hypothesis and most significantly in one instance, brought to attention an item that led to new analysis thinking.

Table 6. Partial Semantic Path Identification results

	Information Item No.			
	Manual		**Automated**	
Concept	**Annotated**	**Relevant**	**Annotated**	**Relevant**
C1002	2	2	8	4
C1005	1	1	2	1
C1011	0	0	0	0
C1012	0	0	2	0
C1013	0	0	0	0
C1014	4	3	4	4
C1015	2	0	2	1
C1016	2	2	4	2
C1017	0	0	0	0
Totals	11	8	22	12

Table 7 lists the timings taken for an analyst performing the report analysis activity mentioned previously, first manually and then after automated semantic annotation. Automation resulted in a 78% overall resource saving over manual activities and removed completely the need for the introductory section analysis.

[19] Based upon the average of a year's quarterly reports for one company.

Table 7. Resource Saving

	Time (minutes)			
Report section	**Manual**	**Automated**	**Difference**	**Resource Saving (%)**
Introduction	10	ignored	-10	100
Main body	80	20	-60	75
Totals	90	20	-70	78

6 Related Work

The application of IE techniques to SEC Form 10-Q business reports is principally similar to the MUC evaluation exercises (e.g. [15]). Recent efforts such as OntoText's KIM [26] have extended the classical (but limited) nature and number of entity recognition sets to include proper names, geographical names, business names etc. These sets however remains insufficient for the nature and type of information that an analyst would wish to extract from the Form 10-Q.

The medical field in particular has seen earlier large scale research efforts conducted to extract complex knowledge from documents (e.g., cf. [10,27]). Typical of these projects is that they draw upon "deep" linguistic and semantic analysis along with well developed models of knowledge representation and reasoning. The last decade has seen the emergence and adaptation of light weight pattern analysis techniques based upon regular expressions such as RegExTest[20] applied to the identification of Nigerian fraud emails [13]. Shallow NLP analysis techniques, in many cases involving pattern analysis and regular expressions such as the GATE system have become mainstream [6].

A more recent step has been the combination of ontologies with information templates as used in IE [21]. The template fillers are instances of a domain ontology concepts of which a selection constitutes the building blocks of a template. In a further step reasoning procedures can be added to the templates (turning the templates into frames in the tradition of Schank [28]. A rule based IE application concerned with market monitoring and technology watch within the employment market place, obtained promising results (precision 97%, recall 92%), be it on a small test-bed of documents [20]. Moreale using the same approach but within an e-learning context, constructed a service that presented the argumentation content of a student essay [24].

Typical of these prototypes is that while the results were said to be favourable, in each case the conclusions were that the tool required further work for large scale deployment. These findings confirm earlier results, namely that IE is improved when based upon an ontology [14,20]. Having conducted literature reviews, we remain unaware of a similar application such as the AWB in a business setting as described above that has as its basis a pressing business case and represents for HP a considerable resource saving opportunity and intelligence gathering assist for the Business Process Re-Engineering Group.

[20] http:// sourceforge.net/projects/regextest/.
[21] A good introduction on combining ontologies with IE can be found in [70].

7 Conclusion

The paper's main contribution is the outline of the AWB prototype offering semantically enhanced analytical capabilities to the business analyst based upon the notion of semantic paths. Preliminary 'in development' findings have been presented outlining promising results of greatly enhanced relevant information identification and association capabilities which resulting in both resource savings and new analysis insight. As efforts to date have been directed toward the partial semantic path we now aim to expand upon this to high light the semantic relationship by implementing the complete semantic path. We are confident that evaluation on the resultant fully functional prototype will reinforce these findings. The other key learning has been that a blanket adaptation of NLP for particular domains (here Business Process Outsourcing) will be problematic without tailored usage of synonym expansion, particularly if a complex understanding of discourse within the domain is required. With the idea of semantic paths providing encouraging results even at this early developmental stage, investigative research into functionality allowing restricted domain natural language querying and reasoning based upon the ontology being expressed in OWL is currently underway.

Acknowledgements

We gratefully thank John Collins, Business Development & Business Engineering Manager, HP Galway, for his evaluation effort, and David O'Sullivan, DERI Galway along with Robert Meersman VUB-STAR Lab, for their comments on draft versions of this text.

References

1. Bontcheva K., Tablan V., Maynard D. & Cunningham H., (2004), Evolving GATE to meet new challenges in language engineering, Journal of Natural Language Engineering 10 (3/4): 349-373
2. Broekstra J., Kampman A., van Harmelen F., (2002), SESAME: A generic architecture for storing and querying RDF and RDF Schema, in Horrocks I. & Hendler J., (eds.), Proc. of the First International Semantic Web Conf. (ISWC02), LNCS 2342, Springer, pp. 54-68
3. Cao T.-D., R.D.-K. & Fiès B. (2004), An Ontology-Guided Annotation System for Technology Monitoring. in Proc. of IADIS International WWW/Internet 2004 Conference.
4. Caroll J., Dickinson I., Dollin C., Reynolds D., Seaborne A., Wilkinson K., (2004), Jena: implementing the semantic web recommendations, Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters
5. Crismore A. and Farnesworth R, (1990), Metadiscourse in popular and professional science discourse. The Writing Scholar, Studies in Academic Discourse, in W. Nash, (ed.),. New Bury Park: Sage. 119-36
6. Cunningham H., Maynard D., Bontcheva K. & Tablan V., (2002), GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications, in Proc. of the 40th Anniversary Meeting of the Association for Computational Linguistics
7. Cunninham H., Bontcheva K. & Li Y., (2005), Knowledge Management and human language: crossing the chasm, Journal of Knowledge Management, 9 (5): 108-131

8. De Bo J., Spyns P. & Meersman R., (2003), Creating a "DOGMAtic" multilingual ontology infrastructure to support a semantic portal. in Meersman R., Tari Z. et al., (eds.), On the Move to Meaningful Internet Systems 2003: OTM 2003 Workshops, LNCS 2889, Springer Verlag, pp. 253 - 266
9. Fellbaum C. (ed.), (1998), Wordnet, An Electronic Lexical Database, MIT Press
10. Friedman C., Hripcsak G., Alderson P., DuMouchel W., Johnson S. & Clayton P., (1995), Natural Language Processing in an operational clinical information system, Journal of Natural Language Engineering 1 (1): 83 - 103
11. Friedman C. & Hripcsak G., (1998), Evaluating Natural Language Processors in the Clinical Domain, Methods of Information in Medicine 37 (4/5): 334 - 44
12. Gao Y & Zhao G., (2005), Knowledge-based Information Extraction: a case study of recognizing emails of Nigerian frauds, in Montoyo A., Munoz R. & Metais E., (2005), Proceedings of the 10th International Conference of Applications of Natural Language to Information Systems (NLDB05), LCNS, 3513, Springer, pp. 161 – 172
13. Guarino N., Masolo C. & Vetere G., (1999), OntoSeek: Content-Based Access to the Web, IEEE Intelligent Systems, (4-5): 70-80
14. Halpin T, (2001), Information Modeling and Relational Databases: from conceptual analysis to logical design, Morgan-Kaufmann, San Francisco.
15. Hirschmann L. (1998), Language Understanding Evaluations: Lessons learned from MUC and ATIS, in Rubio A., Gallardo N., Castro R. & Tejada A. (eds.), 1st International Conference on Language Resources and Evaluation (LREC 98), ELRA, pp. 117-122
16. Hovy, E. & Nirenburg S. (1992). "Approximating an interlingua in a principled way". Proceedings of the DARPA Speech and Natural Language Workshop, http://www.isi.edu/natural-language/people/hovy/papers/92darpa-il.pdf
17. Hyland, K., Exploring corporate rhetoric: metadiscourse in the CEO's letter. The Journal of Business Communication, 1998
18. Hyland K. and Tse P., Metadiscourse in Academic Writing: A Reappraisal. Applied Linguistics, 2004. 25 (2): p. 156-177
19. Jarrar M. & Meersman R., (2002), Formal Ontology Engineering in the DOGMA Approach, in Meersman R., Tari Z. et al., (eds.), On the Move to Meaningful Internet Systems 2002: CoopIS, DOA, and ODBASE; Confederated International Conferences CoopIS, DOA, and ODBASE 2002 Proceedings, LNCS 2519, Springer, pp. 1238 – 1254
20. Maynard D, et al. Ontology-based information extraction for market monitoring and technology watch. in ESWC Workshop "End User Apects of the Semantic Web". 2005..
21. McGuiness D., (2004), Question Answering on the Semantic Web, IEEE Intelligent Systems Jan/Feb 2004: 82-85
22. Meersman R., (1999), The Use of Lexicons and Other Computer-Linguistic Tools, in Zhang Y., Rusinkiewicz M, & Kambayashi Y., (eds.), Semantics, Design and Cooperation of Database Systems, Proceedings of CODAS 99, Springer Verlag, pp. 1 – 14.
23. Meersman R., (2001), Ontologies and Databases: More than a Fleeting Resemblance, In, d'Atri A. and Missikoff M. (eds), OES/SEO 2001 Rome Workshop, Luiss Publications.
24. Moreale E. and Vargas-Vera M., Semantic Services in e-Learning: an Argumentation Case Study. Internat. Forum of Educational Technology & Society, 2004. 4 (7): p. 112-128
25. Nirenburg, S. & Raskin V. (2001). "Ontological Semantics, Formal Ontology, and Ambiguity". Proceedings of the Second International Conference on Formal Ontology in Information Systems. ACM Press, 151 – 161
26. Popov B., Kiryakov A., Kirilov A., Manov D., Ognyanoff D. & Goranov M., (2003), KIM: Semantic Annotation Platform, in Proceedings of the 2nd International Semantic Web Conference (ISWC 03), Springer Verlag, pp. 484-499

27. Sager N, Friedman C & Lyman M., (1987), Medical Language Processing: computer management of narrative data, Addison Wesley
28. Schank R. & Abelson R., (1977), Scripts, Plans, Goals and Understanding, Lawrence Erlbaum Associates, Hillsdale N.J.
29. Spyns P., Meersman R. & Jarrar M., (2002), Data modelling versus Ontology engineering, in Sheth A. & Meersman R. (ed.), SIGMOD Record Special Issue 31 (4): 12-17
30. Spyns P., (2005), Adapting the Object Role Modelling method for Ontology Modelling . In, Hacid M.-S., Murray N., Ras Z. & Tsumoto S.,(eds.), Foundations of Intelligent Systems, Proceedings of the 15th International Symposium on Methodologies for Information Systems, LNAI 3488, Springer Verlag, pp. 276 – 284
31. Spyns P., (2005), Object Role Modelling for Ontology Engineering in the DOGMA framework, in Meersman R., Tari Z., Herrero P. et al. (eds.), Proceedings of the OTM 2005 Workshops, LNCS 3762, Springer Verlag, pp. 710 - 719
32. Thompson, G., Intreraction in academic writing: Learning to argue with the reader. Applied Linguistics, 2001. 22 (1): 58-78
33. Verheyen G. & van Bekkum P., (1982), NIAM, aN Information Analysis Method, in Olle T., Sol H. & Verrijn-Stuart A. (eds.), Information Systems Design Methodologies: A Comparative Review, North-Holland/IFIP WG8.1, pp. 537—590

Ontology Engineering: A Reality Check

Elena Paslaru Bontas Simperl[1] and Christoph Tempich[2]

[1] Free University of Berlin, Takustr. 9, 14195 Berlin, Germany
paslaru@inf.fu-berlin.de
[2] Institute AIFB, University of Karlsruhe, 76128 Karlsruhe, Germany
tempich@aifb.uni-karlsruhe.de

Abstract. The theoretical results achieved in the ontology engineering field in the last fifteen years are of incontestable value for the prospected large scale take-up of semantic technologies. Their range of application in real-world projects is, however, so far comparatively limited, despite the growing number of ontologies online available. This restricted impact was confirmed in a three month empirical study, in which we examined over 34 contemporary ontology development projects from a process- and costs-oriented perspective. In this paper we give an account of the results of this study. We conclude that ontology engineering research should strive for a unified, lightweight and component-based methodological framework, principally targeted at domain experts, in addition to consolidating the existing approaches.

1 Introduction

The emergence of the Semantic Web has marked an important step in the evolution of ontologies. Regarded as a means for a shared knowledge understanding and a way to (formally) represent real world domains, they are expected to play a crucial role in data and application integration at public and corporate level. In the last decades researchers have proposed process methodologies for various ontology engineering scenarios [7]. Given the difficulties related to building and maintaining ontologies, a methodological framework provides important benefits: it structures the process, thus breaking its complexity down to manageable tasks, clarifies the responsibilities of the process participants, increases its traceability and enables systematic quality assurance procedures.

The theoretical results achieved in the ontology engineering field in the last fifteen years are of incontestable value for the prospected large scale take-up of semantic technologies. Their range of application in real-world projects is, however, so far comparatively limited, despite the growing number of ontologies online available.[1] This restricted impact was confirmed in a three month empirical study, in which we surveyed 34 recent ontology engineering projects from industry and academia in order to give an account of the current ontology engineering practice and of the efforts involved in these activities. The study focused on process-related rather than modeling issues; in particular it analyzed the impact of actual research achievements on real world ontology engineering projects, the complexity of particular ontology building tasks, the quality of the support tools, and the various usage scenarios for ontologies.

[1] Refer for example to http://swoogle.umbc.edu/ for recent statistics on this topic.

R. Meersman, Z. Tari et al. (Eds.): OTM 2006, LNCS 4275, pp. 836–854, 2006.

The majority of the investigated case studies did not follow a systematic approach to ontology building, because the participants each underestimated the associated efforts or were not aware of the availability of methodological support. Nevertheless, the inventory of activities carried out in each of the ontology engineering projects largely overlapped with the ones described in the literature—though the concrete order of execution of these activities or the way they were combined were not necessarily the same as foreseen within academic process descriptions.

Accounting for the experiences gained during the survey, this paper argues that there is a need for a unified ontology engineering process model—following the analogue development in the software field, which moved from a multitude of different process models towards the idea of *method engineering*. Complementarily to initiatives aiming to outreach existing results to the industry, ontology engineering research should consider aligning previous disparate efforts in order to provide real added value to the community of domain experts building ontologies. Methodologies should offer their applicants support at a level of detail which is adjusted to the complexity of each ontology development activity and to the challenges of the project setting. They should be customizable to various special needs, such as, the learning of ontologies from text, and should concentrate on providing a comprehensive range of methods which can be arbitrarily combined and exchanged rather than postulating static process models.

The remainder of this paper begins with a review of analytical and empirical evaluations of existing methodologies in Section 2. After a brief description of ontology engineering processes in Section 3, we present our survey and discuss its results in Section 4. We draw conclusions for future research in Section 5. Section 6 summarizes our work and concludes this paper.

2 Related Work

This section introduces surveys on ontology engineering methodologies and empirical ontology engineering case study descriptions previously published in the literature.

2.1 Analytical Surveys

This paper is concerned with the impact of existing methodologies on current ontology engineering practice. Previous work primarily focused on the analytical evaluation of these approaches. The surveys defined a number of criteria derived from various sources and evaluated methodologies accordingly. By contrast, our considerations are based on case study experiences distilled from multiple expert interviews.

[11] summarizes the main ontology engineering activities covered by the methodologies available at that time. The authors identify the need for guidance on ontology reuse.[2] They further demand that ontology engineering methodologies should not be based on singular project experiences, but should be applied in more settings in order to claim generality. They argue that methodologies become useful for practitioners only with a larger record of projects successfully carried out. They conclude that most methodologies offer some guidance on the major engineering activities, but that there

[2] By 1998 no methodologies for ontology reuse had been proposed yet.

is plenty of scope for refinement. [3] compare different ontology engineering methodologies w.r.t. the granularity of their process descriptions. They introduce a series of ontology engineering activities, classified in the categories "ontology management", "ontology development" and "ontology support", and analyze which methodologies implement which activities to which extent. They conclude that no methodology covers all required ontology engineering activities, and that tool support is still missing for most of the analyzed methodologies (due to 2003). More recently, [25] identified requirements on ontology engineering methodologies to support the development of ontologies for knowledge management applications. The authors compare the support offered by existing methodologies against these requirements and outline a number of open issues for further research and development. In particular, they emphasize that current methodologies are not integrated into classical business process models and do not take into account this dimension to a satisfactory extent.

In summary, analytical surveys identify open issues for ontology engineering methodologies *from a theoretical perspective*. This is orthogonal to our empirical approach, which examines the utility of existing methodologies for current practice, thus forming the basis for new analytical evaluations in the future.

2.2 Empirical Studies

In the following we give an overview of the most prominent case studies related to ontologies which have been published in the Knowledge/Ontology Engineering literature from the early nineties to now. Claiming by no means for completeness, this overview concentrates on *empirical* studies reporting on concrete experiences in developing or deploying ontologies—with or without the help of a specific methodological framework. Our aim is to point out the practical conclusions, lessons learned and guidelines derived from these studies, in order to endorse and complete the results of our own investigations.

The case studies can be classified according to two dimensions: the method employed to construct the ontology, and the purpose of the experiment. According to the former we can distinguish among those aiming at building ontologies i) from scratch, ii) by reuse or iii) with the help of (automatic) knowledge acquisition techniques. The objectives followed by the studies are twofold: the majority of the experiments have been carried out to validate a particular methodology or method, or to exemplify the usage of a specific tool for ontology engineering; a considerably lower number of studies applied existing results in ontology engineering as methodological or technological support for creating a specific ontology. This last category of studies gives an account of the impact, the usability and the added value of current ontology engineering research and development in real-world settings.

[2,5,6,14,15,27,29], to name only a few, report on the application of self-developed methodologies and methods to manually build different types of ontologies. The results of these experiments are centered on the (positive) usability of the proposed approach in the designated context and marginally address the question of process operationalization. With this respect each paper emphasizes the need for high quality tools (e.g. for translating, matching or merging, to name the most frequently mentioned ones). Furthermore the authors acknowledge the resource-intensive nature of the

manual application of the proposed approaches, and propose (or highlight the need for) dedicated ontology engineering environments. More details on this type of empirical studies can be found for example in [7].

In the remaining of this section we overview several case studies *applying* existing ontology engineering research and development to real-world settings. Some of these experiments are situated in the originating context of particular methodologies and methods, therefore resorting to these as guidance for the completion of the ontology construction tasks (e.g., the ontologies introduced in [1,13,21,24,30]). The experiments primarily consist of a description of the engineering process followed by superficial observations related to the lack of adequate technological support, at most. By contrast, other experiments evaluate several methodologies and methods w.r.t. their relevance and usability, prior to applying them in a particular application setting, or operate the engineering process without nominally committing to existing techniques [10,12,17,18,22,26,28]. The results of these evaluation procedures reveal the limited usability or the poor impact of the most part of existing ontology engineering methodologies and methods.

Uschold and Healy report on an experiment in which an engineering mathematics ontology is used to detail the specification of a simple software tool and to allow units-conversion and dimensional consistency checking capabilities to this application [28]. In this attempt they tackle some of the most important issues related to ontology engineering processes, from ontology evaluation to more technical activities such as the translation to new representation languages and the integration of multiple ontologies to a new application setting. The case study reported in [28], though not investigating the complete ontology life cycle, reveals several important limitations of ontology-driven research: the difficulties of automatic translation between representation formats and the need for scalable and efficient technologies. Russ and colleagues describe a case study in which an ontology covering the air campaign domain was built by reusing existing ontologies partially covering its context [22]. The conclusions of this case study are comparable to the ones stated by Uschold and Healy: while reusing ontologies was perceived to be beneficial for this particular setting, the authors emphasize the limitations of the techniques available so far, in particular related to language translators and ontology merging. The case study in [17] reports on the feasibility of current technologies in managing and using large scale medical ontologies, emphasizing the need for a more task-oriented approach to ontology engineering at methodological level, and the lack of feasible methods for extracting ontological knowledge from semi-structured models. Paslaru and Mochol point out the limitations of current technologies for translating, comparing and merging ontologies in [18], as resulted from a case study in which a Human Resources ontology was built on the basis of standard eRecruitment classifications. Challenges related to organizational aspects of ontology engineering processes are mentioned in [10,12,26].

Summarizing existing methodologies were applied in a small number of in situ case studies under the direct supervision and with the participation of the methodology developers, at most. The methodologies were only used in the application scenarios they were originally designed for, and little is known about their use for related application areas.

3 Ontology Engineering in a Nutshell

Ontology Engineering (OE) is formally defined as "the set of activities that concern the ontology development process, the ontology life cycle, and the methodologies, tools and languages for building ontologies"[7]. This section summarizes the most important of these activities.

Ontology engineering methodologies support ontology building for centralized ontology applications.[3] [4,20] focus on the consensus building process in collaborative ontology engineering. Methodologies guiding the ontology reuse process, *e.g.*, [6,19] or the ontology learning process, *e.g.*, [16] complete the picture.

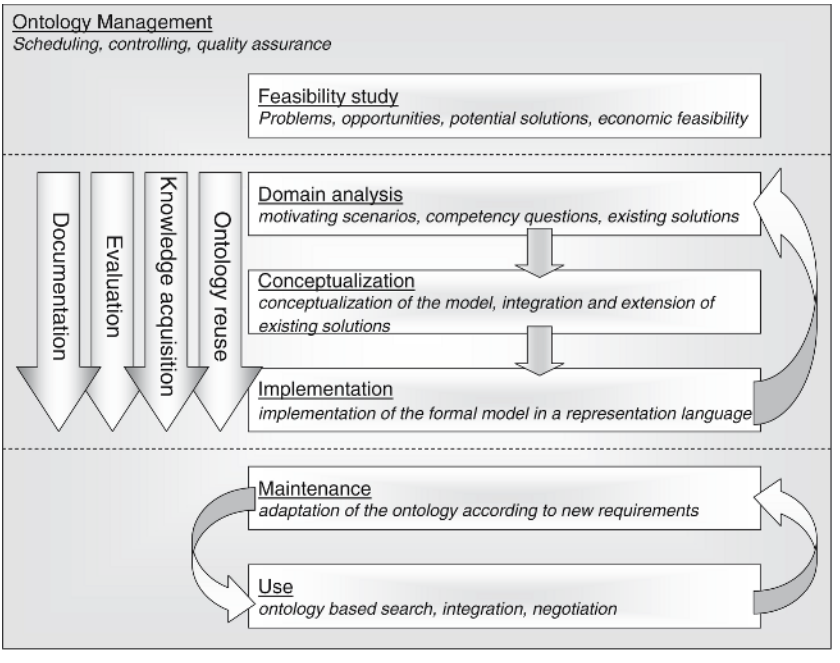


Fig. 1. Ontology Engineering Activities

Methodologies divide the ontology building process in a varying number of stages, and propose a number of activities for each stage. The importance of a particular activity within a methodology primarily depends on, *e.g.*, the characteristics of the ontology-based application, the complexity of the ontology to be built, the availability of information sources, and the experience of the ontology engineers.

[7] differentiates among *management*, *development-oriented* and *support* activities within an ontology engineering process (cf. Fig. 1). The organizational setting of the overall process is covered by so-called ontology management activities. In the pre-development phase the *feasibility study* examines if an ontology-based application or

[3] Refer for example to [7,25] for recent overviews.

the use of an ontology in a given context is the right way to solve the problem at hand. *Domain analysis*, *conceptualization* and *implementation* are classical ontology development activities. The *maintenance* and the *use* of the ontology are post-development activities. Ontology support activities *e.g.*, *knowledge acquisition* (KA), *evaluation*, *reuse*, and *documentation* are performed in parallel to the core development activities.

Methodologies additionally define the roles of the individuals involved in the ontology building process. They primarily differentiate between *domain experts* providing knowledge w.r.t. the domain to be modeled, *ontology engineers* with expertise in fields such as knowledge representation or ontology tools, and *users* applying the ontology for a particular purpose.

4 Our Survey

4.1 Survey Overview

The survey had the objective to capture the basic understanding of semantic technology applicants w.r.t. ontology development, to give an account of current ontology engineering practice, and to identify common problems with available ontology engineering methodologies, methods and tools.

The findings reported in this paper are based on 34 structured interviews conducted within a three months period.[4] After a short tutorial on the utilized ontology engineering terminology, the participants were requested to answer 28 questions related to particular aspects of ontology development. Complementarily to detailed answers to these questions the interviewers collected general comments.

The survey gives a comprehensive assessment of the current state of the art in ontology engineering. Prior to the data collection procedure, the contents, organization and presentation of the survey were evaluated and revised by a group of three academia and industry experts in the area of ontology engineering. Moreover, the respondents are representative for the community of users and developers of semantic technologies. They were IT practitioners, researchers and experts from various disciplines, affiliated to industry or academia, who were involved in the last 3 to 4 years in ontology building projects in areas such as skill management, human resources, medical information systems, legal information systems, multimedia, Web services, and digital libraries. The survey was targeted *exclusively* at technology applicants (as opposed to methodology or tool developers in the Semantic Web area) in order to give a real account on the impact of the results achieved so far beyond their originating context. At the time of the interviews the interviewees possessed an average ontology engineering experience of 1 to 1.5 years. Around 50% were affiliated to industry. Only a small fraction, mostly domain experts, had received ontology engineering training in advance.

The target application the ontologies were built for ranged from proof-of-concept implementations to commercial solutions. Consequently, most of the surveyed ontologies were domain ontologies—either application-dependent or -independent. A single ontology had upper-level character, while 6 were core ontologies. The ontologies had an average size of 1000 ontology entities (concepts, properties, axioms and fixed

[4] The detailed survey results may be obtained from the authors on request.

instances). The development efforts were approximated to 4 person months on the average. 50% of the ontologies were built from scratch. If other ontologies were reused, they made up to 50% of the final ontology.

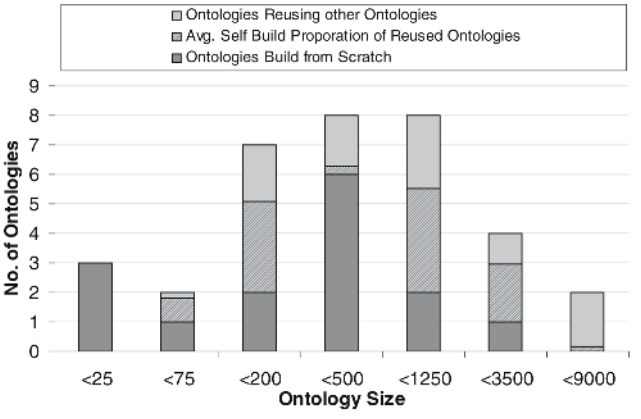


Fig. 2. Size Distribution of Surveyed Ontologies

4.2 Survey Design

The study covered both **open-ended** and **close-ended** questions (cf. Table 1).[5] The former do not impose any constraints on the form or the content of the responses, and are intended to capture general facts about the surveyed ontology engineering projects. By contrast, in the second category the answers of the respondents are limited to a fixed set of responses. Typical examples of close-ended questions are dichotomous (yes/no) questions, multiple choices, as well as scaled (also called ranking) questions using various scale models. In our case we used five point ranking scales to assess the complexity of ontology engineering activities, the quality of the methodological and tool support and the level of experience of the engineering team.

The questionnaire can be divided into 4 categories. 6 **introductory questions** are intended to describe the most important facts about an ontology engineering project: the ontology which was built, its name, size, scope, purpose, as well as the overall development efforts. The second group of 13 questions focused explicitly on particular aspects of the ontology engineering process (**process questions**, cf. Table 1): the domain analysis, the conceptualization, the implementation, the ontology population as well as reuse and knowledge acquisition. For each ranking question the ontology builders rated on a scala from 1 to 5 whether the respective activity was very easy (1) or very difficult (5) to perform in their case. For each rating level and each question, the survey included detailed examples in order to facilitate the ranking and make the results comparable. For instance, in the case of the *conceptualization* the interviewees estimated the score in relation to the complexity of the conceptual model (cf. Figure 3). If reuse was relevant, they were asked to estimate the contingent of the final ontology, which was built

[5] Refer to [23] for a detailed account of questionnaire design principles.

Table 1. Survey Organization

No.	Acronym[6]	Topic	Response
		Introductory questions	
1	ONTNAME	The name of the ontology	open-ended
2	ONTNS	The namespace of the ontology	open-ended
3	SCOPE	Purpose and scope of the ontology	open-ended
4	SIZEO	Total size of the ontology	open-ended
5	TYPE[7]	Ontology type	scaled
6	COSTS	Ontology development effort in person months	open-ended
		Process questions	
7	SIZEB	Percentage of final ontology built from scratch	open-ended
8	DCPLX	Complexity of the domain analysis	scaled
9	CCPLX	Complexity of the ontology conceptualization	scaled
10	ICPLX	Complexity of the ontology implementation	scaled
11	DATA	Complexity of the ontology instantiation	scaled
12	SIZER	Percentage of the final ontology built by reuse	open-ended
13	COMPRH	Complexity of the ontology understandability task	scaled
14	USAB	Complexity of the usability assessment	scaled
15	TRANS	Complexity of translation operations	scaled
16	MOD	Complexity of modification operations	scaled
17	INT	Complexity of merging and integration tasks	scaled
18	DOCU	Complexity of the documentation task	scaled
19	OEVAL	Complexity of the evaluation of the final ontology	scaled
		Organizational questions	
20/21	OCAP/DCAP	Capability of the ontologists/domain experts	scaled
22/23	OEXP/DEXP	Level of experience of the ontologists/domain experts	scaled
24/25	LEXP/TEXP	Level of experience w.r.t. languages and tools	scaled
26	SIZET	Team size	open-ended
27	TURN	Personnel turnover	scaled
		Technological questions	
28	TOOL	Level of technological support for particular activities	scaled

in this way, and to rate the understandability and clarity of the source ontologies, the complexity of the translation between differen representation languages, the difficulties in adapting the reused ontologies to the own needs, or the integration of multiple reused ontologies. The complexity of the evaluation and documentation were further aspects of the questionnaire.

The third part of the survey contained 8 **organizational questions**. The interviewees assessed ranking for the average capability of the ontology building team as domain experts and ontology engineers, respectively. Further questions covered, for instance, the average experience in building ontologies and the available know-how as regarding ontology representation languages and tools. A last part of the survey was concerned with **technological** issues, primarily related to the available tool and methodological

[6] We make use of these notations within the discussion of the survey results.

[7] The scale model for the description of ontology types included 5 elements: upper-level, domain, application, task and core ontologies cf. [8].

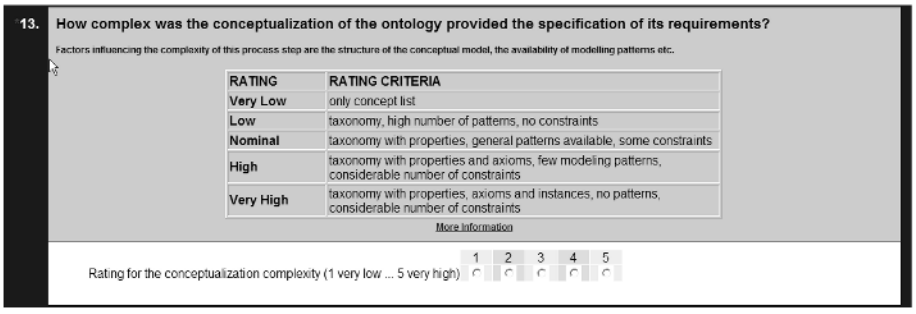


Fig. 3. The Conceptualization Complexity Question

support and its quality. The interviewed persons also commented on the activities which were most difficult to carry out, and on the methods and tools utilized in particular projects.

The survey contributed to a better understanding of the ontology building process in general. Furthermore, the results of the survey provide more detailed information about the effort related to ontology building in the future.

4.3 Survey Results

The presentation of the results of the conducted interviews is divided into four categories: *general issues*, *process issues*, *organizational issues* and *technological issues*.

General Issues. This category of results is not directly related to specific survey questions, but are distilled from the plethora of general concerns and comments expressed by the interviewees during the operation of the study.

The survey clearly pointed out that the popularity of ontologies—as a means to solve many non-trivial IT problems—is not equally shared by the discipline of ontology engineering. While the reasons for this situation are traceable in the case of the interviewed domain experts, many of the IT professionals or researchers seemed not to perceive ontology building as a systematic process which should be performed according to a pre-defined methodology. Nevertheless, the way ontology development has been carried out was on the whole compatible with the recommendations made by methodologies available in the field (see below). A more serious issue is related to the lack of terminological knowledge and to the controversial understanding of the participants w.r.t. core concepts of ontology engineering. Starting from the oft-enunciated fuzzy definition of ontologies, this confusion is further propagated to activities involving the usage of external knowledge sources to aid the manual ontology development. The survey showed that the majority of the participants associate ontologies with almost every type of lightweight conceptual structure, and that they have difficulties in distinguishing between tasks such as ontology reuse, ontology learning and, more generally, knowledge acquisition. Further on, the notion of reuse was often associated with informative

materials consulted by domain experts while constructing the ontology. A last terminological weak-point is constituted by the basic understanding of IT experts, who did not distinguish between a conceptual and an implementation level of an ontology. Terms such as "conceptualization" and "formalization" seemed not to be prevalent in current ontology engineering practice.

Table 2. Summary of General Issues

Result
80% of the projects did not follow a particular ontology engineering methodology
90% of the projects followed implicitly established ontology engineering activities
80% of the participants expressed the need for a terminological clarification
very broad understanding of the term *ontology* in 100% of the cases
ontology reuse interpreted as usage of arbitrary information sources in 90% of the relevant cases

Process Issues. As aforementioned the structure of the survey assumed a "classical" breakdown of ontology engineering processes at the level of activities introduced in Section 3. This set-up proved to match to a satisfactory extent to the way the survey participants carried out the process. The interviews emphasized however some discrepancies between i) the complexity and significance of particular process stages as perceived by ontology engineering practitioners, and ii) the attention these process stages received in the research community so far. This applies in particular to the following issues:

Domain analysis: All participants emphasized the resource-intensive nature of this process step and the lack of low entry barrier methods and tools to support the knowledge elicitation task. Moreover, in projects building highly specialized ontologies such as for the legal or the medical domain, ontology engineers—who were responsible for guiding the rest of the team during this phase—manifested their concern w.r.t. their ability to accomplish this task appropriately. Consequently the interviewed engineers commonly agreed on the (partial) arbitrariness of the knowledge elicitation procedure in the questioned projects. Methodologies tend to handle these particular issues at a very generic level and were thus not utilized for the domain analysis, excerpt in form of competency questions in individual cases. Further on, the interviews revealed the difficulties encountered by ontology developers in combining multiple strategies for building fragments of the same ontology. Some of them were puzzled about the means to choose among alternative strategies and the way and the time point partial engineering results emerging from these complementary activities should be integrated.

Conceptualization and implementation: The majority of the interviewees did not perceive a clear cut between the conceptualization and the implementation steps as necessary. After a lightweight description and classification of the expected outcomes the engineering team implemented the ontology with the help of a common ontology editor.

This task was primarily performed by domain experts, who did not report any particular difficulties in getting familiar with or utilizing simple ontology editors.[8]

Ontology reuse: In over 50% of the cases the final ontology was built with the help of other ontological sources. In 11 of the reuse-relevant projects the engineering team pointed out the considerable resources invested in understanding the reused sources and assessed high complexity ratings to the task of translation. By contrast, issues like merging and integration were not relevant to the investigated scenarios or were assigned lower complexity scores. This situation might, however, be caused by the fact that the average number of reused ontologies per project was very low (maximal 2). The modification of the ontologies to be reused was performed in two stages: first the relevant fragments were extracted from the sources, then they were customized and integrated to the target ontology. The effort required to perform these activities was often estimated to a nominal value.

Ontology evaluation: All domain experts manifested concerns w.r.t. the quality of the implemented ontology. They claimed for usable methods guiding them within this task. This holds true for both ontology engineering experts, who were not able to appropriately assist this endeavor, but also for the non-IT survey participants, and caused acceptance problems. In over 40% of the cases the results of the process were not nominally evaluated. In the remaining ones, 95% have been manually evaluated by their authors without the help of a methodical approach. 3 ontologies have been evaluated through external expert judgement. A further result of the study was the necessity of an incremental approach to ontology evaluation. Ontology developers were not sure about the most adequate point in time for performing first evaluation tests, and pointed out the implications of this issue for evolving ontologies or for long-term projects, in which the application using the created ontology is not timely available for preliminary evaluations. In this context they spoken out the question of how to feasibly determine the real start and the end points of an ontology engineering project, this being a prerequisite for any controlling and planing activities.

Ontology population: The question related to the complexity of the ontology population task addressed two aspects: the data sources used as input to create the ontology instances and the required mapping between the input scheme and the target ontological model. Both were estimated with the highest scores among all questions on ontology development (between 3 and 4), comparable to the ones associated to the domain analysis. In most of the cases the ontology population task was exercised on semi-structured data expressed in natural language and required complex mappings covering concepts and properties.

Ontology maintenance: This task was not properly represented in any of the analyzed projects. Some ontology engineers raised, however, the issue of clearly determining the transition point from ontology development to maintenance or evolution. In correlation with the fuzzy nature of the evaluation task, ontology engineers were not in the position to distinguish between the two phases.

[8] This situation might correlate with the fact that of the ontologies had a relatively simple structure, i.e. taxonomies augmented by properties between concepts.

Documentation: The majority of the projects reported above average documentation efforts, and expressed the need for an automatization of this task.

The survey pointed out that only a small percentage of ontology-related projects follow a systematic approach to ontology building, and even less commit to a specific methodology. Most of the projects are executed in an ad-hoc manner. In the early project phases ontology builders underestimate the efforts underlying this endeavor and the importance of a methodological framework. It is the early project phase, however, where methodologies could offer most gains, as our analysis illustrates. In later stages of the process this perception changes; however, the a posteriori adoption of a certain development strategy is complicated by the lack of methods to evaluate the suitability and the general quality of existing methodologies.

The activities carried out in the examined case studies are covered by at least one methodology. However, the provided guidelines are insufficient for the most challenging process stages and no methodology handles the complete range of activities registered in our survey. Furthermore, different real-world scenarios require customizable method assemblies, rather than pre-defined rigid workflows as proposed by current methodologies.

Table 3. Summary of Process Issues

Question acronym	Average result[9]	Comments
DCPLX	3.3	lack of fine-grained guidance for the domain analysis resource-intensive activity challenging if the ontology integrates knowledge from diff. domains
CCPLX	3	widespread use of ontology editors no clear distinction between conceptualization and implementation
ICPLX	3.5	use of existing structured data sources to populate ontology extraction of instances from text
SIZER	50%	max. 2 ontologies reused for a new ontology 50% of the final ontology built by reuse
TRANS	3.5	high complexity of language translation
MOD	3.5	mainly partial reuse involving customization
OEVAL	3	need for clear guidelines for ontology evaluation very complex activity, rarely performed
DOCU	3.5	documentation is a time consuming activity dedicated tool support required

Additional comments
lack of guidelines for the use of ontology learning algorithms
lack of guidelines to combine different knowledge acquisition techniques
no methodology covers all relevant ontology engineering activities
need for methodological support seldom perceived at project start

Organizational Issues. The surveyed case studies contained on the average relatively small teams (two ontology engineers and two domain experts), which regularly

[9] We used the following five point scale 1:very low, 2:low, 3:nominal, 4:high, 5:very high.

Table 4. Summary of Organizational Issues

Question acronym	Average result
SIZET	4
OEXP/DEXP	1.25 years
TURN	15% personal turnover
Additional comments	
definition of common terminology in teams often difficult process support for consensus making required	

organized F2F meetings to help on the ontology development. The personnel turnover was very low, this being also related to the relatively short project duration. In this context we can not make any reliable statement on the effects of this factor in long-standing projects. Nevertheless, in the projects in which domain experts did not possess IT competency the participants reported communication and comprehension problems. Prior to specifying the ontology, considerable efforts were invested in agreeing on a similar understanding upon the domain of interest. This issue had consequences on the way the engineering process was performed. However, we did not record major problems w.r.t. the achievement of a shared ontology, though the importance of methods assisting the consensus making process was acknowledged by many interviewees.

Technological Issues. As stated in other similar investigations in the past, tools are a crucial factor for an efficient ontology engineering practice. As illustrated in Figure 4, phases such as the conceptualization and the implementation received satisfactory ratings, while the evaluation and the domain analysis are de facto manual tasks.

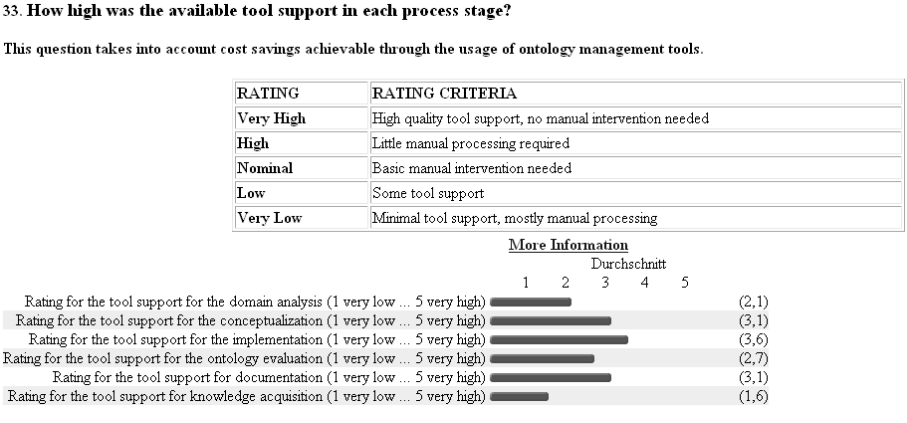


Fig. 4. The Results of the Tool Support Question

Besides the well known absence of computer-aided ontology evaluation support, our survey pointed out the need for dedicated conceptualization tools. All investigated projects used ontology editors for this purpose. Their functionality was positively evaluated by many of the participants. Further on, the answers repeatedly referred to the need

Table 5. Summary of Technological Issues

Question acronym	Average result	Comments
TOOL: domain analysis	2,1	technological support limited to text analysis tools
TOOL: conceptualization	3,1	satisfactory support for conceptualization conceptualization supported by text editors, mind maps and OE environments
TOOL: implementation	3,6	satisfactory results for OE environments no lightweight ontology engineering environments available no tool supports the easy translation of an ontology to different natural languages
TOOL: evaluation	2,7	ontology editors used for technical evaluation of ontologies semantic ontology evaluation not tool supported
TOOL: documentation	3,1	text editors no specialized ontology documentation tools available
TOOL: knowledge acquisition	1,6	existing KA tools not used or not helpful in most cases no tools to leverage ontologies from existing data sources

for tools for translating between representation languages, including methods to convert semi-structured conceptual structures to OWL and RDF(S). The notion of lightweight technological support was mentioned in relation to every ontology engineering activity: apart from the challenging research questions approached in the last fifteen years, the community of ontology engineering practitioners require simple means to extract ontological structures from existing knowledge sources, to translate concept labels to other natural languages and to ease the creation of documentation.

5 Discussions of the Results

The survey highlighted several weak-points of existing ontology engineering methodologies from an empirical point of view. In this section we introduce research directions which explicitly address the problems the community of ontology engineering practitioners is facing, and sketch a possible solution for the alleviation of the present state of the art.

General Issues. Despite the long history and the multitude of research initiatives in the area of ontology engineering methodologies, the visibility of the achieved results remains restricted to a small community of experts affiliated to academia. Methodologists should thus invest more efforts in the dissemination of their results to a wider audience and in promoting the achievable returns of a methodologically supported ontology building approach. Case study reports comparing ontology building efforts with and without methodological support may be one way to demonstrate the efficiency gains. Further on there is a need for support in selecting methodologies which suit certain application settings. The advantages and disadvantages of manual, semi-automatic or

reuse-oriented ontology creation for a specific use case are not obvious for potential ontology builders, even when they have a strong IT background. A methodology selection framework should include application-oriented decision criteria to support this activity. Moreover, the methodology itself could include a step in which the engineers pick from a list of available methods the ones suitable for a particular task and build up their own process model. Template process models covering standard requirements could be made available. This is an alternative to the current trend to create new methodologies for emerging application scenarios. Once the workflow has been specified, the developers need requirements engineering support techniques. This includes, for instance, means to determine the optimal formality level for the prospected ontology or to analyze the trade-off between development effort and size/complexity of the conceptual model.

Process Issues. Knowledge elicitation is one of the most time consuming tasks in the process of building an ontology. Current methodologies lack a comprehensive guidance for this task and instruments to choose among knowledge elicitation techniques. With the introduction of selection metrics this problem may be alleviate. A comprehensive methodology should allow for modular and customizable process models in which ontology builders combine different methods for the stages of the ontology development process they require. Ontology builders may leave out process descriptions for activities

Table 6. Summary of Recommendations

No.	Recommendation
	General recommendations
1	enforce dissemination, *e.g.* publish more best practices
2	define selection criteria for methodologies
3	define a unified methodology following a method engineering approach
4	support decision for the appropriate formality level given a specific use case
	Process recommendations
5	define selection criteria for different KA techniques
6	introduce process description for the application of different KA techniques
7	improve documentation of existing ontologies
8	improve ontology location facilities
9	build robust translators between formalisms
10	build modular ontologies
11	define metrics for ontology evaluation
12	offer user oriented process descriptions for ontology evaluation
	Organizational recommendations
13	provide ontology engineering activity descriptions using domain-specific terminology
14	improve consensus making process support
	Technological recommendations
15	provide tools to extract ontologies from structured data sources
16	build light-weight ontology engineering environments
17	improve the quality of tools for domain analysis, ontology evaluation, documentation
18	include methodological support in ontology editors
19	build tools supporting collaborative ontology engineering

which are automated by tools. Moreover, methodologies should provide more support on the evaluation of ontologies beyond completeness and soundness. In commercial applications trade-offs between additional modeling and costs are balanced. This requires an evaluation w.r.t. the achievable gains of modeling additional ontology entities.

Organizational Issues. Ontology builders had no major organizational problems. For collaborative ontology engineering in distributed environments with participants originating from different domains, methodologies should offer support on the consensus finding process. The reformulation of existing method descriptions in a less computer science-oriented way, using domain-specific terminology, may facilitate the comprehensibility of the engineering process. Positive experiences in the biology domain attest the value of such domain-close descriptions.

Technological Issues. From a technological point of view several ontology management tools (e.g. editors or reasoners) have reached a feasible maturity level. Nevertheless many ontology engineering activities are not supported adequately at technical level. Available tools in this context originate from academic research projects, and focus on solving non-trivial generic research questions instead of operationalizing simple tasks whose automatization is clearly not problematic. Further on ontology builders require easy-to-use ontology engineering environments which are extensible to support different kinds of ontology engineering processes. Current editors do not support discussion-based ontology engineering, in which a number of ontology builders first argue about modeling decisions before they decide on them. The provision of ontology engineering patterns and high-quality ontologies for commonly used domains may also ease the effort to build ontologies.

In summary, this survey demonstrates the limited impact of methodologies in real-world ontology-related projects. However, the results also evidence that ontology engineering research has reached a level of development with a basic inventory of methods and tools which, if properly utilized, considerably ease the work of ontology practitioners. Therefore, a first conclusion of the survey is the need for initiatives aiming at promoting these results to a wider audience and consolidating them for an increased usability. European projects such as Sekt-Semantically-Enabled Knowledge Technologies are taking first steps in this direction. Complementarily, the experiences gained during this survey let us assume that a *method engineering* approach to the area of ontologies could be a viable alternative to the creation of new methodologies and methods. Software engineering is already moving in this direction, *cf.* [9]. They describe software engineering activities according to predefined templates. Templates include input and output factors, available methods to support the activity, required preceding activities and possible succeeding activities among other things. Additional to this new methodological approach practitioners require evaluation methods to compare the trade-offs between investing in additional modeling and gained functionality.

6 Summary and Outlook

Despite the growing popularity of ontologies as a knowledge representation formalism used on the Web, very little is known about the engineering process underlying their

construction in practice. The literature reports predominately on case studies which involved methodologists, while ontologies are envisioned to be built by domain experts possessing limited to no professional skills in ontology engineering. We alleviate this information gap with a survey interviewing over 30 practitioners who developed ontologies for commercial, as well as academic applications for a wide range of domains. Although this number is not large w.r.t. the number of ontologies available on the Web it is the largest study conducted so far; and we continue to collect ontology engineering experiences.

The survey investigated the systematics, the invested effort and the general problems encountered in building these ontologies. The main findings are i) practitioners do not follow any particular ontology engineering methodology, though there is some overlapping ii) they require selection support to choose from manual, semi-automatic or reuse oriented engineering approaches, iii) they need cost benefit analysis methods to determine the transition point between ontology engineering activities and iv) existing ontology management tools have reached a feasible level of functionality to be useful.

In order to overcome some of the problems revealed in this survey we suggest to increase the efforts invested in promoting the advantages of methodological ontology engineering to a wider audience. Furthermore, we propose to establish a unified methodology which supports domain experts to customize ontology engineering process models according to their application scenario. This requires the creation of compliant method components or the adjustment of existing ones to enable their joint utilization. Evaluation metrics for ontologies need to be defined in order to assess the usability of similarly scoped methods in particular circumstances. We will continue our research in these directions.

Acknowledgements. This work has been partially supported by the European Network of Excellence "KnowledgeWeb-Realizing the Semantic Web" (FP6-507482), as part of the KnowledgeWeb researcher exchange program **T-REX**, and by the European project "Sekt-Semantically-Enabled Knowledge Technologies"(EU IST IP 2003-506826). We thank all interviewees for the valuable input without which this paper could not have been produced. We also want to thank York Sure (University of Karlsruhe) and Richard Benjamins (iSOCO) for contributing to the realization of the final version of the survey.

References

1. V. R. Benjamins, D. Fensel, S. Decker, and A. Gómez-Pérez. $(KA)^2$: Building Ontologies for the Internet. *International Journal of Human-Computer Studies*, 51(1):687–712, 1999.
2. A. Bernaras, I. Laresgoiti, and J. Corera. Building and Reusing Ontologies for Electrical Network Applications. In *European Confernce on Artificial Intelligence (ECAI'96)*, 1996.
3. O. Corcho, M. Fernández-López, and A. Gómez-Pérez. Methodologies, tools and languages for building ontologies: where is their meeting point? *Data & Knowledge Engineering*, 46(1):41–64, 2003.
4. J. Euzenat. Building Consensual Knowledge Bases: Context and Architecture. In *Proc. of the 2nd Int. Conference on Building and Sharing Very Large-Scale Knowledge Bases (KBKS)*, pages 143–155, Enschede the Netherlands, 1995.

5. M. Fernández-López, A. Gómez-Pérez, J. P. Sierra, and A. P. Sierra. Building a Chemical Ontology Using Methontology and the Ontology Design Environment. *Intelligent Systems*, 14(1), January/February 1999.
6. A. Gangemi, D.M. Pisanelli, and G. Steve. Ontology Integration: Experiences with Medical Terminologies. In *Formal Ontology in Information Systems*. IOS Press, 1998.
7. A. Gómez-Pérez, M. Fernández-López, and O. Corcho. *Ontological Engineering*. Springer, 2003.
8. N. Guarino. Formal Ontology and Information Systems. In *Proc. of the 1st Int. Conf. on Formal Ontologies in Information Systems (FOIS)*, 1998.
9. B. Henderson-Sellers. Method engineering for OO system development. *Commmunications of the ACM*, 46(10):73–78, 2003.
10. M. Hristozova and L. Sterling. Experiences with Ontology Development for Value-Added Publishing. In *Proceedings of the International Workshop on Ontologies in Agent Systems OAS03 co-located with the AAMAS*, 2003.
11. D. Jones, T. Bench-Capon, and P. Visser. Methodologies for Ontology Development. In *Proc of the IT&KNOWS Conference of the 15th IFIP World Computer Congress*, 1998.
12. N. J. J. P. Koenderink, J. L. Top, and L. J. van Vliet. Expert-Based Ontology Construction: A Case-Study in Horticulture. In *Proceedings of the International Database and Expert Systems Applications Workshops DEXA05*, 2005.
13. I. Laresgoiti, A. Anjewierden, A. Bernaras, J. Corera, Schreiber A. T., and B. J. Wielinga. Ontologies as vehicles for reuse: a mini-experiment. In *Proceedings of the 10th Banff Knowledge Acquisition for Knowledge-Based Systems Workshop K-CAP96*, pages 1–21, 1996.
14. T. Lau and Y. Sure. Introducing Ontology-based Skills Management at a large Insurance Company. In *Modellierung 2002, Modellierung in der Praxis - Modellierung für die Praxis, Tutzing, Deutschland, 25.-27. März 2002*, pages 123–134, 2002.
15. D. B. Lenat and R. V. Guha. *Building large knowledge-based systems. Representation and inference in the CYC project*. Addison-Wesley, Reading, Massachusetts, 1990.
16. A. Maedche. *Ontology Learning for the Semantic Web*. Kluwer Academics, February 2002.
17. E. Paslaru Bontas. Practical Experiences in Building Ontology-Based Retrieval Systems. In *Proceedings of the 1st International ISWC Workshop on Semantic Web Case Studies and Best Practices for eBusiness SWCASE05*, 2005.
18. E. Paslaru Bontas, M. Mochol, and R. Tolksdorf. Case Studies in Ontology Reuse. In *Proc. of the 5th International Conference on Knowledge Management IKNOW05*, 2005.
19. H. S. Pinto and J. Martins. Reusing ontologies. In *AAAI 2000 Spring Symposium on Bringing Knowledge to Business Processes*, pages 77–84, 2000.
20. H. S. Pinto, C. Tempich, and S. Staab. DILIGENT: Towards a fine-grained methodology for DIstributed, Loosely-controlled and evolvInG Engingeering of oNTologies. In *Proc. of the 16th European Conference on Artificial Intelligence (ECAI 2004)*, pages 393–397, 2004.
21. Dip European Project. Financial Ontology (Deliverable 10.3 DIP FP6 - 507483). http://dip.semanticweb.org/documents/D10.3.pdf, 2005.
22. T. Russ, A. Valente, R. MacGregor, and W. Swartout. Practical Experiences in Trading Off Ontology Usability and Reusability. In *Proc. of the Knowledge Acquisition Workshop KAW99*, 1999.
23. Seymour Sudman and Norman M. Bradburn. *Asking Questions : A Practical Guide to Questionnaire Design*. Jossey-Bass, 1st edition, 1982.
24. Y. Sure, S. Bloehdorn, P. Haase, J. Hartmann, and D. Oberle. The SWRC Ontology - Semantic Web for Research Communities. In *Proc. of the 12th Portuguese Conference on Artificial Intelligence - Progress in Artificial Intelligence (EPIA 2005)*, pages 218 – 231, 2005.
25. Y. Sure, C. Tempich, and D. Vrandecic. Ontology Engineering Methodologies. In *Semantic Web Technologies: Trends and Research in Ontology-based Systems*. Wiley, UK, 2006.

26. C. Tautz and K. D. Althoff. A Case Study on Engineering Ontologies and Related Processes for Sharing Software Engineering Experience. In *Proc. of the International Conference on Software Engineering and Knowledge Engineering SEKE00*, 2000.
27. C. Tempich, H. S. Pinto, and S. Staab. Ontology Engineering Revisited: an Iterative Case Study with DILIGENT. In *Proc. of the 3rd European Semantic Web Conference (ESWC 2006)*, pages 110–124, 2006.
28. M. Uschold, M. Healy, K. Williamson, P. Clark, and S. Woods. Ontology Reuse and Application. In *Int. Conference on Formal Ontology and Information Systems FOIS98*, 1998.
29. M. Uschold, M. King, S. Moralee, and Y. Zorgios. The Enterprise Ontology. *Knowledge Engineering Review*, 13(1):31–89, 1998.
30. G. Zhao, J. Kingston, K. Kerremans, F. Coppens, R. Verlinden, R. Temmerman, and R. Meersman. Towards an Ontology of Forensics covering Financial Securities Fraud. In *Proceedings of the International Workshop on Regulatory Ontologies WORM04 co-located with the OTM Conferences*, 2004.

Conceptual Design for Domain and Task Specific Ontology-Based Linguistic Resources*

Antonio Vaquero[1], Fernando Sáenz[1], Francisco Alvarez[2], and Manuel de Buenaga[3]

[1] Universidad Complutense de Madrid, Facultad de Informática, Departamento de Sistemas Informáticos y Programación, C/ Prof. José García Santesmases, s/n, E-28040, Madrid, Spain
[2] Universidad Autónoma de Sinaloa, Ángel Flores y Riva Palacios, s/n, C.P 80000, Culiacán, Sinaloa, México
[3] Universidad Europea de Madrid, Departamento de Sistemas Informáticos, 28670 Villaviciosa de Odón. Madrid, Spain
{vaquero, fernan}@sip.ucm.es, fjalvare@fdi.ucm.es, buenaga@uem.es

Abstract. Regardless of the knowledge representation schema chosen to implement a linguistic resource, conceptual design is an important step in its development. However, it is normally put aside by developing efforts as they focus on content, implementation and time-saving issues rather than on the software engineering aspects of the construction of linguistic resources. Based on an analysis of common problems found in linguistic resources, we present a reusable conceptual model which incorporates elements that give ontology developers the possibility to establish formal semantic descriptions for concepts and relations, and thus avoiding the aforementioned common problems. The model represents a step forward in our efforts to define a complete methodology for the design and implementation of ontology-based linguistic resources using relational databases and a sound software engineering approach for knowledge representation.

1 Introduction

Existing linguistic resources (LR) can be used as a source of knowledge by any natural language processing application. However, most of these LR were developed focusing on coverage and implementation issues rather than on questions of design. This approach to LR construction has yield LR with huge quantities of information but poorly structured. A situation that severely limits their reuse and integration (with other LR), and has proven to be a major obstacle to obtain better results.

We claim that design issues are an important part of the construction of a LR, because in order to develop, reuse and integrate diverse available LR, into a common

* The research described in this paper has been partially supported by the Spanish Ministry of Education and Science and the European Union from the European Regional Development Fund (ERDF) - (TIN2005-08988-C02-01 and TIN2005-08988-C02-02).

R. Meersman, Z. Tari et al. (Eds.): OTM 2006, LNCS 4275, pp. 855 – 862, 2006.

information system (perhaps distributed), requires compatible software architectures and sound data management from the different databases to be integrated. Hence, under this view, a LR must be carefully designed before any implementation is made, by following a software engineering approach.

With that in mind, we have developed a methodology based on relational databases (RDB) and software engineering principles, for the design and implementation of ontology-based LR [1]. The methodology already proposes a conceptual model (an E-R schema). However, the model has to be modified if it is to be used to create structurally sound LR. In this paper, we present an upgrade of this conceptual model as a refinement of the representational power of our previous model. Nevertheless, we only present here the first stage of the classical RDB design (i.e., conceptual design) and only for the ontology side of the model. Thus, leaving the other two stages (i.e., logical and physical) and the lexical side of the model for future papers.

The rest of the paper is organized as follows. In section 2, we point out the importance of conceptual design in the construction of linguistic RDB, and explain our decision to use an ontological model for knowledge representation. In section 3, some common problems of LR are summarized, and the need to develop application-oriented LR is signaled. In section 4, the methodological gaps of past developing efforts that used RDB are underlined. In section 5, we depict a set of ideas intended to help developers to formally specify and clarify the meaning of concepts and relations in more detail. In section 6, a conceptual model that integrates the aforementioned ideas is introduced and described. Finally, in section 7 some conclusions are outlined.

2 Conceptual Design of LR Using RDB

RDB have various drawbacks when compared to newer data models (e.g., the object-oriented model): a) Impossibility of representing knowledge in form of rules; b) Inexistence of property inheritance mechanisms; and c) Lack of expressive power to represent hierarchies. However, as shown in [2, 3, 4] a careful design (i.e., conceptual modeling) can overcome these drawbacks, and let us take advantage of all the benefits of using RDB technology to design and implement linguistic databases [1, 2, 3]. In addition, and following [4, 5, 6, 7], we believe that the computationally proven ontological model with two separated but linked and concurrently developed levels of representation (i.e. the conceptual-semantic level and the lexical-semantic level) is our best choice for linguistic knowledge representation.

3 Some Common Problems in LR

It is relatively easy to create a conceptual model of linguistic knowledge. As seen in the previous section, this has already been done. However, existing LR (ontology-based or not) are plagued with flaws that severely limit their reuse and negatively impact the quality of results. Thus, it is fundamental to identify these flaws in order to

avoid past and present mistakes and create a sound conceptual model that leads to a LR where these errors can be avoided.

Most of the problems of past and present LR have to do with their taxonomic structure. For example, once a hierarchy is obtained from a Machine-Readable Dictionary (MRD), it is noticed that it contains circular definitions yielding hierarchies containing loops, which are not usable in knowledge bases (KB), and ruptures in knowledge representation (e.g., a utensil is a container) that lead to wrong inferences [8]. WordNet and Mikrokosmos have also well-known problems in their taxonomic structure due to the overload of the is-a relation [9, 10]. In addition, Mikrokosmos represents semantic relations as nodes of the ontology. This entails that such representation approach where relations are embedded as nodes of the ontology is prone to suffer the same is-a overloading problems described in [9, 10], as well as the multiple inheritance ones. In the biomedical domain, the UMLS has circularities in the structure of its Metathesaurus [11], because of its omnivorous policy for integrating hierarchies from diverse controlled medical vocabularies. Some of the consequences of these flaws, as well as additional ones have been extensively documented in [5, 6, 9, 12, 13, 14, 15, 16] for these and other main LR.

3.1 Application-Oriented LR

We have come a long way from the days of MRD. However, still today, the focus is on coverage and time-saving issues, rather than on semantic cleanness and application usefulness. Proof of this are the current different merging efforts aimed at producing wide-coverage general LR [16, 17], and the ones aimed at (semi)automatically constructing them from texts [18, 19]. However, no amount of broad coverage will help to raise the quality of output, if the coverage is prone to error [6]. We should have learned by now, that there are no short cuts, and that most experiments aimed at saving time (e.g., automatically merging LR that cover the same domains, or applying resources to NLP that are not built for it, like machine-readable dictionaries and psycholinguistic-oriented word nets) are of limited practical value [20]. Furthermore, in the current trend of LR development, issues such as how to design LR are apparently less urgent, and this is haphazard. More attention must be paid on how LR are designed and developed, rather than what LR are produced.

The experience gained from past and present efforts clearly points out that a different direction must be taken. As [13] pointed out back in the days of MRD: "rather than aiming to produce near universal LR, developers must produce application-specific LR, on a case by case basis". In addition, we claim that these LR must be carefully conceived and designed in a systematic way, according to the principles of a software engineering methodology. This is especially true if RDB are to be used as a knowledge representation schema for LR.

4 Methodological Gaps in the Development of LR Using RDB

Since we are interested in the development of LR using RDB, it is worth mentioning that all the cited efforts in section 2, although they produced useful resources, they

forgot about the methodological nature of RDB. They all stopped at the conceptual design phase and then presented the interface(s) of their respective resources. Thus, there is not a complete description of the entities, relationships and constraints involved in the conceptual and logical design of the DB.

The methodology we propose in [1] encompasses all of the database design stages. Nonetheless, the conceptual model from which it departs has several problems with respect to ontology representation; mainly, its does not foresee any control and verification mechanism for clarifying the semantics relations, a problem that as seen in section 3 is of main concern.

Therefore, if we are to design an ontology-based LR using RDB, our conceptual model must take also into account the semantic relations issue. Thus, as a first step, we enhance the conceptual model presented in [1] as shown in the next section.

5 Refining the Semantics of Concepts and Relations

In order to give our first step towards the enhancement of the conceptual model, we need to clearly state what are the elements that will be abstracted and represented in our upgraded conceptual model, that will help us to: a) build application-oriented LR (as pointed out in section 3.1); and b) avoid the problems present in existing LR as described in section 3.

These elements are concepts, properties of concepts, relations, and algebraic and intrinsic properties of relations. They will help an ontology developer to specify for concepts and relations formal and informal semantics that clarify the intended meaning of both entities in order to avoid the problems discussed in section 3. Informal semantics are the textual definitions for both concepts and relations, as opposed to formal semantics that are represented by the properties of concepts and relations.

However, the fact that these elements will be part of the enhanced conceptual model does not imply that they are an imposition but rather a possibility, a recommendation that is given to each ontology developer.

In the following, we detail the elements surrounding the basic element of our model: concepts.

5.1 Properties of Concepts

These are formal semantic specifications of those aspects that are of interest to the ontology developer. In particular, these specifications may be the metaproperties of [10] (e.g., R, I, etc.). In our application-oriented approach to LR development, only the properties needed for a concrete application domain should be represented. These properties play an important role in the control of relations as it will be seen later.

5.2 Relations

Instead of relations with an unclear meaning (e.g. subsumption), we propose the use of relations with well-defined semantics, up to the granularity needed by the ontology

developer. Moreover, we refuse to embed relations as nodes of the ontology (because of the problems commented in section 3) or to implicitly represent any relation as it is done in Mikrokosmos with the is-a relation. This represents a novelty and an improvement when compared to similar design and implementation efforts as [4] based on ontological semantics and RDB. In the next two subsections, we will describe the elements that help clarifying the semantics of relations.

5.3 Algebraic Properties of Relations

The meaning of each relation between two concepts must be established, supported by a set of algebraic properties from which, formal definitions could be obtained (e.g., transitivity, asymmetry, reflexivity, etc.). This will allow reasoning applications to automatically derive information from the resource, or detect errors in the ontology [21]. Moreover, the definitions and algebraic properties will ensure that the corresponding and probably general-purpose relational expressions are used in a uniform way [21]. Tables 1 and 2 (taken from [21]) show a set of relations with their definitions and algebraic properties.

Table 1. Definitions and Examples of Relations

Relations	Definitions	Examples
C is-a C_1	Every *C* at any time is at the same time a C_1	*myelin is-a lipoprotein*
C part-of C_1	Every *C* at any time is part of some C_1 at the same time	*nucleoplasm part-of nucleus*

Table 2. Algebraic Properties of Some Relations

Relations	**Transitive**	**Symmetric**	**Reflexive**
Is-a	+	-	+
part-of	+	-	+

5.4 Intrinsic Properties of Relations

How do we assess, for a given domain, if a specific relation can exist between two concepts? The definitions and algebraic properties of relations, although useful are not enough. As [9, 10] point out, we need something more. Thus, for each relation, there must be a set of properties that both a child and its parent concept must fulfill for a specific relation to exist between them. We call these properties, intrinsic properties of relations. For instance, in [10] the authors give several examples (according to their methodology) of the properties that two concepts must have so that between them there can be an is-a relation.

6 Designing the Conceptual-Semantic Level for a LR

In this section, we present a conceptual model (an E/R scheme upgraded from our model in [1]) for the conceptual-semantic level of an ontology-based LR as a result of the first design phase, where all the ideas described in section 5 have been incorporated. However, as it was previously established in the introduction, the model will reflect only the ontology part of the LR.

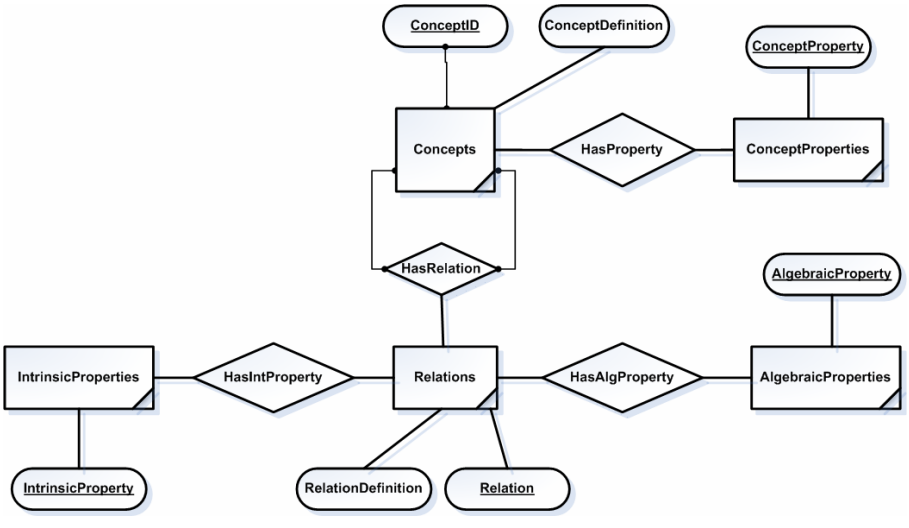


Fig. 1. Conceptual Model for an Ontology-Based LR

The entity set Concepts denotes the meaning of words, and it has two attributes: ConceptID (artificial attribute intended only for entity identification), and ConceptDefinition, intended for the textual definition of the meaning (informal semantics). The entity set ConceptProperties represents the set of formal properties described in section 5.1, and it has one attribute: ConceptProperty used to represent each property.

The entity set Relations represents the set of relations that can exist in an ontology, and it has two attributes: Relation that captures the textual name of each relation (e.g., is-a, part-of, etc.), and RelationDefinition for the textual definition of relations (informal semantics) as illustrated in table 1.

The entity set AlgebraicProperties represents the properties of relations (formal semantics) as seen in table 2, and it has one attribute: AlgebraicProperty that denotes each algebraic property. The entity set IntrinsicProperties conveys the set of properties mentioned in section 5.4 and has one attribute: IntrinsicProperty which represents each intrinsic property.

The relationship set HasProperty is used to assign properties to concepts. The ternary relationship set HasRelation is used to represent that two concepts in an ontology can be linked by a given relation. The relationship set HasAlgProperty is

used to convey that relations could have attached a set of algebraic properties; the same applies for the relationship set HasIntProperty, but for intrinsic properties.

7 Conclusions

We have pointed out that an important and normally put aside step in the development of linguistic databases is the conceptual modeling or conceptual design step. With that in mind, we have used a semantic data model (i.e., the E-R model) to create a conceptual model (departing from the one in [1]), which accounts for a set of ideas that could help developers to create domain and task specific ontology-based LR, where the use of semantic relationships can be controlled. Although we have selected RDB to represent lexical and conceptual knowledge, the model is totally independent of any knowledge representation schema (i.e., databases or knowledge bases). In this paper, we have focused on the ontology side of the model; however, the lexical side of our previous model (see [1]) also needs to be upgraded as it is quite limited. Thus, we are considering the integration of the E-R model for the lexical side of an ontology-based LR proposed and described in [2].

Moreover, a thing that must be clearly understood is that our efforts lean towards the establishment of a software engineering methodology for the design and implementation of ontology-based LR using RDB. However, it is not a methodology aimed at saving time by: a) constructing or extracting a LR from texts using machine learning methods [18, 19] or b) merging different LR into a definitive one [16, 17]. We follow a software engineering approach (where thinking precedes action) by focusing on analysis, design and reuse (as understood by software engineering) aspects. Thus, we apply the principled methods and techniques of software engineering (which guide the development of user-oriented, readable, modular, extensible, and reusable software) to the design and implementation of ontology-based LR.

Finally, a very important aspect in developing a LR is the development of its graphical user interface(s). However, the majority of the management software tools for LR are just briefly described, and although some are extensively described [4, 14], there is no declared software engineering approach for their development [1]. Although not covered in this paper, our methodology encompasses this aspect too.

References

1. Sáenz, F. and Vaquero, A. Applying Relational Database Development Methodologies to the Design of Lexical Databases. Database Systems 2005, IADIS Virtual Multi Conference on Computer Science and Information Systems, (2005)
2. Moreno A. Diseño e Implementación de un Lexicón Computacional para Lexicografía y Traducción Automática. Estudios de Lingüística Española, vol(9), (2000)
3. Hayashi, L. S. and Hatton, J. Combining UML, XML and Relational Database Technologies - The Best of all Worlds for Robust Linguistic Databases. Proceedings of the IRCS Workshop on Linguistic Databases, (2001)
4. Moreno, A. and Pérez, C. Reusing the Mikrokosmos Ontology for Concept-Based Multilingual Terminology Databases. In Proc. of the 2nd International Conference on Language Resources and Evaluation , (2000) pp 1061-1067.

5. Nirenburg, S., McShane, M. and Beale, S. The Rationale for Building Resources Expressly for NLP. In Proc. of the 4th International Conference on Language Resources and Evaluation, (2004)
6. McShane, M.; Nirenburg, S. and Beale, S. An Implemented, Integrative Approach to Ontology-based NLP and Interlingua . Working Paper #06-05, Institute for Language and Information Technologies, (2005)
7. Cimino, J. Desiderata for controlled medical vocabularies in the twenty-first century. Methods of Information in Medicine, 37(4-5):394—403, (1998)
8. Ide, N., and Veronis, J. Extracting Knowledge Bases from Machine-Readable Dictionaries: Have we wasted our time? In Proc. of the First International Conference on Building and Sharing of Very Large-Scale Knowledge Bases, (1993)
9. Guarino, N. Some Ontological Principles for Designing Upper Level Lexical Resources. A. Rubio et al. (eds.), In Proc. of the First International Conference on Language Resources and Evaluation, (1998) pp 527-534.
10. Welty, C. and Guarino, N. Supporting ontological analysis of taxonomic relationships", Data and Knowledge Engineering vol. 39(1), (2001) pp 51-74.
11. Bodenreider O. Circular Hierarchical Relationships in the UMLS: Etiology, Diagnosis, Treatment, Complications and Prevention. In Proceedings of the AMIA Symposium, (2001)
12. Bouaud, J., Bachimont, B., Charlet, J. and Zweigenbaum, P. Acquisition and Structuring of an Ontology within Conceptual Graphs. In Proceedings of the ICCS'94 Workshop on Knowledge Acquisition using Conceptual Graph Theory, (1994)
13. Evans, R., and Kilgarriff, A. MRDs, Standards and How to do Lexical Engineering. In Proceedings of the Second Language Engineering Convention, (1995) pp. 125–32.
14. Feliu, J.; Vivaldi, J.; Cabré, M.T. Ontologies: a review. Working Paper, 34. Barcelona: Institut Universitari de Lingüística Aplicada. DL: 23.735-2002 (WP), (2002)
15. Martin, P. Correction and Extension of WordNet 1.7. In Proc of the 11th International Conference on Conceptual Structures, (2003) pp 160-173.
16. Oltramari, A.; Prevot, L.; Borgo, S. Theoretical and Practical Aspects of Interfacing Ontologies and Lexical Resources. In Proc. of the 2nd Italian SWAP workshop, (2005)
17. Philpot, A., Hovy, E. and Pantel, P. The Omega Ontology. In IJCNLP Workshop on Ontologies and Lexical Resources, (2005) pp 59-66.
18. Makagonov, P., Ruiz Figueroa, A., Sboychakov, K. and Gelbukh, A. Learning a Domain Ontology from Hierarchically Structured Texts. In Proc. of Workshop "Learning and Extending Lexical Ontologies by using Machine Learning Methods" at the 22nd International Conference on Machine Learning, (2005)
19. Makagonov, P., Ruiz Figueroa, A., Sboychakov, K. and Gelbukh, A. Studying Evolution of a Branch of Knowledge by Constructing and Analyzing Its Ontology. In Christian Kop, Günther Fliedl, Heinrich C. Mayr, Elisabeth Métais (eds.). Natural Language Processing and Information Systems. 11th International Conference on Applications of Natural Language to Information Systems, (2006)
20. Nirenburg, S., McShane, M., Zabludowski, M., Beale, S. and Pfeifer, C. Ontological Semantic text processing in the biomedical domain. Working Paper #03-05, Institute for Language and Information Technologies, University of Maryland Baltimore County, (2005)
21. Smith B, Ceusters W, Klagges B, Köhler J, Kumar A, Lomax J, Mungall CJ, Neuhaus F, Rector A, Rosse C. Relations in Biomedical Ontologies. Genome Biology, 6(5), (2006)

Model-Driven Tool Interoperability: An Application in Bug Tracking*

Marcos Didonet Del Fabro, Jean Bézivin, and Patrick Valduriez

ATLAS Group, INRIA and LINA University of Nantes
2, rue de la Houssinière, BP 92208, 44322, Nantes cedex 3, France
{marcos.didonet-del-fabro, jean.bezivin}@univ-nantes.fr,
patrick.valduriez@inria.fr

Abstract. Interoperability of heterogeneous data sources has been extensively studied in data integration applications. However, the increasing number of tools that produce data with very different formats, such as bug tracking, version control, etc., produces many different kinds of semantic heterogeneities. These semantic heterogeneities can be expressed as mappings between the tools metadata which describe the data manipulated by the tools. However, the semantics of complex mappings (*n:1*, *1:m* and *n:m* relationships) is hard to support. These mappings are usually directly coded in executable transformations using arithmetic expressions. And there is no mechanism to create and reuse complex mappings. In this paper we propose a novel approach to capture different kinds of complex mappings using correspondence models. The main advantage is to use high level specifications for the correspondence models that enable representing different kinds of mappings. The correspondence models may be used to automatically produce executable transformations. To validate our approach, we provide an experimentation with a real world scenario using bug tracking tools.

Keywords: complex mappings, semantic heterogeneities, tool interoperability, MDE (Model Driven Engineering).

1 Introduction

A software tool, e.g., text editing, bug tracking, needs to manipulate data that may be persistent (e.g., stored in a relational database) or transient (e.g., the execution state of the tool). Today, many different tools can be used to solve similar problems. As a result of increased collaboration between organizations and to rapidly changing environments, it is often necessary that one tool uses the data produced by another tool. However, the data produced by distinct tools are often heterogeneous with very different data formats, thus making tool data integration difficult.

The integration of heterogeneous data sources has been studied for a long time in data integration applications [21, 26, 1, 10]. In order to integrate the data of different tools, it is necessary to identify the semantic heterogeneities. The format and

* This work is partially supported by ModelPlex project.

R. Meersman, Z. Tari et al. (Eds.): OTM 2006, LNCS 4275, pp. 863 – 881, 2006.

semantics of tool data is typically specified as tool metadata. Semantic heterogeneities can be expressed as mappings which specify the relationships between elements of tools metadata.

Many solutions have proposed different kinds of mappings ranging from 1-to-1 correspondences [26, 27, 24] to ontology bridges [22, 13, 11]. However, they typically provide a limited set of semantic relationships, e.g., equality and equivalence. They do not provide support to explicitly define the semantics of complex kinds of mappings such as mapping expressions. Mapping expressions are manipulations over tool elements that involve *1:m*, *n:1* or *n:m* relationships, e.g., splitting an element *Address* into *Street* and *Number*. Most solutions implement complex mappings directly in executable transformations using generic arithmetic expressions, e.g., *project_duration = end_date – start_date*, *name = first_name + last_name*. In this case, the semantics of the entire mapping (e.g., *"name concatenation"*) is not defined in the mapping specification, but in the mapping expression itself. Therefore, it is difficult to create and reuse these expressions. The lack of explicit semantics also hardens the task of deriving these mappings into executable transformations. The transformations are responsible to translate the data produced by a tool into a different format that can be understood and consumed by another tool.

In this paper, we propose a practical solution based on Model Driven Engineering together with data integration techniques. Our approach is useful to specify and capture complex semantic heterogeneities, and to automatically produce executable transformations. In our approach, the data manipulated by a tool is a model, called a tool model. A model conforms to a metamodel which is a formal description of the model.

We classify different kinds of tool semantic heterogeneities according to their complexity, and we propose a practical solution to express the mapping semantics in a correspondence metamodel, i.e., at the specification level. The metamodel elements are created with a vocabulary close to their semantic meanings, e.g., *override*, *concatenate*, *split*. A correspondence model conforming to this metamodel contains the mappings between the tool metamodels.

The correspondence models are used to generate executable transformations. A transformation is also a model, so the heterogeneities (e.g., mapping expressions) are translated into constructs of specific transformation languages, e.g., XSLT. We generalize the process of producing transformations into a pattern that is automatically executed. This pattern may be incrementally modified to handle different semantic heterogeneities. This is a frequently executed operation in model driven engineering which can be encapsulated in a *TransfGen* operation.

The main contributions of this paper are the following. First, we develop correspondence metamodel extensions that fully capture different kinds of semantic heterogeneities between tool metamodels. We emphasize the creation of complex mapping expressions. Second, we provide a generic pattern to automatically generate transformations based on correspondence models. Third, we consider all entities as models. This allows us to apply the same principles to manipulate the tool, correspondence and transformation models. To validate our approach, we provide an experimentation with a real world interoperability scenario using bug tracking tools and our AMW (Atlas Model Weaver) prototype [9].

This paper is organized as follows. Section 2 describes a motivating example in bug tracking tool interoperability. Section 3 presents the base concepts and the definition of tool correspondences. Section 4 explains how these correspondences are translated into a transformation model. Section 5 presents our experimental validation. Section 6 discusses related work. Section 7 concludes.

2 Motivating Example

We illustrate different kinds of semantic heterogeneities in tool interoperability with a bug tracking scenario. Bug tracking tools manage the bugs (reporting, fixing, etc.) of a given application. Today, many bug tracking tools are available, e.g., GNATS, Mantis, Bugzilla, and many others [15]. Consider two autonomous software development companies, C_A and C_B, and a set of N bug tracking tools. Company C_A uses tool T_i and company C_B uses tool T_i. They need to collaborate without aligning their software development practices. This is due to pragmatic reasons, e.g., the companies already participate in other cooperative projects.

We illustrate this situation using two bug tracking tools, Bugzilla [6] and Mantis [23]. Bugzilla is a general purpose, open source bug tracking tool. It provides features such as error tracking and quality assurance management. The metadata of Bugzilla is illustrated in Figure 1.

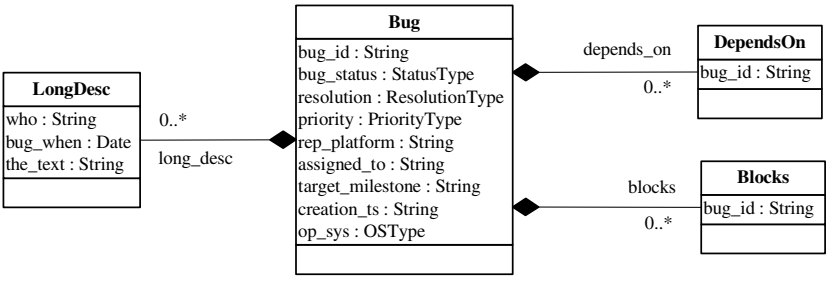


Fig. 1. Bugzilla metadata

Mantis is another bug tracking tool. It differs from Bugzilla as a light weight tool which allows adding new modules. The metadata of Mantis is illustrated in Figure 2.

We observe that it is possible to establish different kind of mappings between the elements of the tools metadata. The most common kind of mappings is equality, where two concepts are said to be equal. For example, a software bug is represented by *Bug* in Bugzilla and *Issue* in Mantis. As another example, the date a bug is created is represented by *creation_ts* and *date_submitted*. There are also elements representing equivalent data, but not the same, e.g., *target_milestone* is the version where a bug will be fixed, and *fixed_in_version* is the version where a bug was fixed.

There are also more complex kinds of mappings. For example, Bugzilla has two kinds of relationships between bugs: *depends_on* and *blocks*. In *Mantis*, bugs are

related to each other using the element *relationships*, which points to *Relationship*. The relationship type is stored in the element *type*. As another example, *assigned_to* contains the responsible to solve a given bug in Bugzilla. In Mantis the relationship *assigned* points to element *Person* (that contains elements *login*, *value* and *id*).

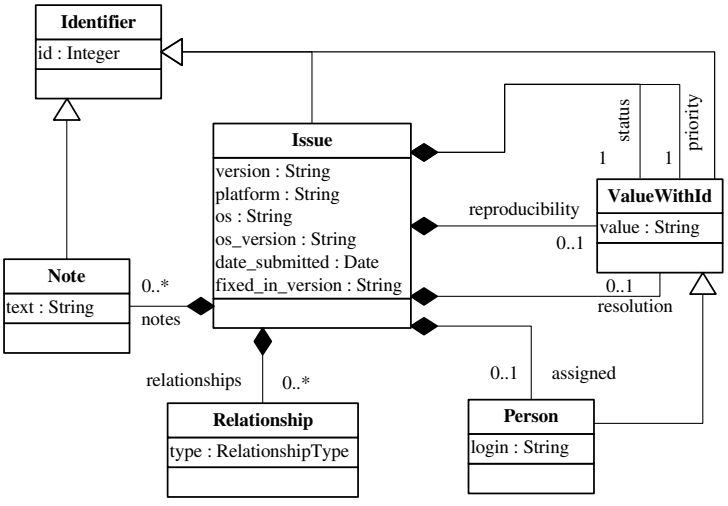


Fig. 2. Mantis metadata

In addition, there are semantic heterogeneities at the data level. For instance, the element *bug_status* in Bugzilla and relationship *status* in Mantis (that points to *ValueWithId*) contains the bug state (e.g., a bug was included in the database, a bug was solved, etc.), and the element *priority* contains the priority to solve a given bug (e.g., immediate, urgent). Each tool has its own set of status and priorities. For example, it is necessary to take into account that the *priority* with value "*P_1"* in *Bugzilla* is translated into the value "*urgent"* in Mantis. The same analogy applies to the element *status.* Different kinds of heterogeneities and the other elements not explained here are discussed later in the paper.

Traditional data integration applications usually create mappings to capture similarity heterogeneities (e.g., equality, equivalence). These mappings can be used to produce transformations that execute the translations from Bugzilla to Mantis. However, complex mapping expressions and data level heterogeneities are coded either in some element in the mappings, either in the produced transformations. For example the developer must code how to translate between the enumerations values in one specific language. The lack of explicit semantics for complex expressions hardens the creation of mappings because there is no domain information about the possible mappings. The possible mappings are virtually unlimited when using generic arithmetic expressions. This way is not possible to understand all the mappings without analyzing the entire expression in the produced transformations. This also reduces the reusability of these expressions. In addition, there is not enough semantic information to automatically

produce the transformations, which is a frequently executed operation in model management. The mappings and produced transformations must be kept synchronized.

In order to efficiently achieve tool interoperability, all these kinds of mappings must be explicitly specified. These mappings must be derived into executable transformations. This process must be efficient, such that new transformations between other tools can be rapidly developed.

3 Tool Heterogeneity

In this section, we motivate the use of correspondence metamodel extensions to capture different kinds of tool semantic heterogeneities. First, we define what is a model which is the basic concept underlying our solution. Second, we define the tool heterogeneity problem and a core correspondence metamodel. Finally, we classify different kinds of tool semantic heterogeneities and we propose a set of metamodel extensions to express these heterogeneities.

3.1 Models

We abstract implementation and representation issues by using an integrated modeling platform. We present the model definition below (following [17]).

Definition 3.1 (Directed graph). A directed multigraph $G = (N_G, E_G, \Gamma_G, v)$ consists of a finite set of nodes N_G and a finite set of edges E_G, a mapping function $\Gamma_G : E_G \rightarrow N_G \times N_G$ and a labeling function $v : N_G \cup E_G \rightarrow A$. The type *A* is of any data type, such as characters, integers or classes.

Definition 3.2 (Model). A model $M = (G, \omega, \mu)$ is a triple where:
- $G = (N_G, E_G, \Gamma_G)$ is a directed multigraph,
- ω is itself a model (called the reference model of M) associated to a graph $G_\omega = (N_\omega, E_\omega, \Gamma_\omega)$,
- $\mu : N_G \cup E_G \rightarrow N_\omega$ is a function associating elements (nodes and edges) of G to nodes of G_ω.

The relation between a model and its reference model is called *conformance*. This definition allows an indefinite number of levels. However, we observe from different domains (XML, RDBMS, ontologies) that only three levels are needed. We call these three levels metametamodel (M3), metamodel (M2) and terminal model (M1). A *metametamodel* is a model that is its own reference model. A *metamodel* is a model such that its reference model is a metametamodel. A *terminal model* is a model such that its reference model is a metamodel.

3.2 Tool Heterogeneity

The tool data and metadata are represented as models and metamodels. Thus, the tool heterogeneities are expressed as mappings between tool metamodels. The mappings types are specified in a correspondence metamodel. We define tool, mappings and correspondence metamodel below.

Definition 3.3 (Tool). A tool T is a tuple $<M_t, S_t>$, where:

- $M_t = (G, MM_t, \mu)$ is the tool model. M_t is the data that is manipulated by T,
- MM_t is the reference model (metamodel) that represents the tool metadata,
- $S_t = \{s_i; i = [1..n]\}$ is the set of services (querying, updating, inserting, etc.) provided by T. Every service $s \in S_t$ must respect the constraints specified in MM_t.

Consider a bug tracking tool $T_a = <M_{ta}, S_{ta}>$. The metamodel MM_{ta} specifies how the bugs are organized, the properties of a bug, the possible states of a bug during its life cycle, etc. The model M_{ta} has the value of the bugs, e.g., that a given bug "B" has a status of "in correction" to a developer called "Joseph". The set S_{ta} contains miscellaneous services: the inclusion a new bug in the database, the update of a bug status, the query of a set of bugs, and so on.

Consider another bug tracking tool, $T_b = <M_{tb}, S_{tb}>$ with a different model, reference model and set of services. The semantic heterogeneities between metamodels MM_{ta} and MM_{tb} are expressed as mappings. The mappings between tool metamodels have different types, structures and semantics. However, intuitively, they depict the notion of typed-links between (meta) model elements.

Definition 3.4 (Mapping). Given two models M_{ta} and M_{tb}, a mapping *M* is a tuple $<S_a, S_b, T>$, where:

- S_a is a set of elements from the model M_{ta},
- S_b is a set of elements from the model M_{tb},
- T is the type of mapping between the sets S_a and S_b.

There are many different kinds of mappings, for instance *equality*, *equivalence* or, *generalization*. These are simple kind of mapping that express element similarity, usually denoting 1-to-1 links. Complex mappings have multiple cardinalities and semantic meaning. These kind of mappings abstract commonly used mapping expressions, e.g., the average between a set of elements, or the concatenation of strings. We specify the different mapping types in a correspondence metamodel.

Definition 3.5 (Correspondence metamodel). A correspondence metamodel is a model $M_C = (G_C, \omega_C, \mu_C)$ that define mapping types, such that:

- G_C has two basic types of nodes: *links* and *link endpoints*,
- *link* denote the mapping type, and refers to multiple *link endpoints,*
- *link endpoints* refer to the mapped elements.

Consider the mapping expression $t = s_1 + s_2 + s_3 + s_4 / 4$. The mapping language contains the addition and subtraction operators, plus the tokens (the model elements). The language does not explicitly specify that it is possible to create average expressions. The semantic is only known if we analyze the expression itself. In our solution, we create a link type *average* that abstracts the semantics provided by the combination of operations "+" and "/". This process is the *promotion* of the mapping semantics into the correspondence metamodel. The link type refers to a link endpoint with cardinality N (the source elements), and to a link endpoint with cardinality 1 (the target element). The mapping expression (the link between the elements) is created in a correspondence model conforming to the correspondence metamodel.

3.2.1 Core Correspondence Metamodel

We create a core correspondence metamodel based on Definition 3.5. The metamodel is illustrated in Figure 3. The core metamodel has elements with information about link type, link endpoints and element identification. Element identification is a practical solution for saving unique identifiers for the linked elements.

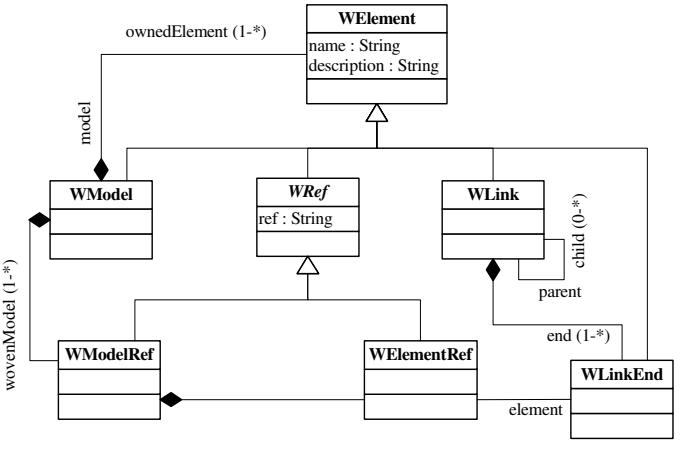


Fig. 3. The core correspondence metamodel

WElement is the base element from which all other elements inherit. It has a name and a description. *WModel* represents the root element that contains all model elements. *WLink* denotes the link type. *WLink* has a reference *end* to associate it with a set of link endpoints (*WLinkEnd*). *WLink* can have children links (*child* reference). Every *WLinkEnd* references one *WElementRef*. The attribute *ref* contains a unique identifier of the linked elements. *WElementRef* is not referenced directly by *WLink* because it is possible to refer to the same model element by different link endpoints, e.g., one model element may participate in more than one mapping expression. *WModelRef* is similar to *WElementRef*, but it contains references to the models as a whole. The *WLink* element must be extended to create different link types, e.g., equality, average and others. Different link types and link endpoints are added using metamodel extensions [9].

3.3 Metamodel Extensions for Tool Interoperability

As already stated, it is not possible to create a metamodel containing all types of links for tool interoperability. We propose to create metamodel extensions to capture different types of links. We classify them in three major groups according to the complexity of the links semantics.

The link types are defined in a simplified version of the KM3 metametamodel (see the complete syntax in [17]). KM3 is formed by classes. Classes may inherit from other classes, and are formed by attributes and references (attributes and references have a type and a cardinality). The syntax of KM3 is similar to object notations.

3.3.1 Similarity Expressions

Similarity expressions represent resemblance links between metamodel elements. These expressions are the link types encountered in most semantic-based mapping solutions. There are different kinds of similarity expressions. We describe them below.

Equality: a pair of element represents exactly the same information. For example the platform of the application the bug was detected is represented by *rep_platform in Bugzilla and platform* in Mantis. The link type does not specify the exact data type (*String*, *Class*, etc.). The data type is specified when deploying the solution (as extensions of *WLinkEnd*).

***class** Equal **extends** WLink { **ref** source : <DataType>; **ref** target : <DataType> }*

Equivalence: the linked elements contain similar information, but not exactly the same. However, the translation semantics may be the same as in equality links, i.e., one target element receives the value of a source element. We add a description attribute to provide additional information about the equivalence, and a similarity measure.

***class** Equivalence **extends** WLink {**ref** source:<DataType>;*
* **ref** target: <DataType>; **attr** description : String; **attr** similarity : Integer}*

Disjointness: two elements cannot be present at the same time because they have incompatible data. The link type also contains a description.

***class** Disjoint **extends** WLink {**ref** source:<DataType>;*
* **ref** target : <DataType> ; **attr** description : String }*

Generality: one model element is more general than the other.

***class** Inherit **extends** WLink { **ref** parent : <DataType>; **ref** child : <DataType>}*

Non equivalence: it is not always possible to translate all the information produced by one tool into another tool. Some elements from the tool metamodels do not have any semantic relationship, or are not relevant for a given translation and do not need to be generated. The element may be simply ignored. However, it is important for the application developer to be aware of what is not translated. For example the *reproducibility* element contains the frequency of reproduction of a given issue. This element does not exist in Bugzilla.

***class** Unique **extends** WLink { **ref** element : <DataType> }*

3.3.2 Mapping Expressions

Mappings expressions are mappings that involve a set of source elements and a set of target elements. The definition of high level mapping expressions capable of capturing different kinds of semantic heterogeneities is a main contribution of this paper. The correspondence metamodel encapsulates mapping expressions in metamodel elements.

Many-to-one: many-to-one expressions links set of elements of the source model with a single target element. For example the elements *os* and *os_version* from Mantis contains the operating system and the operating system version. In Bugzilla, this information is available in one single attribute *op_sys*.

class *ManyToOne* ***extends*** *WLink {*

ref *source [*]: <DataType>;* ***ref*** *target: <DataType>}*

Split or one-to-many: the opposite of many-to-one expressions, i.e., split expressions link more than one target element with a single source element.

class *OneToMany* ***extends*** *WLink {*

ref *source: <DataType>;* ***ref*** *[*] target: <DataType>}*

Many-to-many: links a set of elements of source models with a set of elements of target models, for instance the reorganization of the elements of *LongDesc* into the elements of *Note*.

class *ManyToMany* ***extends*** *WLink {*

ref *source [*]: <DataType>;* ***ref*** *[*] target: <DataType>}*

New values in the target: generates values in the target model that do not have a correspondence in the source model. The values are automatically generated (e.g., to automatic generate IDs elements) or take a predefined value from user input.

The class *AutoSetValue* is extended into *AutomaticGenInt* and *ManualInput.* The class *AutomaticGenInt* reads the element that is referred by the *target* reference and generates a random number for it. The class *ManualInput* sets the *target* reference attribute with the value of *sourceValue*.

class *AutoSetValue* ***extends*** *Equal {}*
class *AutomaticGenInt* ***extends*** *AutoSetValue {}*
class *ManualInput* ***extends*** *AutoSetValue {* ***attr*** *sourceValue : <DataType>}*

3.3.3 Data Value Expressions

Data value expressions differ from mapping expressions because they also evaluate the model elements, not only the metamodel elements. Data value expressions modify the source model values to make them compatible with the target model.

The class *DataExpression* refers to a set of value equivalences. The *source* element is evaluated, and if it is equal to one *sourceValue* from the set of equivalences, it sets the target element with the corresponding *targetValue.* The equivalences may be of any data type.

class *DataExpression* ***extends*** *WLink {* ***ref*** *equiv [*] : Equivalence}*
class *Equivalence* ***extends*** *WLinkEnd {*
ref *sourceValue : <DataType>;* ***ref*** *targetValue : <DataType> }*

We illustrate data value expressions with the *resolution* element. The resolution contains the correction status of a bug (e.g., if it was fixed or not). In Mantis, this element may have the values: *OPEN, FIXED, REOPENED*. The possible values in Bugzilla are: *NEW, FIXED, INVALID, WONTFIX*.

4 Interpreting Tool Heterogeneity

In the previous section, we explained how to define different metamodel extensions to capture semantic heterogeneities. The next step is to create a correspondence model conforming to these extensions and to derive the model into executable transformations. These transformations translate one tool model into another.

In this section, we first introduce a match operation that creates a correspondence model. Then, we present a generic pattern used to automatically produce a model transformation, which is responsible to translate one tool model into another.

4.1 Match Operation

The match operation creates a correspondence model conforming to an extended correspondence metamodel. The match operation is divided into an automatic and a manual phase. The automatic phase executes a set of matching algorithms to search for similar concepts in the tool metamodels. First, we generate a correspondence model containing the cross product of all metamodel elements of each tool. Then, a set of matching algorithms is executed sequentially to calculate a similarity value for every pair of elements. We use existing string comparison algorithms [7], e.g., Levenshtein distance, edit distance and QGrams, and an adaptation of the similarity flooding algorithm [24]. We match only classes, attributes and references. The result is filtered to obtain only the best similarity values, based on a similarity threshold. The correspondence model contains only equality mappings.

The manual phase refines the correspondence metamodel by deleting wrong equality matchings and by adding the complex mapping expressions and data value expressions. This operation is done with the help of a user interface.

4.2 Generic Transformation Pattern

The definition of the generic transformation pattern relies on three facts. First, the core correspondence metamodel is formed by links, link endpoints and extensions of these elements. Second, declarative transformations languages have similar structure. Third, we use declarative transformation patterns that specify only what to transform, and not how to transform. The transformation pattern contains the execution semantics of the correspondence model, because it transforms the different types of links into executable mapping expressions in some transformation language.

We use higher-order transformations (HOT) to specify the generic pattern. A HOT takes as input a correspondence model conforming to an extension of the correspondence metamodel and transforms it into a transformation model.

Definition 4.1 (Higher-order transformation). A higher-order transformation is a transformation $T_{HOT} : T_{IN} \rightarrow T_{OUT}$, such that the input and/or the output models are transformation models. Higher-order transformations either take a transformation model as input, either produce a transformation model as output, or both.

We create a simple syntax for a transformation metamodel to define the generic pattern (as illustrated in Figure 4). The keywords are in bold font. The transformation has a set of declarative rules. The *input* element matches the input correspondence metamodels. The *output* element creates a new element in the output model. The output element has *bindings* to assign the source values to the target elements. The correspondence metamodel has one extension of *WLink* (as shown below) to denote source and target elements. The pattern can also be used with different metamodel extensions.

class *WLinkST* ***extends*** *WLink { **ref** source : WLinkEnd; **ref** target : WLinkEnd }*

The pattern contains four rules (see Figure 4). The rule *newRule* creates transformation rules. The rule *newOutput* creates the output elements. Both are based on the value of the *target* reference of a given link. The rule *newInput* creates the input element, and it is based in the value of the *source* reference. This rule may have a filtering condition depending on the link type. The rule *newExpression* creates different mapping expressions. The mapping expressions are created as bindings to the output elements.

```
Module TransfGen (C: ωC)
inputModel: C: ωC /* a correspondence model conforming to a metamodel ωC*/
outputModel: T: ωT /* a transformation model conforming to ωT */
rule newRule
 input WLinkST (parent isA WModel) /*classifiers (classes, references, attributes)*/
 output Rule
     input ← source
     output ← target
rule newInput
 input WLinkEnd (link.source = self)
 output InputElement
    element ← getElement (element.ref)
    condition ← /*depends on the WLinkST and WLinkEnd types*/
rule newOutput
 input WLinkEnd (link.target = self)
 output OutputElement
    element ← getElement (element.ref)
    bindings ← link.child /*get the sibling WLinkEnd*/
rule newExpression
 input WLinkST (parent isA WLinkST)
  output Binding
    source ← MapExp (getElement (source.element.ref) ) /*mapping expressions here,*/
    target ← getElement (target.element.ref)          /*according to the WLinkST type*/
```

Fig. 4. Higher-order transformation pattern

This pattern is the basis to define a new model management operation called *TransfGen.* This way it is possible to separate the tool interoperability process into distinct operations. The correspondence model is created by a *Match*. The correspondence model is translated into a transformation model using *TransfGen*. The translation between models are encapsulated in automatically generated transformations, which are themselves specific data transformation operations.

5 Experimental Validation

In this section we validate our approach with experiments using the bug tracking tools from the motivating example, Mantis and Bugzilla. The experiments are conducted using our model management platform, which is composed by different plugins to manipulate models. The two plugins used are the AMW (ATLAS Model Weaver) plugin [9] and the ATL (ATLAS Transformation Language) plugin [16]. AMW is responsible for managing the metamodel extensions and for the manual match. ATL is used to implement all the model transformations of the process. ATL has a textual

concrete syntax and an engine to execute the transformations. Both plugins are open source and are available as Eclipse subprojects [2, 3].

We first show the creation of a correspondence model based on the correspondence metamodel extensions from Section 3. Then we demonstrate how we use the generic transformation pattern to interpret the correspondence model and to automatically produce model transformations. We end with a discussion about our results.

5.1 Correspondence Model

The metamodels of both tools are stored in different data sources. The tool metamodels are originally in SQL-DDL. They are translated into Ecore [12], which is the metametamodel used by our plugins (AMW and ATL). The semantics of Ecore is very close to KM3. This allows us to write metamodels using KM3 textual syntax. One part of the metamodels is illustrated in the graphical concrete syntax in the motivating example. The Bugzilla metamodel has 146 elements. The Mantis metamodel has 62 elements.

We implement the metamodel extensions defined in Section 3. We show below an excerpt of the correspondence metamodel. It specifies a data value expression used to translate enumeration values. It compares the value of given *source* element with the set of *sourceValue*, and sets the *target* element with the corresponding *targetValue*.

```
class EnumerationEquiv extends DataExpression {ref equiv [*] : EnumEqual};
class EnumEqual extends Equivalence {
      ref sourceValue: String; ref targetValue: String };
```

We create the correspondence model using ATL transformations to execute the sequence of matching algorithms, which refine the initial input (the cross-product of elements) and generate a correspondence model. Our AMW plugin is used to generate the interoperability metamodel based on a set of extensions and to refine the correspondence model during the manual phase.

An excerpt of the correspondence model is shown in Figure 5. We use a human readable syntax to represent information models, similar to HUTN [29].

```
EnumerationEquiv = {
    source.ref = Left.priority.id;
    target.ref = Right.priority.id;
    equivalence = { source = "NONE";  target = "pt_null"};
    equivalence = { source = "low";  target = "pt_P1"};
};
Left = {
    name ="Mantis";
    ref = "c:\Tool_interoperability\Mantis.ecore";
    priority {  id = "EAttribute_priority";   }
};
Right = {
   name ="Bugzilla";
   ref = "c:\Tool_interoperability\Bugzilla.ecore";
    priority {   id = "EAttribute_priority";   }
}
```

Fig. 5. A correspondence model

The model contains the equivalencies between the *priority* values. Note that both tool models have a priority property and both have the same ID "EAttribute_priority". This does not cause problems because it is relative to the containing model.

The complete correspondence model has 312 elements. This difference on the number of elements is due to the structure of the correspondence metamodel, because for every couple of referred elements there is at least one element indicating the link type, plus the source and target elements. In addition, the source and target elements refer to an element that contains their identifiers (in the *Left* and *Right* elements).

5.2 Interpreting the Correspondence Model

The execution semantics of the correspondence model is specified in a transformation that takes the correspondence model as input and produces a transformation model as output. The transformation (485 lines) is implemented based on the generic transformation pattern. The ATL transformation rules are divided in three parts: the *from* block filters the appropriated model elements by their type; the *to* block contains the declarative code; the *do* block contains imperative code. We show in Figure 6 the rules that interpret the metamodel extension to translate the enumeration values. The AMW identifier denotes the correspondence metamodel. The ATL identifier denotes the transformation metamodel.

```
rule EnumDataTranslation {
  from amw : AMW!EnumerationEquiv
  to atl : ATL!Binding (
       propertyName <- MOF!EClassifier.getInstanceById(amw.target.element.ref).name
    )
  do { atl.value <- thisModule.CreateEnum(amw, amw.enumEqual);}
}
rule CreateEnum(amw: AMW!EnumerationEquiv, attrEnum: Sequence (AMW!EnumEqual)){
  to ifExp : ATL!IfExp (
          thenExpression <- targetEnum,
          condition <- operation
        ),
        operation : ATL!OperatorCallExp (
          operationName <- '=',
          arguments <- sourceEnum
        ),
        endExp : ATL!StringExp (),
        sourceEnum: ATL!StringExp (
          stringSymbol <- attrEnum->first().sourceValue.toString()
        ),
        targetEnum : ATL!StringExp (
          stringSymbol <- attrEnum->first().targetValue.toString()
        )
  do { operation.source <- amw, amw.source->collect(e | e.element.ref),true);
          if ( attrEnum->size() = 1 ) {
               ifExp.elseExpression <- endExp;
          } else {
               ifExp.elseExpression <- thisModule.CreateIfEnum(amw,
                                       attrEnum->subSequence(2,attrEnum->size()));
          }
  }
}
```

Fig. 6. Higher-order transformation

The rule *EnumDataTranslation* matches the element *EnumerationEquiv* from the correspondence model. It produces a *Binding* element conforming to the ATL metamodel. A binding has a *propertyName* that corresponds to the target model element. The target element is obtained by *getInstanceById* function. The property *value* calls the rule *CreateIfEnum.* It receives the set of enumerations as parameters and produces a model with a set of nested *IfExp* (conditional expressions).

The *IfExp* contains a *condition* expression, which is formed by an equality operator (*OperatorCallExp*). This operation compares the source value of the enumerations and sets the correct target value specified at *thenExpression*. The *StringExp* elements return the *sourceValue* and *targetValue* (an empty *String* if there is no equivalence). The complete transformation produces a transformation model with a set of rules. This model is extracted into a text representation that is executed in the ATL engine.

5.3 Discussion

The metamodel extensions enable producing a domain specific (tool interoperability) correspondence metamodel. Among the different metamodel extensions that are created, the most used are the concatenation of elements (e.g., *os* concatenated with *os_version*), data type conversions (e.g., *Integer* to *String,* references to attributes, etc.) and conversion of enumerations values.

One interesting observation is that the values of the enumerations from Mantis are not described in the metamodel, only in a Php file. Since the tool metamodels cannot be modified (otherwise the services provided might not work properly), the enumerations are added in one metamodel extension. This is a very specific extension, which is probably not useful outside the bug-tracking example, but it is still necessary to be able to create the output transformation.

The correspondence model has composite elements that conform to a combination of metamodel extensions. For instance we combine the conversion of “references to attributes” extension with the “concatenation” extension. This way, it is possible to create more complex output transformation models with the same set of extensions.

The metamodel extensions ease the task of repeatedly creating complex mapping and data value expressions between tool metamodels. The adaptive user interface is used together with semi-automatic matching algorithms (see the survey at [32]). The extensibility of the correspondence metamodel enables leaving human intervention essentially on the matching phase, because all the necessary information to produce transformations is available in the correspondence model. This is different from traditional approaches that have an extra step of mapping discovery [19, 26]. However, it is still possible that a correspondence metamodel covers most semantic interoperability cases, but not all. Complex expressions that are not often used can be coded manually in the final generated transformation.

The declarative structure of the correspondence metamodel allows a clear separation of the input model (the correspondence model) from the output model (a transformation model). Thus, it is relatively straightforward to modify only the output model and produce different transformation models. This also enables generating

different expressions in the output transformation. For instance, the translation of enumeration values may be implemented as nested ifs (our final choice), or using *case*-like statements. This opens the possibility of optimizations of the output transformations (however, this is not the focus in this work). On the negative side, transformation languages may have complicated metamodels, in particular for querying and navigation expressions (e.g., OCL, XPath).

Another important result is that we are capable to use most of the metamodel extensions also in the importing process from SQL-DDL to an Ecore metamodel. The process is the following: we create a SQL-DDL metamodel conforming to Ecore (to support standard CREATE TABLE statements). The textual SQL-DDL is translated into a model conforming to the SQL-DDL metamodel. We then use most part of the correspondence metamodel extensions (excluding for instance the extensions concerning enumerations) to create a correspondence metamodel with AMW, and then a correspondence model to link the SQL-DDL model with a KM3 metamodel The translation from KM3 to Ecore is straightforward. The SQL-DDL model has 48 elements. The KM3 metamodel has 47. The correspondence model has 132. The output transformation has 83 lines. This transformation translates the SQL-DDL model into a KM3 model. In this case, extensions to generate default values are constantly used, because KM3 models have attributes such as *lower*, *upper* (for cardinality), *isAbstract*, that are not present in the SQL-DDL definition.

To summarize, our experiments demonstrate that the use of MDE enables to improve two data integration phases (matching and transformation production) to solve tool interoperability problems in a practical and efficient manner. We are able to define different extensions of the core correspondence metamodel to cope with distinct kinds of semantic heterogeneity. We create a correspondence model using some matching algorithms and a user interface. We implement the transformation pattern that automatically generates a transformation to transform the tool models.

6 Related Work

There has been extensive work on data integration that can be applied to tool interoperability. The usual approach is to identify the relationships between elements and to save these relationships in some kind of mapping. The most common mappings are 1-to-1 correspondence [1, 27, 26, 24]. These correspondences are not adapted to represent complex mappings semantics.

The use of model-based correspondences was introduced in [30]. The correspondence model is used to merge models. However, it has only equality and similarity link types. More expressive representations have been proposed to bridge between different ontologies [28, 22, 11]. These approaches have mappings as first class entities. The set of valid mapping constructs involve complex axioms, such as equivalence and generalization. The main limitation is that the fixed set of mapping constructs cannot be extended in a straightforward way as in our approach.

In our solution, we present a correspondence metamodel that is capable of capturing virtually all of the representations above, because the metamodel is

extensible. This means we may specify a domain specific mapping with only 1-to-1 relationships until complex structures as in ontology-based approaches.

InfoQuilt [31] provides ways to represent mapping expressions through a library of mapping functions. However, the library can be used with no restriction, i.e., they are not separated by application domain. The functions are not part of the mapping definition, but expressions written in terms of the mapping language. The work in [18] presents a classification of the semantic and syntactic differences between schemas. This work proposes a *semPro* predicate to formalize the semantic proximity between elements. It is a formal work that focuses in the semantic heterogeneities, and not a complete integration platform as in our solution. It is a basis for our classification of tool heterogeneities, but we separate the heterogeneity types based on their complexity.

Our approach is complementary to existing matching algorithms, as we provide an efficient way to represent mapping expressions. For example the iMAP prototype [8] could be used to create a set of complex mapping expressions in our solution. iMAP implements different complex searchers. Every searcher could be associated with a correspondence metamodel extension.

The work in [14] proposes the alignment of ontologies based on the computation of similarities of 1-to-1 and 1-to-m mappings. The similarities are computed taking into account ontological structures. However, the similarities denote only equivalence mappings. The 1-to-m similarities could be used as input to algorithms that generate correspondence models with complex kinds of mappings.

The mappings are used to produce transformations. Clio [26] is one of the first solutions to provide a semi-automatic mechanism to produce transformations based on a set of correspondences. Our proposal has a similar architecture. However, Clio focuses on the generation of nested structures and on foreign key dependencies. There is no support for different kinds of complex mapping expressions. The work in [19] proposes an algorithm to generate XQuery. The algorithm uses 1-to-1 correspondences between a set of input XML schemas.

We differ from both approaches because we factor out part of the generation problem into a generic pattern. We leave the complexity of creating expressions to the matching phase, as in [8]. This means for instance that we do not implement a *chase* procedure to identify possible joins as in Clio. The generic pattern is independent of the structure of the input models (e.g., nested format), though still dependent of the core correspondence metamodel.

Model management solutions [5, 24, 4, 20] propose the creation of operations that encapsulate the most frequently executed metadata tasks. The work in [25] implements a model management platform using a logic mapping language. The logic language is translated into XSLT using an ad-hoc implementation. Our approach presents a model management solution focusing on the creation of element level constructs. The correspondence model as a whole acts as a high level specification for data integration operations.

To the best of our knowledge, none of the existing solutions consider the transformations and correspondences as models at the same time as in our approach. The model management operations may be applied to transformations as well. This

enables using the declarative pattern to generate transformations from a correspondence model, and to encapsulate this pattern into a *TransfGen* operator.

7 Conclusions

In this paper, we have presented a practical and flexible approach that improves data integration techniques applied to tool interoperability problems. We based our solution on MDE principles to capture the semantic heterogeneities and to produce operational mappings between these tools.

Considering two tools in a set of tools dealing with the same problem domain (bug tracking in our case), the main problem is to deal with different kinds of semantic heterogeneities, in particular, complex heterogeneities that involve mapping expressions. After having provided a classification of semantic heterogeneities between tools, we have shown how this classification may be translated in various types of links defined in a correspondence metamodel. Furthermore, the correspondence metamodel may be seen as an extension of a core metamodel that provides basic support for link management. The main original aspect of our approach is to offer maximum extensibility to capture the semantic of complex mapping and data value expressions.

We have shown that metamodel extensions allow expressing the different kinds of semantic heterogeneities with a dedicated vocabulary and in a declarative way. Every domain specific metamodel prevents from developing a generic language (and not well adapted) without the capability to explicitly express the semantic heterogeneities.

The correspondence models conforming to these metamodel were used to produce transformations. We have shown that the correspondence model can be interpreted following a generic and declarative pattern. The semantic of this pattern is the basis for a novel model management operation called *TransfGen*. Based on this pattern, we were capable to develop higher-order transformations that automatically produced output transformation models. The transformations were generated automatically because we leave all the human intervention to the matching phase.

Finally, considering all entities as models (tools, correspondence and transformations) enabled to manipulate all of them using the same set of principles. The main principle is to define different types of domain models and to apply transformations between them. This was particularly useful when specifying the semantic heterogeneities and when translating a correspondence model into executable transformation models.

We validated our approach within our model management platform using AMW and ATL plugins. We developed a domain specific metamodel to solve a set of tool interoperability problems. We created metamodel extensions for mapping expressions, data value expressions, and for elements that do not have equivalencies. We applied our solution in bug tracking tools using a real world setting.

As future work, we plan to extend the correspondence metamodel for different tool interoperability scenarios. We envisage verifying if our techniques adapt well to create *ModelGen* [4] operations. We also plan to study how to adapt existing matching algorithms to automatically create complex mappings.

References

1. Abiteboul S, Cluet S, Milo T. Correspondence and Translation for Heterogeneous Data. In proc. of ICDT 1997, pp 351-363
2. AMW: The ATLAS Model Weaver. Ref. site: http://www.eclipse.org/gmt/amw, 06/2006
3. ATL: ATLAS Transformation Language. Ref. site: http://www.eclipse.org/gmt/atl, 06/2006
4. Atzeni P, Cappellari P, Bernstein P A. Model independent schema and data translation. In proc. of EDBT 2006, pp 368-385
5. Bernstein P A. Applying Model Management to Classical Meta Data Problems. In proc. of the 1st CIDR 2003, pp 209-220
6. Bugzilla Bug Tracking Tool. Reference site: http://www.bugzilla.org, 06/ 2006
7. Cohen W, Ravikumar P, Fienberg S E. A Comparison of String Distance Metrics for Name-Matching Tasks. In proc. of IIWeb 2003, pp 73-78
8. Dhamanka R, Lee Y, Doan A, Halevy A, Domingos P. iMAP: Discovering Complex Semantic Matches between Database Schemas.In proc. of SIGMOD 2004
9. Didonet Del Fabro M, Bézivin J, Jouault F, Valduriez P. Applying Generic Model Management to Data Mapping. In proc. of BDA 2005, Saint-Malo, France, pp 343-355
10. Doan A, Halevy A. Semantic Integration Research in the Database Community: A Brief Survey. AI Magazine, Special Issue on Semantic Integration, Spring 2005, pp 83-94
11. Ehrig M, Haase P, Hefke M, Stojanovic N. Similarity for Ontologies - A Comprehensive Framework. In proc. of ECIS 2005
12. EMF. Eclipse Modelling Framework. Reference site: http://www.eclipse.org/emf, 06/2006
13. Euzenat J. An API for Ontology Alignment. In proc. of ISWC 2004, pp 698-712
14. Euzenat J, Valtchev P. Similarity-based ontology alignment in OWL-Lite. In proc. of ECAI2004, pp 333-337, Valencia, Spain, August 2004
15. Flanakin M. Web Log. Comments and complaints on software and technology in general. Comparison: Web-based Tracker. 08/08/2005. http://geekswithblogs.net/flanakin/articles/CompareWebTrackers.aspx
16. Jouault F, Kurtev I. Transforming Models with ATL. In proc. of the Model Transformations in Practice Workshop at MoDELS 2005, Montego Bay, Jamaica, pp 128-138
17. Jouault F, Bézivin J. KM3: a DSL for Metamodel Specification. In proc. of 8th FMOODS, LNCS 4037, Bologna, Italy, 2006, pp 171-185
18. Kashyap V, Sheth A P. Semantic and Schematic Similarities Between Database Objects: A Context-Based Approach. VLDB J. 5(4): 276-304, 1996
19. Kedad Z, Xue X. Mapping discovery for XML data integration. In proc. of CoopIS 2005, Agia Napa, Cyprus, November 2005, pp 166-182
20. Kensche D, Quix C, Chatti M A, Jarke M. GeRoMe: A Generic Role Based Metamodel for Model Management. OTM Conferences (2) 2005, pp 1206-1224
21. Lenzerini M. Data Integration: A Theoretical Perspective. In PODS 2002. pp 233-246
22. Maedche A, Motik B, Silva N, Volz R. Mafra - a mapping framework for distributed ontologies. In proc. of EKAW 2002, pp 235-250
23. Mantis Bug Tracking System. Reference site: http://www.mantisbt.org/, 06/2006
24. Melnik, S. Generic Model Management: Concepts and Algorithms, Ph.D. Dissertation, University of Leipzig, Springer LNCS 2967, 2004
25. Melnik S, Bernstein P A, Halevy A, Rahm E. Supporting Executable Mappings in Model Management. In proc. of SIGMOD 2005, Maryland, US, pp 167-178

26. Miller R J, Hernandez M A, Haas L M, Yan L-L, Ho C T H, Fagin R, Popa L. The Clio Project: Managing Heterogeneity. In SIGMOD Record 30, 1, 2001, pp 78–83
27. Milo T, Zohar S. Using Schema Matching to Simplify Heterogeneous Data Translation. In proc. of VLDB 1998, pp 122-133
28. Mitra P, Wiederhold G, Kersten M. A graph-oriented model for articulation of ontology interdependencies. LNCS, 1777:86+, 2000
29. OMG (Object Management Group). Human Usable Textual Notation (HUTN) Specification, Final Adopted Specification. (ptc-02-12-01)
30. Pottinger R A, Bernstein P A. Merging Models Based on Given Correspondences. In proc. of VLDB 2003. Berlin, Germany, pp 862-873
31. Sheth A P, Thacker S, Patel S. Complex relationships and knowledge discovery support in the InfoQuilt system. VLDB Journal. 12(1): 2-27, 2003
32. Shvaiko P, Euzenat J. A Survey of Schema-Based Matching Approaches. Journal of Data Semantics IV: 146-171 (2005)

Reducing the Cost of Validating Mapping Compositions by Exploiting Semantic Relationships

Eduard Dragut[1] and Ramon Lawrence[2]

[1] Department of Computer Science, University of Illinois at Chicago
[2] Department of Computer Science, University of British Columbia Okanagan

Abstract. Defining and composing mappings are fundamental operations required in any data sharing architecture (e.g. data warehouse, data integration). Mapping composition is used to generate new mappings from existing ones and is useful when no direct mapping is available. The complexity of mapping composition depends on the amount of syntactic and semantic information in the mapping. The composition of mappings has proven to be inefficient to compute in many situations unless the mappings are simplified to binary relationships that represent "similarity" between concepts. Our contribution is an algorithm for composing metadata mappings that capture explicit semantics in terms of binary relationships. Our approach allows the hard cases of mapping composition to be detected and semi-automatically resolved, and thus reduces the manual effort required during composition. We demonstrate how the mapping composition algorithm is used to produce a direct mapping between schemas from independently produced schema-to-ontology mappings. An experimental evaluation shows that composing semantic mappings results in a more accurate composition result compared to composing mappings as morphisms.

Keywords: mapping, composition, integration, model management, semantics, metadata.

1 Introduction

The focus of this work is the study of mappings between models. A model is a schema, an ontology, or some other data representation construct. Mappings are designed to relate the information stored in models. There has been a wide variety of model and mapping representations [2,6,7,9,15,19] defined. Each representation has its own benefits and characteristics related to expressability, applicability, and operability. The problem of semi-automatically creating mappings can be divided into two steps. First, a *match* between two schemas specifies semantic correspondences between their elements [18]. These correspondences can be as simple as *inter-schema correspondences* [16] (or *morphisms* [15]) or they can convey certain semantic relationships (e.g. IsA, HasA) [8]. Second, these correspondences are elaborated to generate *instance (data) level mappings* (e.g. using a system such as Clio [16]). In this paper we are concerned with the former, which we refer to as *metadata level mappings*.

Our contribution is an efficient algorithm for composing semantic mappings at the metadata level that is able to focus human attention only on the potential problems areas in the composition result. Previous composition/reusing algorithms [2,4,5,12,15]

R. Meersman, Z. Tari et al. (Eds.): OTM 2006, LNCS 4275, pp. 882–890, 2006.

did not use semantic mappings, and consequently, require the user to validate the entire mapping. We propose a semantic mapping that can be easily expressed and constructed, preserves semantics under composition, complements data level mappings defined using views and it is supported by the state of art matching algorithms [3,4,8,14]. We show that by capturing explicit semantics in mappings, the composition result contains more semantic information, and it is possible to quickly identify problems in the computed mapping that require human intervention. We identify the composition issues both analytically and experimentally.

The organization of this paper is as follows. In Section 2, we overview existing metadata mapping representations and discuss the issues in capturing semantics and performing composition. Our main contribution is in Sections 3 and 4 where we define semantic metadata mappings between models and provide a set of rules for inverting and composing mappings. An experimental evaluation in Section 5 shows that composing semantic mappings, as opposed to morphisms, results in a better mapping composition result. The paper closes with future work and conclusions.

2 Background

Various forms of metadata level mappings have been used to define model management [2] operators and their semantics. In this setting, models are assumed to have the same expressive characteristics as an EER model. *We assume the same definition for models in this work.* Two different mapping representations have been proposed: morphisms [15] and helper models [2,13].

Morphisms. are simple binary relationships between model elements [15,16] that carry loose semantics about the actual relation of the concepts mapped. Formally, given two models, A and B, a mapping map between them is defined as a morphism that consists of a set of pairs $< a, b >$, where $a \in A$ and $b \in B$, which implies that a and b are *similar*. Composition of morphisms assumes similarity relation to be transitive. That is, given a morphism, m_1, between models A and O, and another morphism, m_2, between O and B, and if $< a, o > \in m_1$ and $< o, b > \in m_2$, then a is assumed to be similar to b.

The composition of morphims has several problems. First, any time an m:1 mapping is composed with an 1:n mapping, the composition result is a cross-product consisting of $m \times n$ correspondences [4,5]. Many of these correspondences incorrectly specify a relationship between the elements. Second, an even worse case occurs when there are no direct correspondences between the concepts. For example, there are no direct relationships between {Street1, Street2} and {City, Country, State, Street}. These problems arise due to the lack of an explicit semantic information in the mapping representation.

Mappings with helper models. [2,17] have also been defined for use in model management [2]. In this case, a mapping between two models A and B can be expressed by using a model, map, and two morphisms, one between A and map and the other between map and B. This mapping format allows a set of objects of A to be related to a set of objects in B in complex ways. It uses the elements and the relationships within the helper model to relate *sets of objects* in A and B (see [2] for a comprehensive description). A mapping composition algorithm is presented in [2,17] for use

in model management. This composition algorithm is imperfect as it does not exploit relationships in the helper model to yield a more accurate composition result. Consequently, the algorithm may miss some of the relationships and second it may suggest false relationships.

3 Mapping Representation

We aim to provide a mapping definition that subsumes most of the relationship kinds that the state of the art matching algorithms discover and investigate operations over this mapping definition. A mapping is a metadata level semantic correspondence between elements in different models.

Definition 1. *Given two models A and B, a mapping between them map consists of a set of mapping elements. Each mapping element is a directed, kinded binary relationship between a pair of elements not in the same model. That is, it is a set of triplets $< a, type, b >$, where $a \in A$, $b \in B$, and* type *is the semantic type of the relationship. The values of* type $\in$ {IsA, AKindOf, HasA, PartOf, =, Contains, ContainedBy, Unknown, Complex, NoRel}.

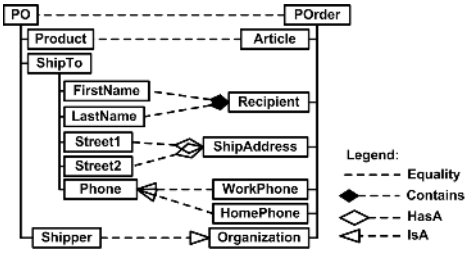


Fig. 1. Mapping example

The mapping is directed from A to B. The relationships used in our mapping representation are well-known except for *Unknown*, *Complex*, and *NoRel* which will be defined later. We assume a set-theoretic framework for defining the semantics of the relations as in [8,19,22]. For instance, a *IsA* b if the content of b contains the content of a, a *HasA* b if the content of b is *part of* the content of a. An example mapping in our representation is in Figure 1. The semantic relationship types are:

- **Equality:** An $Equality$ relationship between two elements a and b, $< a, =, b >$, means that the concept a is equivalent to b.
- **IsA:** An IsA relationship between two elements a and b, $< a, IsA, b >$, means that the concept a is a specialization of b. As this is a binary relation its inverse can be defined as in the algebra of relations [10]. That is, $< b, a > \in IsA^{-1}$ iff $< a, b > \in IsA$. We assign a more descriptive name to the inverse of IsA, i.e. *AKindOf*. Since all remaining relations are binary, a similar approach is assumed for the definition of their inverses.
- **HasA:** A $HasA$ relationship between two elements a and b, $< a, HasA, b >$, means that the concept a has a concept b as part of its representation. The inverse of $HasA$ relationship is $HasA^{-1}$ and its descriptive name is $PartOf$.
- **Contains:** A $Contains$ relationship between two elements a and b, $< a, Contains, b >$, means that the concept a fully encapsulates concept b which cannot independently exist without a. $Contains$ is a stronger relationship than

$HasA$. The inverse of $Contains$ relationship is $Contains^{-1}$ and its descriptive name is $ContainedBy$.

- **Complex:** A $Complex$ relationship between two elements a and b, $< a, Complex, b >$, means that the relationship between concept a and b may require a functional specification: $f(a) = b$. That is, two concepts related by $Complex$ do not have a direct equivalence relationship, but the equivalence relationship can be revealed through a function, whose purpose is to "equalize" the meaning of the two concepts. $Complex$ is considered a Level 2 relationship in [6] and it is the chief subject of the work in the iMap project [3].
- **NoRel:** A $NoRel$ relationship between two elements a and b, $< a, NoRel, b >$, means that there is no relationship between concept a and b. The inverse of the $NoRel$ relationship is still a $NoRel$ relationship (i.e. it is symmetric).
- **Unknown:** An $Unknown$ relationship between two elements a and b, $< a, Unknown, b >$, means that the relationship between concept a and b is not known. $Unknown$ relationships often arise after composition and can be considered the default relationship between all elements until a mapping is defined.

Our mapping relationships capture the semantics of the relationship between model elements like using helper models, but retain the simplicity of inversion and composition given by algebra of relations [20,21]. The usefulness of relationship types besides equality and similarity in specifying mappings is clear from the example in Figure 1. For instance, the IsA relationship clearly captures the semantic relationship between WorkPhone and HomePhone to Phone. It is evident that if morphisms were used then we would have lost the semantic information between these concepts.

4 Invert and Compose Operators

The two fundamental operators required in integration scenarios is the ability to $Invert$ a mapping and the ability to $Compose$ two mappings to produce another mapping. We discuss how these operators are defined in this section.

Invert Operator: The $Invert$ operator has been defined for morphisms [15] and is simply the swapping of the left and right elements of the morphism. Such a simple definition is possible since the similarity relationship is symmetric.

A mapping in our framework consists of a set of directed, kinded, binary relationships. Thus, the semantic type of the relationship must also be inverted when applying the $Invert$ operator. The approach is very similar in spirit with other works that manipulate relations, e.g. [1] in the field of temporal logic.

Definition 2. *Consider two models A and B and a mapping, map, between them. Let* m *be a mapping element whose expression is* $< a, type, b >$*, where type is one of the types defined above. Then its corresponding inverted mapping element, denoted* m^{-1}*, is given by the following expression:* $< b, type^{-1}, a >$*. Moreover, the invert of map, denoted by* map^{-1}*, is defined from B to A and its expression is given by:* $map^{-1} = \{< b, type^{-1}, a > | < a, type, b > \in map\}$.

For example, $< HomePhone, IsA, Phone >^{-1} = < Phone, IsA^{-1}, HomePhone >$.

Table 1. Composition rules

(a,b) (b,c)	=	IsA	IsA^{-1}	Contains	$Contains^{-1}$	HasA	$HasA^{-1}$
=	=	IsA	IsA^{-1}	Contains	$Contains^{-1}$	HasA	$HasA^{-1}$
IsA	IsA	IsA	Unknown	Unknown	Unknown	Unknown	Unknown
IsA^{-1}	IsA^{-1}	Unknown	IsA^{-1}	Contains	$Contains^{-1}$	HasA	$HasA^{-1}$
Contains	Contains	Contains	Unknown	Contains	Unknown	Contains	Unknown
$Contains^{-1}$	$Contains^{-1}$	$Contains^{-1}$	Unknown	Unknown	$Contains^{-1}$	Unknown	$Contains^{-1}$
HasA	HasA	HasA	Unknown	Contains	Unknown	HasA	Unknown
$HasA^{-1}$	$HasA^{-1}$	$HasA^{-1}$	Unknown	Unknown	$Contains^{-1}$	Unknown	$HasA^{-1}$

Compose Operator: Composing two mappings involves defining a composition operation between the elements of the mappings (i.e. triplets of form $< a, type, b >$). The relationship types that constitute the core of a mapping definition have well defined composition properties. Table 1 provides the composition rules that govern our composition technique. This table can be read where a cell that has a row header of type T, a column header of type U, and a cell value of V means that composing the mapping elements $< a, T, b >$ with $< b, U, c >$ results in a composed mapping element of $< a, V, c >$. For instance, composing $< a, =, b >$ with $< b, IsA, c >$ results in a composed mapping element of $< a, IsA, c >$.

Definition 3. *Given two mappings map_{AB} between models A and B and map_{BC} between models B and C the Compose operator, denoted by $\circ$, produces a new mapping $map_{AC} = map_{BC} \circ map_{AB} = map_{BC}(map_{AB})$ between models A and C:*

$$map_{AC} =_{df} \{< a, t, c > | \exists b :< a, t_1, b >\in map_{AB} \wedge < b, t_2, c >\in map_{BC} \wedge$$

$$t = T(t_1, t_2)\}, \text{ where } T(\cdot, \cdot) \text{ is the entry in Table 1}$$

Some of the entries in the table were presented in other works. For instance, the composition between IsA, IsA^{-1}, *Equality* and *NoRel* are presented in [6] and some of the composition outcomes between *IsA*, *Contains* and *HasA* are reported in [17].

There are no entries for the $Unknown$, $Complex$ and $NoRel$ relationships in the table because $Unknown$ composed with any other type gives $Unknown$, $Complex$ composed with $Equality$ preserves $Complex$ and composed with any other type is $Unknown$. We do not explicitly handle $NoRel$ relation because we prefer $Unknown$ over $NoRel$ since it helps us establish desirable indirect relationships, which would have been lost employing $NoRel$. For instance, if we followed the directions of other works that handle relations (e.g. [1, 6]) we would have obtained *No Info*, which is not a desirable outcome. An example (in Figure 2) has convergent IsA relationships (<Shipper, IsA, Organization> and <Organization, IsA^{-1}, Supplier>). For this case it is desirable to retain the information that the two are IsA siblings. The motivation is two fold: (1) it is hard, if not impossible, to define a direct relationship between such concepts and (2) such information might prove valuable if a merge or an alignment [11] operation is performed between the models.

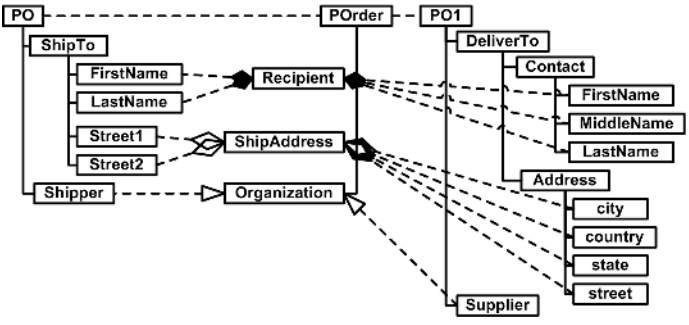


Fig. 2. An example of the problematic cases

Having relationship types that capture more semantics besides the generic *similarity* relationship, we are able to create a composition result with better semantics and avoid the false relationships that simple similarity can produce. Further, the cases where the result is $Unknown$ are especially important as they are direct indicators that the composition of these elements are not well-defined and should be investigated by the user.

The mappings in this system have the following desirable properties:

Proposition 1. *Consider two models A and B and a mapping, map, between them. We denote the invert of map by map^{-1}. The following properties hold:*

1. $b \in (map(map^{-1}))(b)$, $b \in B$,
2. $a \in (map^{-1}(map))(a)$, $a \in A$.

Proposition 2. *Mapping composition is symmetric in our framework, i.e.*

$$map_{AC}^{-1} = (map_{BC} \circ map_{AB})^{-1} = map_{AB}^{-1} \circ map_{BC}^{-1} = map_{CA}.$$

An important feature of the composition algorithm is that it only suggests correct directed, kinded relationships in the composition result. This can be seen by examining the table of composition rules. False results are not produced as transitivity is only applied when it is safe to do so, and the $Compose$ operator uses the $Unknown$ relationship to indicate when it is not possible (in general) to suggest a relationship type given only the information expressed in the two mappings. Thus, $Unknown$ relationship types are valuable as they are the ones that require semi-automatic resolution (e.g. human intervation). Note that we do not claim to retrieve the set of *all* true relationships, but state that the set of relationships we retrieve consists *only* of true relationships.

5 Experimental Evaluation

The experiment has two main goals: to show that our framework is robust when applied to real world application and that we are able to correctly identify problematic cases.

Experiment Setup: We considered five real-world XML schemas in the purchase order domain: CIDR, Excel, Noris, Paragon, and Apertum [14]. They are assigned

Table 2. Mapping distribution

relations	CIDR	Excel	Noris	Paragon	Apertum
equality	18	30	32	36	34
isa	6	11	24	2	13
hasa/part of	14	18	0	10	2
complex	0	0	2	2	2
overlap ratio	0.775	0.84	0.65	0.62	0.61

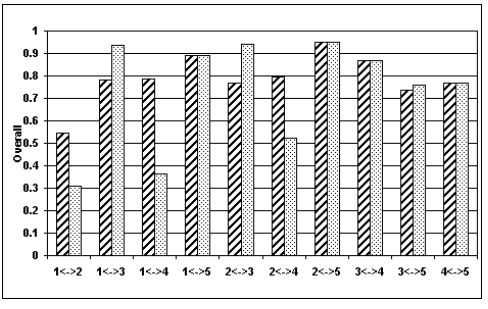


Fig. 3. Schema-to-Schema Mapping Statistics

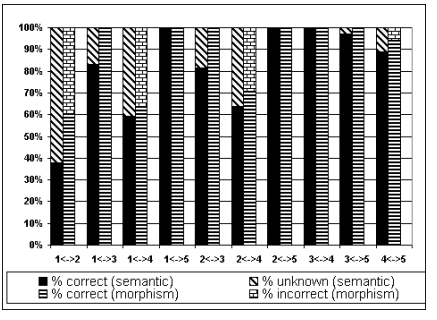


Fig. 4. Algorithm Output Breakdowns

numbers 1, 2, 3, 4, and 5 respectively in order to reference them in our graphs. We also considered a reference ontology to which each XML schema is manually mapped. The reference ontology (see [5]) was created such that it did not have all concepts in the five schemas, such as *unitOfMeasure*, *count*, and *VAT* information. Table 2 shows in the last row the ratio of schema elements that can be mapped to the ontology. These ratios are an useful indicator of the inherent limitation in the performance of the composition because the composition may miss some valid correspondences. Table 2 also shows the mapping broken down in terms of the relationship types defined in this paper for each of the schema-to-ontology mappings.

Measures for composition quality: We measure the performance of our system via three metrics: *Precision*, *Recall*, and *Overall* [4,5,14]. Overall is defined as $Overall = Recall * (2 - 1/Precision)$. We introduce a new metric, called *User Effort*, meant to characterize the post-processing user effort after the composition algorithm is applied.

The Experiment: After we have manually defined the mappings between the five schemas and the ontology we have applied the composition algorithm developed in this work to compute the direct mappings between the schemas. We have also applied composition when the mappings are morphisms (using a natural join). The results are given in terms of the Overall statistic in Figure 3. The first striped bar is our semantic approach, while the second bar is for morphisms. The results are shown for all 10 possible schema-to-schema mappings.

Observe that overall our composition algorithm outperforms composition with morphisms. In five cases, the performances of the two systems are comparable. These cases are an indication that transitivity plays by the rules and both systems are able to

correctly compute the composition. However, in the bad cases, where assuming transitivity is costly and produces many false matches, composing semantic mappings is much better and results in much higher overall performance. For instance, the composed mapping between schemas 1 and 4 produced using our approach has an overall value double that of morphisms. This supports our statement that our composition rules preserve the transitivity of composition as exhibited by morphisms only when it is safe to do so.

Let's consider the two cases when our composition is apparently outperformed, i.e. 1<–>3 and 2<–>3 in Figure 3. In order to understand the behavior of composition in these situations we provide in Figure 4 a detail breakdown of the output of the two composition algorithms. The result of our algorithm is divided into the percentage of the correct and $Unknown$ relationships suggested (as we do not produce incorrect relationships). The morphism output is divided into the percentage of correct and incorrect relationships suggested. All percentages are relative to the total number of output relationships. In these two cases our composition algorithm casts doubts on the set of relationships that later turn out to be true. They are mostly the effect of composing divergent $HasA$ relationships.

Other interesting cases are 1<–>2 and 1<–>4. For these two cases the percentage of the incorrect relationships among the set of all $Unknowns$ is 64% and 89%, respectively. Every time incorrect relationships are likely to be produced during composition our algorithm is clearly better than the composition over morphisms.

User Effort: Define the *User Effort* metric as the ratio of the number of $Unknown$ relationships to the number of all produced relationships. These statistics are represented by the percentage of $Unknown$'s in Figure 4. This measure does not consider the effort required to find the relationships missed by the composition operator. Within the mapping composition frameworks presented in Section 2 (e.g. morphisms) the user effort is at its peak since every proposed relationship needs to be investigated and validated.

In our experimental environment, on average the User Effort is only 19%, and even lower rates of user involvement occur when transitivity can be safely applied and thus no validation is required. The worst two cases that require a substantial user effort are the computation of a direct mapping between CIDR and Excel, and CIDR and Paragon.

6 Future Work and Conclusions

Our contribution is a mapping representation that captures explicit semantics that can be efficiently inverted and composed. Unlike morphisms, semantics are preserved during composition and only correct mapping correspondences are retained. $Unknown$ mappings formed during composition are highlighted for semi-automatic correction by the user. We have shown using experiments that the mappings are efficient to construct and the composition result is much improved as opposed to composing morphisms. This is important as the mappings are simple enough such that it is realistic that they could be semi-automatically generated, but also expressive enough to ensure that composition is well-defined and produces correct results. The composition algorithm is closed under the defined mapping relationships and composition rules and does not produce false

relationships. In addition, we have provided sound arguments that this representation system is able to isolate the hard composition cases, which significantly reduces user effort in finding bad matches and validating the entire mapping produced.

Our future efforts will strive to further confine the cases captured by the $Unknown$ relationship and to find a set of heuristics that would allow us to efficiently suggest relationships between model elements.

References

1. James F. Allen. Maintaining knowledge about temporal intervals. In *Communications of the ACM, v.26 n.11*, pages 832–843, 1983.
2. P. Bernstein. Applying Model Management to Classical Meta Data Problems. In *CIDR*, 2003.
3. R. Dhamankar, Y. Lee, A. Doan, A. Halevy, and P. Domingos. iMAP: Discovering Complex Mappings between Database Schemas. In *SIGMOD Conference 2004*, pages 383–394, 2004.
4. Hong Hai Do and E. Rahm. COMA - A System for Flexible Combination of Schema Matching Approaches. In *VLDB*, pages 610–621, 2002.
5. E. Dragut and R. Lawrence. Composing mappings between schemas using a reference ontology. In *ODBASE*, pages 783–800, 2004.
6. J. Euzenat. An API for Ontology Alignment. In *International Semantic Web Conference*, pages 698–712, 2004.
7. R. Fagin, P. Kolaitis, L. Popa, and W. Tan. Composing schema mappings: Second-order dependencies to the rescue. In *PODS*, pages 83–94, 2004.
8. F. Giunchiglia, P. Shvaiko, and M. Yatskevich. S-Match: an Algorithm and an Implementation of Semantic Matching. *ESWS*, pages 61–75, 2004.
9. A. Halevy. Answering queries using views: A survey. *VLDB Journal*, 10(4):270–294, 2001.
10. Robin Hirsch and Ian Hodkinson. *Relation algebras by games*. North-Holland, Amsterdam, 2002.
11. K. Kotis and G. A. Vouros. The HCONE Approach to Ontology Merging. In *ESWS*, pages 137–151, 2004.
12. J. Madhavan, P. Bernstein, A. Doan, and A. Halvey. Corpus-based Schema Matching. In *ICDE*, 2005.
13. J. Madhavan, P. Bernstein, P. Domingos, and A. Halevy. Representing and reasoning about mappings between domain models. In *AAAI*, pages 80–86, 2002.
14. J. Madhavan, P. Bernstein, and E. Rahm. Generic Schema Matching with Cupid. In *VLDB*, pages 49–58, 2001.
15. S. Melnik, E. Rahm, and P. Bernstein. Rondo: A Programming Platform for Generic Model Management. In *SIGMOD Conference 2003*, pages 193–204, 2003.
16. L. Popa, Y. Velegrakis, R. Miller, M. Hernndez, and R. Fagin. Translating web data. In *VLDB*, pages 598–609, 2002.
17. R. Pottinger and P. Bernstein. Merging Models Based on Given Correspondences. In *VLDB*, pages 826–873, 2003.
18. E. Rahm and P. Bernstein. A survey of approaches to automatic schema matching. *VLDB Journal*, 10(4):334–350, 2001.
19. S. Spaccapietra and C. Parent. View integration: A step forward in solving structural conflicts. *IEEE Trans. Knowl. Data Eng.*, 6(2):258–274, 1994.
20. A. Tarski. On the calculus of relations. *J. Symbolic Logic.*, 6:73–89, 1941.
21. A. Tarski. Some methodological results concerning the calculus of relations. *J. Symbolic Logic.*, 18:188–189, 1953.
22. L. Xu and D. Embley. Discovering Direct and Indirect Matches for Schema Elements. In *DASFAA*, pages 39–46, 2003.

Using Fuzzy Conceptual Graphs to Map Ontologies

David Doussot[1], Patrice Buche[1], Juliette Dibie-Barthélemy[1], and Ollivier Haemmerlé[2]

[1] UMR MIA INA P-G/INRA, Mét@risk MIA INRA
16, rue Claude Bernard, F-75231 Paris Cedex 05
[2] Département de Mathématiques-Informatique
Université Toulouse le Mirail
5, Allées Antonio Machado 31058 Toulouse Cedex 1
{David.Doussot, Patrice.Buche, Juliette.Dibie}@inapg.inra.fr,
Ollivier.Haemmerle@univ-tlse2.fr

Abstract. This paper presents a new ontology mapping method. This method addresses the case in which a non-structured ontology is to be mapped with a structured one. Both ontologies are composed of triplets of the form (object, characteristic, value). Structured means that the values describing the objects according to a given characteristic are hierarchically organized using the *a kind of* relation. The proposed method uses fuzzy conceptual graphs [8] to represent and map objects from a source ontology to a target one. First, we establish a correspondence between characteristics of the source ontology and characteristics of the target ontology based on the comparison of their associated values. Then, we propose an original way of translating the description of an object of the source ontology using characteristics and values of the target ontology. The description thus translated is represented as a fuzzy conceptual graph. Finally, a new projection operation is used to find mappings between translated objects and actual objects of the target ontology. This method has been implemented and the results of an experimentation concerning the mapping of ontologies in the field of risk in food are presented.

1 Introduction

Strong capacities of treatment associated with the availability of huge amounts of data storage make it possible to deal with huge amounts of data. On top of that, network capacities allow one to gather information from many different sources. Nevertheless, those data are generally not or poorly organized. Even if they are, this organization is rarely based on on an ontology standardized expressly for the domain, but on many different ones. That is why a work on ontology mapping has to be done in order to merge data coming from different sources and based on different ontologies.

Much work has already been performed on ontology mapping [7]: the existing mapping algorithms generally deal with well-structured ontologies and their

R. Meersman, Z. Tari et al. (Eds.): OTM 2006, LNCS 4275, pp. 891–900, 2006.

goal is to align those structures. The structure is usually represented by a set of hierarchically organized classes containing attributes. The instances of classes provide values for those attributes. We can distinguish two families of mapping algorithms. The first family of algorithms tries to establish a correspondence between classes and then between attributes of both ontologies [1,3]: this correspondence is generally based on linguistic similarities of class and attribute names [11] combined with heuristics. The second family tries to find common instances in both ontologies and to deduce a correspondence between classes [2,10].

In this paper, we want to address the mapping process of a non-structured ontology noted O with a structured one noted R and considered as a reference. There is no class categorization, but each ontology is composed of triplets of the form (object, characteristic, value) and the number of triplets is different for each object described in the ontology. We say that an ontology is structured if the values of a given characteristic describing an object are organized according to the *a kind of* partial value function. As one of the two ontologies is not structured, the different methods proposed in the bibliography are not suitable. It is the reason why we propose to use fuzzy conceptual graphs. This choice has been mainly motivated by: (i) the support of the conceptual graph model which is well adapted to the representation of the taxonomies of ontology R, (ii) the projection operation which takes into account the specialization relation between concepts (mainly used in this paper to compare values of the ontologies), (iii) the fuzzy extension which permits to represent similarities between objects (resp. between values) of both ontologies. In the first step of the processing, we establish a correspondence between characteristics of both ontologies. Then we use fuzzy conceptual graphs to represent objects, to establish correspondences and to calculate scores of similarity between objects of both ontologies.

In section 2, we recall some backgrounds about the fuzzy set theory and fuzzy conceptual graphs. In section 3, we present the terminological matching technique we use, based on strings structure similarity to map values. In section 4, we propose a new method based on fuzzy conceptual graph operations to map objects of both ontologies. Then in section 5, we present first experimentations.

2 Backgrounds

2.1 Fuzzy Set Theory

In this article, we use the representation of fuzzy sets proposed in [12,13].

Definition 1. A *fuzzy set* f on a definition domain $Dom(f)$ is defined by a membership function μ_f from $Dom(f)$ to $[0, 1]$ that associates the degree to which x belongs to f with each element x of $Dom(f)$.

In this paper, the fuzzy set formalism is used to represent similarities, one of the three semantics usually associated with fuzzy sets [4]. More precisely in our approach, a fuzzy set represents the similarities between one value belonging to

ontology O and a list of values belonging to ontology R. Values in R are organized according to the *a kind of* relation. A part of the taxonomy of values we have used in our experimentation is presented in figure 1. The fuzzy set of figure 2 represents the similarities between the value *Cheese: pressed-curd* of ontology O and values of ontology R. The calculus of the membership function of such a fuzzy set is presented in section 3.

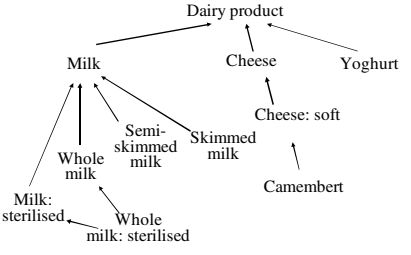


Fig. 1. A part of the taxonomy of values belonging to ontology R used in our experimentation: arrows represent the *a kind of* relation

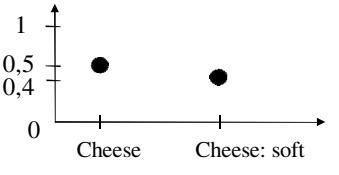


Fig. 2. A fuzzy set noted (0.5/Cheese + 0.4/Cheese: soft) representing similarities of values belonging to ontology R with the value *Cheese: pressed-curd* of ontology O

2.2 Fuzzy Conceptual Graphs

The conceptual graph model we use is based on the formalization presented in [9]. In the conceptual graph model, conceptual graphs are defined using a predefined vocabulary contained in the support. The support is notably composed of a concept type set, a set of markers and a relation type set. The concept type set defines the types of objects, characteristics, values, events and states which have to be represented. The concept type set is partially ordered according to the "a kind of" relation. A part of the concept type set we have used in our experimentation is presented in figure 1. An individual marker represents an instance of a concept. An unspecified instance of a concept is represented by the generic marker, noted $*$. The relation type set defines the types of semantic links which can be represented between the concepts.

We have proposed an extension of the conceptual graph model to represent fuzzy values in [9]. We only remind that extension through an example. A fuzzy set can appear in two ways in a concept vertex: (i) as a concept with a fuzzy type, the type being a fuzzy set defined on a subset of the concept type set;

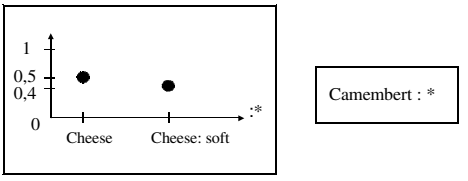


Fig. 3. Two examples of vertices: the left one is a vertex with a fuzzy type, the right one is a classic vertex with a crisp type

(ii) as a concept with a fuzzy marker, the marker being a fuzzy set defined on the set of individual markers. In this paper, we only use concepts with a fuzzy type. One example of vertex with a fuzzy type is given in figure 3. In [9], we proposed different kinds of comparisons between fuzzy conceptual graphs. A mandatory preliminary step to perform these comparisons is that the fuzzy types possibly included in the fuzzy conceptual graphs are all defined in the same definition domain: the concept type set. It is the reason why we have introduced the notion of *fuzzy type closure* in [9]. Intuitively, the degrees associated with the concept types which belong to the fuzzy type are propagated to more specific values of the concept type set. The value 0 is associated with the other concept types of the concept type set. An example of fuzzy type closure corresponding to the fuzzy type of figure 3 is presented in figure 4. In this paper, we need to compare a fuzzy conceptual graph representing a fuzzy knowledge to a crisp (ie not fuzzy) conceptual graph representing a crisp knowledge. We use the following comparison operation for that purpose.

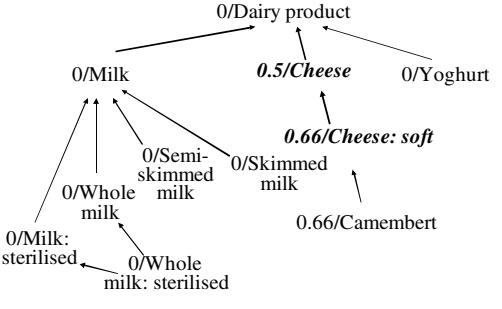


Fig. 4. The closure of the fuzzy type presented in figure 2: concept types belonging to the fuzzy type and their associated degree appear in italic bold

Definition 2. *A δ-projection from a fuzzy conceptual graph G into a crisp conceptual graph G' is a triple (g, h, δ), g (resp. h) being a mapping from the set of concept (resp. relation) vertices of G to the set of concept (resp. relation) vertices of G' such that: (i) the edges and their numbering are preserved; (ii) $\forall$ concept $c_i \in G$, $i \in [1, \ldots, n]$, labeled $t_i : m_i$, t_i being a fuzzy type, c_i is mapped with its image $g(c_i) \in G'$ labeled $t'_i : m'_i$, t'_i being a crisp type, with a satisfaction degree*

$\delta_i = \mu_{clos(t_i)}(t'_i)$, $\mu_{clos(t_i)}$ *being the membership function of the fuzzy type closure of* t_i. *The satisfaction degree of* G *by* G', *noted* δ, *is defined by* $\delta = min_{i=1,\ldots,n}\delta_i$.

Example 1. *According to definition 2, there is a 0.4-projection from the conceptual graph restricted to the fuzzy concept vertex of figure 3 into the conceptual graph restricted to the crisp concept vertex of figure 3. This projection has been obtained thanks to the fuzzy type closure given in 4 which propagates the degree 0.4 from the type* $Cheese : soft$ *to the more specific type* $Camembert$.

3 A Syntactic Relevance Score for Fuzzy Matching of Values

In order to compare ontology O with ontology R, we propose to compute a relevance score between each value of O with values of R. For that, we chose the *Dice* coefficient based on the words common to both values (see [5] for a review of similarity measures).

Definition 3. *Let* V=$\{v_1, \ldots, v_n\}$ *and* W=$\{w_1, \ldots, w_p\}$ *be the sets of lemmatized words respectively of a value of ontology O and a value of ontology R. The relevance score between* V *and* W *is defined as their Dice coefficient:*

$$relevance(V, W) = 2 * \frac{|V \cap W|}{|V| + |W|}$$

A set of values $\{W_1, \ldots, W_q\}$ of ontology R can thus be associated with a given value V of ontology O, weighted by their syntactic closeness to the value V. We propose to represent such a set by a discrete fuzzy set defined on $\{W_1, \ldots, W_q\}$ whose membership function is the relevance score between the value V and each value in the set $\{W_1, \ldots, W_q\}$. Note that for a given value V of ontology O, the set of associated values belonging to ontology R can be empty.

Example 2. *Let* **fresh fish** *be an object of ontology O. The list of couples (characteristic : value) associated with* **fresh fish** *is:*

- *presentation: whole*
- *which fish ? cod*

Let **cod, raw** *be an object of ontology R. The list of couples (characteristic : value) associated with* **cod, raw** *is:*

- *product type: seafood or seafood product*
- *origin of main ingredient: cod or codfish*
- *part of plant or animal: skeletal meat part, without bone or shell*
- *physical state, shape: whole, shape solid*
- *preservation method: preserved by refrigeration or freezing*

relevance(whole, whole shape solid)=0.5 and relevance(cod, cod codfish)=0.66 are two examples of comparisons between a value of ontology O and a value of ontology R.

This syntactic mapping between values of ontology O with values of ontology R allows the identification of correspondences between characteristics of O and characteristics of R. Those correspondences are refined manually such that each characteristic of O corresponds to at most one characteristic of R. When a characteristic of O has no correspondence with a characteristic of R, such a characteristic could be interpreted as no meaningful in ontology R.

Example 3. *According to the mapping between values of example 2, the characteristic "presentation" of ontology O is associated with the characteristic "physical state, shape" of ontology R and the characteristic "which fish ?" of ontology O is associated with the characteristic "origin of main ingredient" of ontology R.*

So, at the end of the first step we have identified a set of *linked values* of O with values of R and a set of *linked characteristics* of O with characteristics of R as defined below.

Definition 4. We call *linked values* of ontology O, noted $LV_{\mathcal{O}}$, the set of values of O such that each of them is associated with a set of values of ontology R with a given relevance score, represented by a discrete fuzzy set. We call *linked characteristics* of ontology O, noted $LC_{\mathcal{O}}$, the set of characteristics of O such that each of them is associated with one characteristic of ontology R.

4 Using Conceptual Graphs for Fuzzy Matching of Objects

We are now interested in comparing objects of ontology O with objects of ontology R. We have chosen the conceptual graph model in order to represent and to compare objects. In order to compare objects of ontology O with objects of ontology R, we would like to use the projection operation. But the objects of ontology O are not defined with the same vocabulary as the objects of ontology R. So, according to the results of the previous section, we propose to define each object of ontology O with characteristics and values of ontology R, such that all the objects are expressed with the same vocabulary, that of the reference ontology R.

Since the support of the conceptual graph model contains the ground vocabulary, we first present how the support is built from ontologies O and R. The concept type set is composed of the set of objects of ontology O, the set of objects of ontology R, the set of characteristics of ontology R and the hierarchized[1] set of values of ontology R . The relation type set is composed of the two relation types *HasForValue* and *HasForCharac*. The set of markers is limited to the generic marker. Each object of ontology R can be represented by a conceptual graph as defined below.

[1] According to the *a kind of* relation.

Definition 5. Each object X of ontology R can be represented by the conceptual graph, noted G_X, where the concept vertices of types $Charac_1$, ..., $Charac_n$ represent the characteristics of the object X and the concept vertices of types $Value_1^1$, ..., $Value_{p_1}^1$, ..., $Value_1^n$, ..., $Value_{p_n}^n$ their corresponding values, one characteristic being able of having several values.

Example 4. *Figure 5 presents an example of conceptual graph G_X. It is associated with the object "cod, raw" of ontology R presented in example 2.*

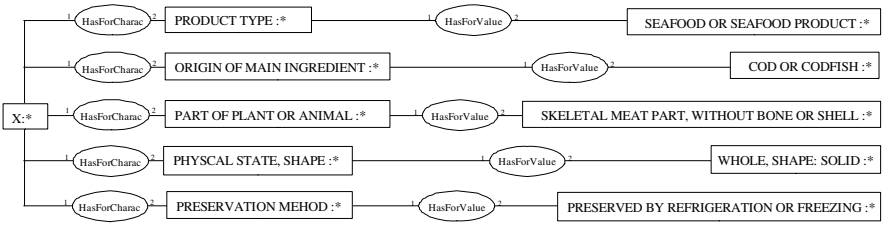


Fig. 5. The conceptual graph G_X associated with the object *Cod, raw* of ontology R

Each object Y of ontology O can be represented by a *translated conceptual graph* which corresponds to the description of the object Y in ontology R. More precisely, for a given object Y, each characteristic and its associated values, belonging respectively to the sets $LC_{\mathcal{O}}$ and $LV_{\mathcal{O}}$ (see definition 4) of O are translated in their corresponding characteristic and values in R.

Definition 6. Let f be the fuzzy value function which associates each value of $LV_{\mathcal{O}}$ with its corresponding values in R associated with their relevance score. Let g be the value function which associates each characteristic of $LC_{\mathcal{O}}$ with its corresponding characteristic in R. Let $Charac_1$, ..., $Charac_m$ be the list of characteristics of the object Y of ontology O belonging to $LC_{\mathcal{O}}$. Let $Value_1^1$, ..., $Value_{p_1}^1$, ..., $Value_1^m$, ..., $Value_{p_m}^m$ be the list of values associated with the characteristics $Charac_1$, ..., $Charac_m$ of the object Y and belonging to $LV_{\mathcal{O}}$. Each object Y of ontology O can be represented by the *translated conceptual graph*, noted G_Y^T, where the concept vertices of types $g(Charac_1)$, ..., $g(Charac_m)$ represent the characteristics associated with the object Y in ontology R and the concept vertices of types $f(Value_1^1)$, ..., $f(Value_{p_1}^1)$, ..., $f(Value_1^m)$, ..., $f(Value_{p_m}^m)$ their corresponding fuzzy values in R. Each fuzzy value is represented by means of a concept vertex with a fuzzy type as presented in section 2.2.

Example 5. *Figure 6 presents an example of translated conceptual graph G_Y^T. It is associated with the object "fresh fish" of ontology O presented in example 2 using the mappings of characteristics given in example 3.*

Since the objects of ontologies O and R are represented by comparable conceptual graphs using the same vocabulary, we can now study their correspondences. The identification of correspondences between an object of ontology O with objects

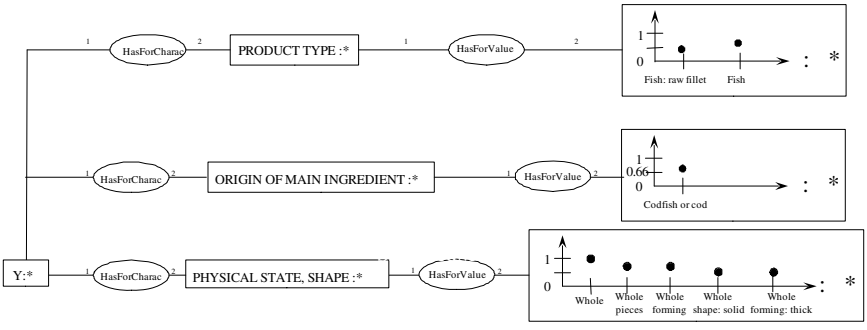


Fig. 6. The translated conceptual graph G_Y^T associated with the object *fresh fish* of ontology O

of ontology R rests on the δ-projection the definition of which has been given in definition 2. It is a flexible operation of mapping between a fuzzy conceptual graph representing a fuzzy knowledge and a crisp conceptual graph representing a precise knowledge. The fuzzy knowledge is the translated conceptual graph G_Y^T corresponding to the description of an object Y of ontology O in the vocabulary of ontology R. The crisp conceptual graph is G_X, the conceptual graph associated with an object X of ontology R which has to be compared with Y. In order to avoid empty answers, we do not project the entire translated conceptual graph G_Y^T into G_X. Subgraphs corresponding to each couple (characteristic, value) of G_Y^T are extracted. We say that an object X of ontology R is a *candidate* for an object Y of ontology O if there exists δ-projections from subgraphs of G_Y^T into G_X. Two parameters characterize the relation between the objects X and Y: an adequation degree, which reflects the similarity between values characterizing the objects, and a cover score which indicates the proportion of common characteristics of both objects.

Definition 7. Let G_Y^T be the translated conceptual graph associated with an object Y of ontology O as defined in definition 6. Let $G_Y^T(i,j)$, $i \in [1,m]$, $j \in [1,q_i]$, be a subgraph of G_Y^T composed of the concept vertices of types $Charac_i$ and $Value_{q_i}^i$ linked by the relation vertex *HasForValue*. Let $\mathcal{KB}$ be the knowledge base composed of the conceptual graphs associated with all the objects of ontology R as defined in definition 5. An object X of ontology R is a *candidate* for the object Y iff there exists a δ-projection from at least one subgraph $G_Y^T(i,j)$ of G_Y^T into G_X of $\mathcal{KB}$. The adequation degree, noted AD, of the object X to the object Y is $AD{=}min_{i\in[1,l],j\in[1,r_i]}(\delta_{ij})$, where $G_Y^T(i,j)$, $i \in [1,l]$, $l \leq m$, $j \in [1,r_i]$, $r_i \leq q_i$, are the subgraphs of G_Y^T such that there exists a δ-projection from each of them into G_X with a satisfaction degree δ_{ij}. The cover score, noted CS, of the object X to the object Y is: $CS = \frac{2n_3}{n_1+n_2}$, n_1 (resp. n_2) being the number of characteristics of G_Y^T (resp. G_X), n_3 being the number of subgraphs of G_Y^T such that there exists a δ-projection from each of them into G_X.

Example 6. *According to definition 7, the object "Cod, raw" of ontology R, presented as a conceptual graph in figure 5, is a candidate for the object "fresh fish" of ontology O, presented as a translated conceptual graph in figure 6, with the adequation degree* $AD = 0.5$ *and the cover score* $CS = 0.44$.

5 Preliminary Experimentations

We have used the exposed method to map two ontologies used to index databases concerning food products. On the one hand, an ontology called CONTA is used as a reference ontology R. Its objects represent food products. They are described using Langual, an international food description system [6] composed of fourteen predefined characteristics and predefined associated values organized as a taxonomy using the *a kind of* relationship. The food product *cod, raw* presented in example 2 belongs to the CONTA ontology. The CONTA ontology is used in a chemical contamination database to store levels of contamination of a given instance of a food product by a given contaminant. On the other hand, an ontology called CONSO, provided by a private firm, is used as an ontology O. Its objects also represent food products. They are described with a varying number of characteristics depending on the food product. A predefined set of "flat" predefined values is associated with each characteristic. The food product *fresh fish* presented in example 2 belongs to the CONSO ontology. The CONSO ontology is used in a set of data providing household purchases of food products. The CONSO ontology contains 420 product names whereas the CONTA ontology contains about 2500 product names. Upon the 420 product names belonging to the CONSO ontology, only 250 have been associated with product names belonging to the CONTA ontology by an expert manual mapping.

We consider that an object belonging to the CONSO ontology is correctly mapped if more than 50% of the associated mappings in the CONTA ontology, ordered by best scores, are correct. Each object belonging to the CONSO ontology is mapped in average with 30 objects belonging to the CONTA ontology. Our first experimentation gives 203 mappings including 78 errors. In the classical meaning of information retrieval, we can then conclude that our experimentation has a recall of 50% and a precision of 61%. 20% of good results have been obtained thanks to the projection operation of the conceptual graph model which takes into account specializations of concepts.

6 Conclusion

In this paper, we have proposed a new method to map objects belonging to two different ontologies. The main originality of this method is to compute an approximate comparison between the descriptions of objects using fuzzy conceptual graphs. Consequently, it provides, for a given object of a source ontology O, a list of ranked objects of a target ontology R, candidates for the mapping. This approximate comparison is based on the description of objects available in the form of triplets (object, characteristic, value). This format conforms to the

representation of ontologies in the standards of the semantic web. A preliminary experimentation has been done and permits a first encouraging evaluation of the method. Several perspectives of enhancement will be studied in the near future. For instance, the relevance of the matching score can be enhanced if we take into account the fact that all the characteristics do not have the same importance to describe an object.

References

1. D. Aumueller, H. Do, S. Massmann, and E. Rahm, *Schema and ontology matching with coma++.*, SIGMOD Conference, 2005, pp. 906–908.
2. A. Doan, J. Madhavan, P. Domingos, and A. Y. Halevy, *Learning to map between ontologies on the semantic web.*, WWW, 2002, pp. 662–673.
3. E. Dragut and R. Lawrence, *Domain independent learning of ontology mappings*, International Conference on Ontologies, 2004, pp. 783–800.
4. D. Dubois and H. Prade, *The three semantics of fuzzy sets*, Fuzzy Sets and Systems **90** (1997), 141–150.
5. L. Egghe and C. Michel, *Strong similarity measures for ordered sets of documents in information retrieval*, Information Processing and Management **38** (2002), 823–848.
6. J. D. Ireland and A. Moller, *Review of international food classification and description*, Journal of food composition and analysis **13** (2000), 529–538.
7. Y. Kalfoglou and W. M. Schorlemmer, *Ontology mapping: The state of the art.*, Semantic Interoperability and Integration, 2005.
8. J.F. Sowa, *Conceptual structures: information processing in mind and machine*, Addison Wesley Publishing Company, 1984.
9. R. Thomopoulos, P. Buche, and O. Haemmerlé, *Different kinds of comparisons between fuzzy conceptual graphs*, Proceedings of the 11th ICCS, LNAI #2746 (Dresden, Germany), Springer, July 2003, pp. 54–68.
10. F. Wiesman and N. Roos, *Domain independent learning of ontology mappings*, AAMAS, 2004, pp. 844–849.
11. G. Ming Y. Yang, L. Shanping and H. Sailong, *A comparative study on three ontology mapping methods in the integration of product knowledge*, WCICA, 2004, pp. 3093–3097.
12. L. Zadeh, *Fuzzy sets*, Information and control **8** (1965), 338–353.
13. ———, *Fuzzy sets as a basis for a theory of possibility*, Fuzzy Sets and Systems **1** (1978), 3–28.

Formalism-Independent Specification of Ontology Mappings – A Metamodeling Approach

Saartje Brockmans[1], Peter Haase[1], and Heiner Stuckenschmidt[2]

[1] AIFB, Universität Karlsruhe (TH), Germany
[2] University of Mannheim, Germany

Abstract. Recently, the advantages of metamodeling for the graphical specification of ontologies have been recognized by the semantic web community. This has lead to a number of activities concerned with the development of graphical modeling approaches for the Web Ontology Language based on the Meta Object Facility (MOF) and the Unified Modeling Language (UML). An aspect that has not been addressed so far is the need to specify mappings between heterogenous ontologies. With an increasing number of ontologies being available, the problem of specifying mappings is becoming more important and the rationales for providing model based graphical modeling support for mappings is the same as for the ontologies themselves. In this paper, we therefore propose a MOF-based metamodel for mappings between OWL DL ontologies.

1 Motivation

Initially, ontologies have been introduced as a solution for the problem of semantic heterogeneity and as a facilitator of semantic integration. With the increasing use of ontologies, however, it has turned out that the problem of semantic integration has only been lifted to a different level of abstraction at which different ontologies describing the same domain have to be aligned. There are two main lines of research addressing the problem of ontology alignment. The first line is concerned with the development of methods for identifying semantic relations between elements in different ontologies. The second line of research is concerned with formalisms for encoding and using semantic relations (mappings) between ontologies. These formalisms are often based on non-standard extensions of the logics used to encode the ontologies. Examples of such mapping formalisms are [1,3,4,5]. In a recent comparison of these approaches, it has been shown that these approaches are mostly orthogonal in terms of assumptions made about the right interpretation of mapping relations [6]. This means that the approaches cover a large variety of possible interpretations of semantic relations, but it also means that they are incompatible with each other and that the choice of a particular formalism is an important decision with significant influence on remaining options for interpreting and using mappings. Further, making the right decision with respect to a mapping formalism requires in depth knowledge of the corresponding logics and the hidden assumptions made as well as the specific

R. Meersman, Z. Tari et al. (Eds.): OTM 2006, LNCS 4275, pp. 901–908, 2006.

needs of the application. In order to make an informed decision about which mapping formalism to use, this decision should be made as late as possible in the modeling process because it is often not possible to decide whether a given mapping formalism is suitable for specifying all relevant connections. Therefore, mappings should first be specified on a purely informal level by just marking parts of the ontologies that are somehow semantically related. In a next step, the kind of semantic relation that exists between the elements should be specified. In order to support this process, we need a formalism-independent format for specifying mappings. On the other hand, we have to make sure that concrete mapping representations can be derived automatically from this model in order to support the implementation and use of the mappings. In order to meet these requirements, we propose a metamodel based approach to specifying ontology mappings independent on the concrete mapping formalism. In particular, we propose a Meta Object Facility-based metamodel for describing mappings between OWL DL ontologies as well as a UML profile that defines a graphical format for mapping modeling. When building the metamodel there is a natural trade-off between coverage and precision of the metamodel: In this paper, we focus on approaches that connect description logic based ontologies where mappings are specified in terms of logical axioms. This allows us to be more precise with respect to the nature and properties of mappings. At the same time, we cover a number of relevant mapping approaches that have been developed that satisfy these requirements, including the approaches mentioned in [6].

2 Ontology Mapping Formalisms

In a recent discussion on the nature of ontology mappings, some general aspects of mapping approaches have been identified [7]. We briefly discuss these aspects in the following and clarify our view on mappings that is reflected in the proposed metamodel with respect to these aspects.

What do mappings define ? In this paper, we restrict our attention to declarative mapping specifications. In particular, we see mappings as axioms that define a semantic relation between elements in different ontologies. Most common are the following kinds of semantic relations:

Equivalence ($\equiv$). Equivalence states that the connected elements represent the same aspect of the real world according to some equivalence criteria. A very strong form of equivalence is equality, if the connected elements represent exactly the same object.

Containment ($\sqsubseteq, \sqsupseteq$). Containment states that the element in one ontology represents a more specific aspect of the world than the element in the other ontology. Depending on which of the elements is more specific, the containment relation is defined in the one or in the other direction.

Overlap (o). Overlap states that the connected elements represent different aspects of the world, but have an overlap in some respect. In particular, it

states that some objects described by the element in the one ontology may also be described by the connected element in the other ontology.

In some approaches, these relations are supplemented by their negative counterparts. The corresponding relations can be used to describe that two elements are *not* equivalent ($\not\equiv$), *not* contained in each other ($\not\sqsubseteq$) or *not* overlapping or disjoint respectively ($\not\circ$). Adding these negative versions of the relations leaves us with eight semantic relations to cover all existing proposals for mapping languages.

In addition to the type of semantic relation, an important distinction is whether the mappings are to be interpreted as extensional or as intensional relationships: In *extensional* mapping definitions, the semantic relations are interpreted as set-relations between the sets of objects represented by elements in the ontologies. In the case of *intensional* mappings, the semantic relations relate the elements directly, i.e. considering the properties of the element itself.

What are the formal properties of mappings ? There are a number of verifiable formal properties that mappings can be required to satisfy. Examples of such formal properties are the satisfiability of the overall model, the preservation of possible inferences or the preservation of answers to queries. The question of what is preserved by a mapping is tightly connected to the hidden assumptions made by different mapping formalisms. A number of important assumptions that influence this aspect have been identified and formalized in [6]. The assumptions identified in the referred paper are:

- The naming of instances (are instances with the same name assumed to denote the same object)
- The way inconsistency affects the overall system (does an inconsistency in one ontology also cause the mapped ones to become inconsistent)
- The assumptions about the relationships between the mapped domains (where with the *global domain assumption* both ontologies describe exactly the same set of objects, while with the *local domain assumption* the sets of objects may also be completely disjoint or overlap each other)

In [6] it has been shown that the differences between existing proposals of mapping languages for description logics can completely be described in terms of the kinds of semantic relations than can be defined and the assumptions mentioned above. This means that including these aspects in the metamodel ensures that we can model all currently existing mapping approaches and that we are able to distinguish them based on specifications that instantiate the metamodel.

What do mappings connect ?. In the context of this work, we decided to focus on mappings between ontologies represented in OWL DL. This restriction makes it much easier to deal with this aspect of ontology mappings as we can refer to the corresponding metamodel for OWL DL specified in [2]. In particular, the metamodel contains the class `OntologyElement`, that represents an arbitrary part of an ontology specification. While this already covers many of the existing mapping approaches, there are a number of proposals for mapping languages that rely on

the idea of view-based mappings and use semantic relations between queries to connect models, which leads to a considerably increased expressiveness.

How are mappings organized ? The final question is how mappings are organized. They can either be part of a given model or be specified independently. Mappings can be uni- or bidirectional. In this work, we use a mapping architecture that has the greatest level of generality in the sense that other architectures can be simulated. In particular, a mapping is a set of mapping assertions that consist of a semantic relation between elements in different ontologies. Further mappings are first-class objects that exist independent of the ontologies. Mappings are directed and there can be more than one mapping between two ontologies. These choices leave us with a lot of freedom for defining and using mappings.

3 A Metamodel for Ontology Mappings

We propose a formalism-independent metamodel covering OWL ontology mappings as described in Section 2. The metamodel is a consistent extension of our earlier work on metamodels for OWL DL ontologies and SWRL rules [2]. It has constraints defined in OCL [9] as well, which we omit here due to lack of space and instead refer to [2] for a complete reference. Figure 1 shows the metamodel for mappings. In the figures, darker grey classes denote classes from the metamodels of OWL DL and rule extensions. The central class in the metamodel is

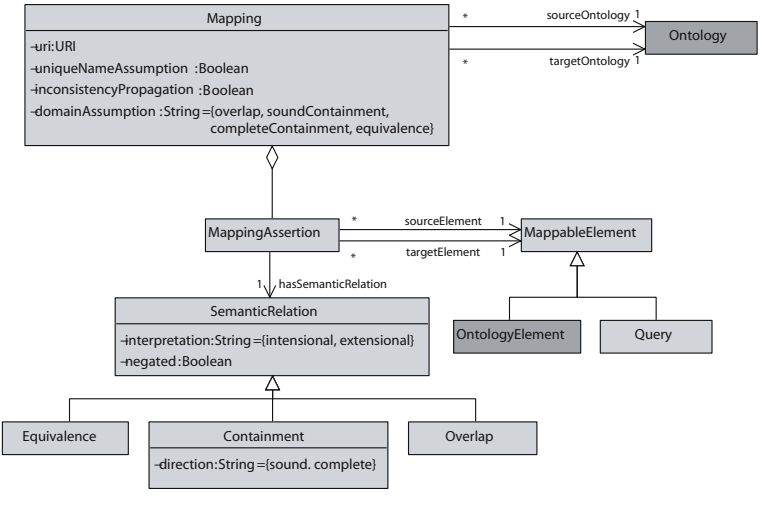


Fig. 1. Metamodel for ontology mappings

the class `Mapping` with four attributes. The URI, defined by the attribute `uri`, allows to uniquely identify a mapping and refer to it as a first-class object. A mapping is always defined between two ontologies. An ontology is represented by

the class `Ontology` in the OWL DL metamodel. Two associations from `Mapping` to `Ontology`, `sourceOntology` and `targetOntology`, specify the source respectively the target ontology of the mapping. Cardinalities on both associations denote that to each `Mapping` instantiation, there is exactly one `Ontology` connected as source and one as target. A mapping consists of a set of mapping assertions, denoted by the MOF aggregation relationship between the two classes `Mapping` and `MappingAssertion`. The elements that are mapped in a `MappingAssertion` are defined by the class `MappableElement`. A `MappingAssertion` is defined through exactly one `SemanticRelation`, one source `MappableElement` and one target `MappableElement`. This is defined through the three associations starting from `MappingAssertion` and their cardinalities. We defined four semantic relations along with their logical negation to be defined in the metamodel. Two of these relationship types are directly contained in the metamodel through the subclasses `Equivalence` and `Overlap` of the class `SemanticRelation`. The other two, containment in either direction, are defined through the subclass `Containment` and its additional attribute `direction`, which can be `sound` ($\sqsubseteq$) or `complete` ($\sqsupseteq$). The negated versions of all semantic relations are specified through the boolean attribute `negated` of the class `SemanticRelation`. For example, a negated `Overlaps` relation specifies the disjointness of two elements. The other attribute of `SemanticRelation`, `interpretation`, defines whether the mapping assertion is assumed to be interpreted intensionally or extensionally. Please note that the metamodel in principle supports all semantic relations for all mappable elements, including individuals.

A mapping assertion can connect two mappable elements, which may be ontology elements or queries. To support this, `MappableElement` has two subclasses `OntologyElement` and `Query`. The former is previously defined in the OWL DL metamodel. The class `Query` reuses constructs from the SWRL metamodel. The reason for reusing large parts of the rule metamodel lies in the fact that conceptually, rules and queries are of very similar nature [8]: A rule consists of a rule body (antecedent) and rule head (consequent), both of which are conjunctions of logical atoms. A query can be considered as a special kind of rule with an empty head. The distinguished variables specify the variables that are returned by the query. Informally, the answer to a query consists of all variable bindings for which the grounded rule body is logically implied by the ontology.

4 A UML Profile for Ontology Mappings

The UML profile mechanism is an extension mechanism to tailor UML to specific application areas. Our proposed UML profile defines a visual notation for optimally supporting the specification of OWL ontology mappings. This visual syntax is based on the metamodel and is independent of a concrete mapping formalism. Mappings in both directions between the metamodel and the profile have to be established. The profile is a consistent extension of our earlier work on a profile for OWL DL ontologies and SWRL rules [2]. Figures 2 and 3 show two ontologies of a domain, depicted using the UML profile for OWL DL.

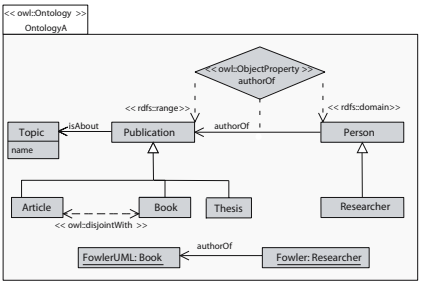


Fig. 2. A First Sample Ontology Depicted using the UML Profile for the Ontology Metamodel

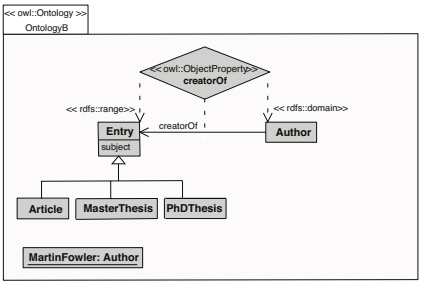


Fig. 3. A Second Sample Ontology Depicted using the UML Profile for the Ontology Metamodel

Fig. 4. Sample containment relation between two concepts

Our goal is to allow the user to specify mappings without having decided yet on a specific mapping language or even on a specific semantic relation. This is reflected in the proposed visual syntax which is, like the metamodel, independent from a concrete mapping formalism. In Figure 4, a source concept `Publication` is defined to be more specific than the target concept `Entry`. Both source and target elements of mapping assertions are represented in a box, connected to each other via a dependency with the corresponding symbol of the semantic relation. In the first step of the process, when users just mark elements being semantically related without specifying the type of semantic relation, the dependency does not carry any relation symbol. Stereotypes in the two boxes denote source- and target ontology. Like defined in the metamodel, these mapped elements can be any element of an ontology (metaclass `OntologyElement`) or a query (metaclass `Query`). They are represented like defined in the UML profile for OWL and rules. The parts of the mappable elements which are effectively being mapped to each other, are denoted via a double-lined box, which becomes relevant if the mapped elements are more complex constructs, as explained in the following.

A more complex example mapping assertion is pictured in Figure 5. The example defines that the union of the classes `PhDThesis` and `MasterThesis`, is equivalent to the class `Thesis`.

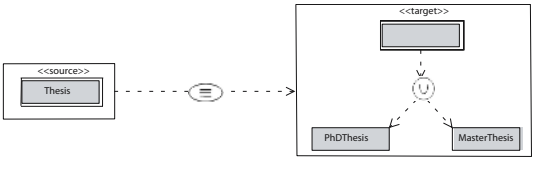


Fig. 5. Sample equivalence relation between complex class descriptions

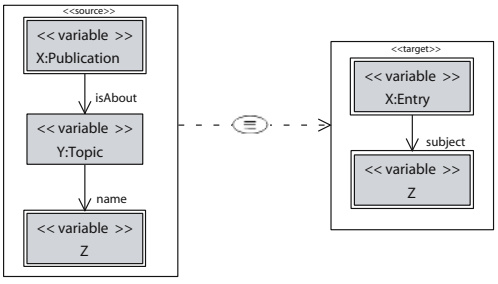


Fig. 6. Sample equivalence relation between two queries

Figure 6 shows an example of an equivalence relation between two queries. The first query is about a Publication X with a Topic Y named Z. The target query is about an Entry X with subject Z. The mapping assertion defines the two queries to be equivalent. The effective correspondences are established between the two distinguished variables X and Z, again denoted with a double-lined box.

5 Discussion

We have presented a MOF based metamodel as well as a UML profile to support formalism independent graphical modeling of mappings between OWL ontologies. The metamodel ties in with previous work on similar metamodels for OWL DL and rule extensions. We considered an abstract metamodel that was designed to cover a range of existing formalisms for specifying mappings, however, the modular approach allows to extend the metamodel for additional constructs or characteristics, or for additional formalisms in a straight-forward manner. While the work presented addresses an important gap in the existing modeling infrastructure, it has to be seen as the basis for a more complete framework for mapping modeling based on MOF and UML. In order to be able to provide support not only for the acquisition of mappings but also for their implementation in one of the existing formalisms, three additional steps have to be taken. In a first step, we have to link the abstract metamodel presented in this paper to concrete mapping formalisms. This can best be done by creating specializations of the generic metamodel that correspond to individual mapping formalisms. This normally means that restrictions are added to the metamodel in terms of OCL constraints that formalize the specific properties of the respective formalism. In a second step, we have to develop a method for checking the compatibility of a given graphical model with a particular specialization of the metamodel. This is necessary for being able to determine whether a given model can be implemented with a particular formalism. Provided that specializations are entirely described using OCL constraints, this can be done using an OCL model checker. Finally, we have to develop methods for translating a given graphical model into an appropriate mapping formalism. This task can be seen as a special case of code generation where instead of executable code, we generate a formal mapping model that can be operationalized using a suitable inference engine.

In summary, the work presented here is the first step towards a comprehensive, model based approach for modeling and implementing ontology mappings. In contrast to many existing proposals, this approach takes a knowledge-level perspective on mapping modeling and supports an iterative development process where the mapping model is refined in a stepwise manner and the decision for a specific implementation formalism is only taken later in the process.

Acknowledgements. Research for this paper has been partially funded by the EU in the project NeOn (IST-2005-027595) and by the German Research Foundation (DFG) under the Graduate School IME – Universität Karlsruhe (TH).

References

1. P. Bouquet, F. Giunchiglia, F. van Harmelen, L. Serafini, and H. Stuckenschmidt. C-OWL: Contextualizing ontologies. In *Second International Semantic Web Conference ISWC'03*, volume 2870 of *LNCS*, pages 164–179. Springer, 2003.
2. S. Brockmans and P. Haase. A Metamodel and UML Profile for Networked Ontologies – A Complete Reference. Technical report, Universität Karlsruhe, April 2006. http://www.aifb.uni-karlsruhe.de/WBS/sbr/publications/ontology-metamodeling.pdf.
3. D. Calvanese, G. De Giacomo, and M. Lenzerini. A framework for ontology integration. In *Proceedings of the Semantic Web Working Symposium*, pages 303–316, Stanford, CA, 2001.
4. D. Calvanese, G. De Giacomo, and M. Lenzerini. Description logics for information integration. In A. Kakas and F. Sadri, editors, *Computational Logic: Logic Programming and Beyond*, volume 2408 of *Lecture Notes in Computer Science*, pages 41–60. Springer, 2002.
5. P. Haase and B. Motik. A mapping system for the integration of owl-dl ontologies. In *In Proceedings of the ACM-Workshop: Interoperability of Heterogeneous Information Systems (IHIS05)*, November 2005.
6. L. Serafini, H. Stuckenschmidt, and H. Wache. A formal investigation of mapping languages for terminological knowledge. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence - IJCAI05*, Edinburgh, UK, August 2005.
7. H. Stuckenschmidt and M. Uschold. Representation of semantic mappings. In Yannis Kalfoglou, Marco Schorlemmer, Amit Sheth, Steffen Staab, and Michael Uschold, editors, *Semantic Interoperability and Integration. Dagstuhl Seminar Proceedings*, volume 04391, Germany, 2005. IBFI, Schloss Dagstuhl.
8. S. Tessaris and E. Franconi. Rules and queries with ontologies: a unifying logical framework. In Ian Horrocks, Ulrike Sattler, and Frank Wolter, editors, *Description Logics*, volume 147 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2005.
9. J. Warmer and A. Kleppe. *Object Constraint Language 2.0*. MITP Verlag, 2004.

Virtual Integration of Existing Web Databases for the Genotypic Selection of Cereal Cultivars*

Sonia Bergamaschi and Antonio Sala

Dipartimento di Ingegneria dell'Informazione
Universitá di Modena e Reggio Emilia
bergamaschi.sonia@unimore.it, sala.antonio@unimore.it

Abstract. The paper presents the development of a virtual database for the genotypic selection of cereal cultivars starting from phenotypic traits.

The database is realized by integrating two existing web databases, Gramene [1] and Graingenes [2], and a pre-existing data source developed by the Agrarian Faculty of the University of Modena and Reggio Emilia. The integration process gives rise to a virtual integrated view of the underlying sources. This integration is obtained using the MOMIS system (Mediator envirOnment for Multiple Information Sources), a framework developed by the Database Group of the University of Modena and Reggio Emilia (www.dbgroup.unimo.it). MOMIS performs information extraction and integration from both structured and semistructured data sources. Information integration is performed in a semi-automatic way, by exploiting the knowledge in a Common Thesaurus (defined by the framework) and the descriptions of source schemas with a combination of clustering and Description Logics techniques. Momis allows querying information in a transparent mode for the user regardless of the specific languages of the sources. The result obtained by applying MOMIS to Gramene and Graingenes web databases is a queriable virtual view that integrates the two sources and allow performing genotypic selection of cultivars of barley, wheat and rice based on phenotypic traits, regardless of the specific languages of the web databases. The project is conducted in collaboration with the Agrarian Faculty of the University of Modena and Reggio Emilia and funded by the Regional Government of Emilia Romagna.

1 Introduction

In the last few years the progress in the field of the molecular biology gave rise to an exponential growth of data available to researchers. The great problem they are now facing is how to have access to this great amount of data in order to exploit them for their research activity. Many resources are available on Web

* This work is supported by the Italian Ministry of Research and University through the PRIN2004 "WISDOM" project.

[1] http://www.gramene.org/

[2] http://wheat.pw.usda.gov/

R. Meersman, Z. Tari et al. (Eds.): OTM 2006, LNCS 4275, pp. 909–926, 2006.

databases, but usually these informations reside in different, heterogeneous and sometimes numerous sources. Another problem is that these databases usually present different interfaces and structure of the information and are thus difficult to be queried by biology researchers. For these reasons a maybe simple information search can take long time and eventually fails because of the number of different data sources to be accessed.

What is needed to solve these problems is the definition of methods for:

1. Extracting and fusing the information coming from different (and heterogeneous) information sources (e.g. web sites and web databases)
2. Presenting the information according to a unique interface.

The paper presents the MOMIS system (Mediator envirOnment for Multiple Information Sources) [2], a mediator framework to perform information extraction and integration from heterogeneous distributed data sources and query management facilities to transparently support query posed to the integrated data sources. The framework consists of a language and two main components:

- The ODL_{I^3} language is an object-oriented language, with an underlying Description Logic; it is derived from the standard ODL-ODMG [9].
- The Ontology Builder: sources integration is performed in a semi-automatic way, by exploiting the knowledge in a Common Thesaurus (defined by the framework) and ODL_{I^3} descriptions of source schemas with a combination of clustering techniques and Description Logics. This integration process gives rise to a virtual integrated view of the underlying sources (the Global Schema, GVV) for which mapping rules and integrity constraints are specified to handle heterogeneity.
- The MOMIS Query Manager is the coordinated set of functions which take an incoming query, decompose the query according to the mapping of the GVV onto the local data sources relevant for the query, send the subqueries to these data sources, collect their answers, perform any residual filtering as necessary, and finally deliver the answer to the requesting user.

The MOMIS system is based on a conventional wrapper/mediator architecture, and provides methods and open tools for data management in Internet-based information systems (Fig.1). The MOMIS development begun as a joint collaboration between the University of Modena and Reggio Emilia and University of Milano and Brescia, within the INTERDATA national research project. The research activities continued within the SEWASIE European research project (IST-2001-34825) (www.sewasie.org).

MOMIS can analyze the contents of the source web databases and build a Global View. This Global View (GVV) is "Virtual" i.e. it is not materialized: data reside on the "local" sources and the View is an entry point for data retrieving. Consequently, only the changes in the information source structures have a side effect on a built GVV. Data changes do not have any influence on it. Moreover the existence of semantic tags in the sources, or the semantic annotations with respect to a lexical ontology (e.g. WordNet) are exploited to build

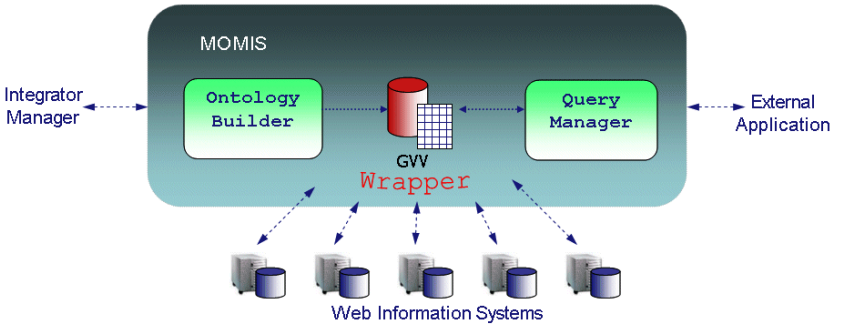


Fig. 1. The Momis Architecture

the GVV which can be seen as a domain ontology of the involved sources. The GVV, can be exported in RDFS and OWL thus guaranteeing interoperability with other external applications/ontologies or external users.

In this paper we present the use of the MOMIS system to perform intelligent data integration of existing databases to create a Virtual View for the genotypic selection of cereal cultivars based on their phenotype. This GVV has been realized as a part of the CEREALAB project conducted by the Agrarian faculty of the University of Modena and Reggio Emilia in collaboration with the Database Group of the University of Modena and Reggio Emilia and funded by the Regional Government of Emilia Romagna. The aim of the CEREALAB project is to make available to the cereal breeders of the Emilia Romagna region the knowledge learnt in the research activity by the Universities, in particular to provide them with a tool to perform genotypic selection of cereal cultivars from phenotypic traits. Thus, the idea was to create the CEREALAB database as an integration of two existing web databases, Gramene (concerning maize and rice) and Graingenes (concerning wheat and barley), with another data source storing the information achieved by the research group of the project. In this way the CEREALAB database performs both the tasks of (1)providing a valid support for the research activity, suppling information from the existing databases, and (2)being a knowledge base to store the data obtained by researchers.

The outline of the paper is the following: the first section describes the use of MOMIS to create the CEREALAB GVV: in particular Sect.2 describes the sources integration approach, Sect.3 sketches out the querying process presenting some examples. Finally Sect.4 compares MOMIS with other system and gives conclusions.

1.1 The CEREALAB Domain

The main entities of the CEREALAB domain are:

– Cultivar, which identify an assemblage of plants that has been selected for a particular attribute or combination of attributes and is clearly distinct, uniform and stable in its characteristics.

– Trait, an inherited feature of a plant.
– Gene, the unit of heredity in living organisms, which controls the physical development of the organism. An allele is any one of a number of viable DNA codings of the same gene occupying a given locus (position) on a chromosome.
– QTL, quantitative trait locus, a region of DNA that is associated with a particular trait. Though not necessarily genes themselves, QTLs are stretches of DNA that are closely linked to the genes that underlie the trait in question.
– Marker, a known DNA sequence (e. g. a gene or part of gene) that can be identified by a simple assay, associated with a certain phenotype. A genetic marker may be a short DNA sequence, such as a sequence surrounding a single base-pair change, or long one, like microsatellites.

Bearing in mind these main entities, we developed a kernel GVV (i.e. a bootstrap ontology) of the CEREALAB database to be used as a reference for the integration process of the available data sources. This ontology was created with MOMIS as a relational data source (see Fig.2), re-engineering the set of files used by the CEREALAB research group.

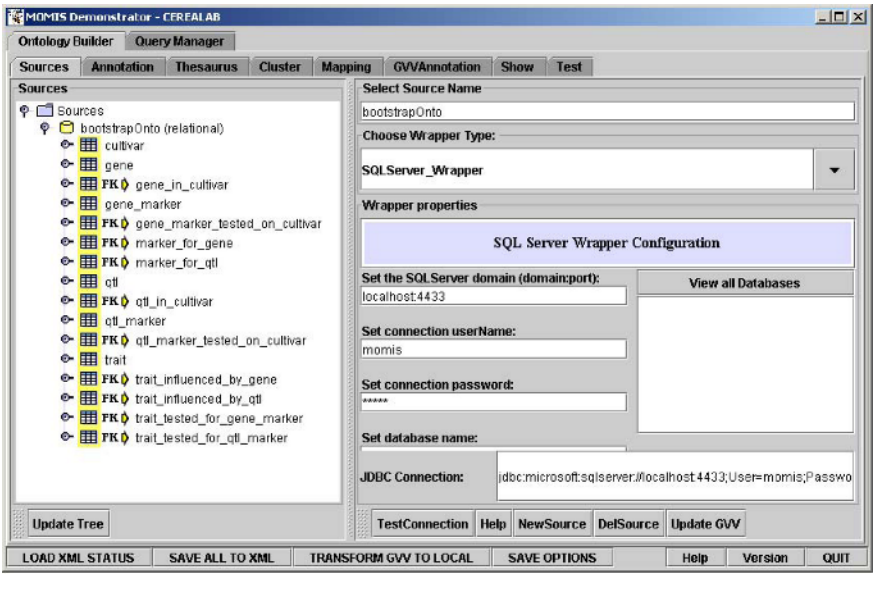


Fig. 2. The bootstrap Ontology for the CEREALAB database

2 The MOMIS Integration Methodology

In this section, we describe the information integration process for building the GVV. The process, shown in Fig.3, gives rise to Global Virtual View of several

specific data sources. The GVV-generation process in our case has been modified compared with the usual MOMIS approach [5] as we have a pre-existing bootstrap ontology which has to be enriched with other data source:

1. Insertion of a pre-existing ontology as local source. This data source stores the information achieved research group of the CEREALAB project.
2. Local source schemata extraction. Wrapper generates schemas for the involved sources and translates them into the common language ODL_{I^3} [2]
3. Local source annotation with WordNet. The integration designer chooses a meaning for each element of a local source schema, according to the WordNet lexical ontology (http://www.cogsci.princeton.edu/~wn). This phase may be executed semi-automatically: a tool supports the integration designer in the choice by proposing a WordNet concept for each source element.
4. Common thesaurus generation. Starting from the annotated local schema, MOMIS constructs a set of relationships describing inter and intraschema knowledge about classes and attributes of the source schemata.
5. GVV generation. The MOMIS methodology, applied to the common thesaurus and the local schemata descriptions, generates a global schema and sets of mappings with local schemata
6. Mapping refinement. The system automatically generates a Mapping Table for each global class of the GVV which can be extended by the designer.

The above methodology is described in the following sections. The Ontology Builder Tool supports the integration designer in all the GVV generation process phases to realize our virtual view for the genotypic selection of cereal cultivars.

2.1 The ODL_{I^3} Language

As a common data model for integrating a given set of local information sources, MOMIS uses an object-oriented language called ODL_{I^3}. ODL_{I^3} extends ODL with the following relationships expressing intra- and inter-schema knowledge for the source schemas: SYN (synonym of), BT (broader terms), NT (narrower terms) and RT (related terms). By means of ODL_{I^3}, only one language is exploited to describe both the sources (the input of the synthesis process) and the GVV (the result of the process). The translation of ODL_{I^3} descriptions into one of the Semantic Web standards such as RDF, DAML+OIL, OWL is a straightforward process. In fact, from a general perspective an ODL_{I^3} concept corresponds to a Class of the Semantic Web standard, and ODL_{I^3} relationships are translated into properties. Figure 3 shows the global schema generation, where local schemas are annotated according to the lexical ontology WordNet, the Common Thesaurus generation, and finally the GVV global classes. In particular, these ones are connected by means of mapping tables to the local schemas and are (semi-automatically) annotated according to WordNet. The designer can refine the mappings supported by the Ontology Builder.

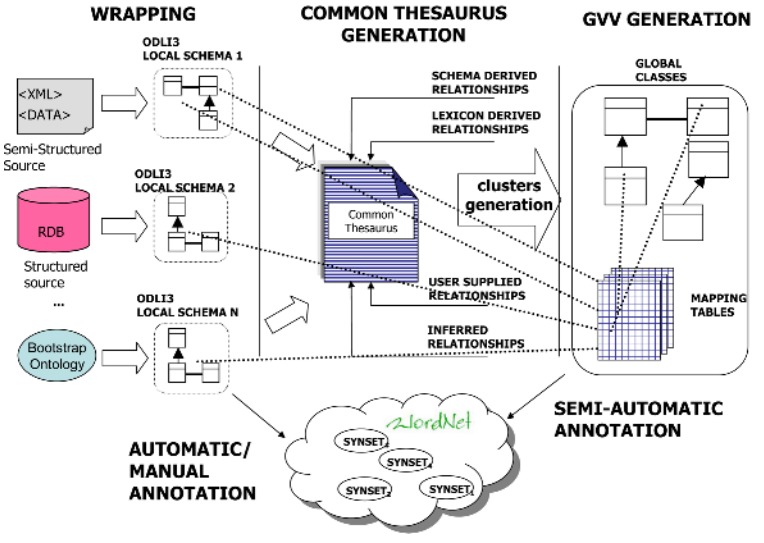


Fig. 3. Integration Process Overview

2.2 Insertion of Pre-existing Ontology / Wrapping: Extracting Data Structure for Sources

The first phase of the integration process is the choice of the data sources and their translation into ODL_{I^3} format. In our case a pre-defined ontology, that we call bootstrap ontology, existed and could be enriched by other data sources. Gramene and Graingenes have been chosen as further local sources, as they are the most significant for the domain. The translation process is performed by the MOMIS wrappers, which logically converts the source data structure into the ODL_{I^3} model. The wrapper architecture and interfaces are crucial, because wrappers are the focal point for managing the diversity of data sources. For conventional structured information sources (e.g. relational databases), schema description is always available and can be directly translated. In our case wrapping was easy as it is possible to download the two underlying relational databases of Gramene and Graingenes, creating two local sources. In this way we were able to use the MOMIS SqlServer Wrapper to manage both the sources.

After this step we have three local sources: the bootstrap ontology (CEREALAB), and the Gramene and Graingenes sources.

2.3 Semi-automatic Annotation of a Local Source with WordNet

The goal of the annotation phase is to assign a name and a set of meanings belonging to the WordNet [14] lexical system to each local class and attribute of the local schemata. For each element of a local schema the system automatically suggests a word form corresponding to the given term (if it exists): thus the designer may confirm or change the word form or meaning of each element.

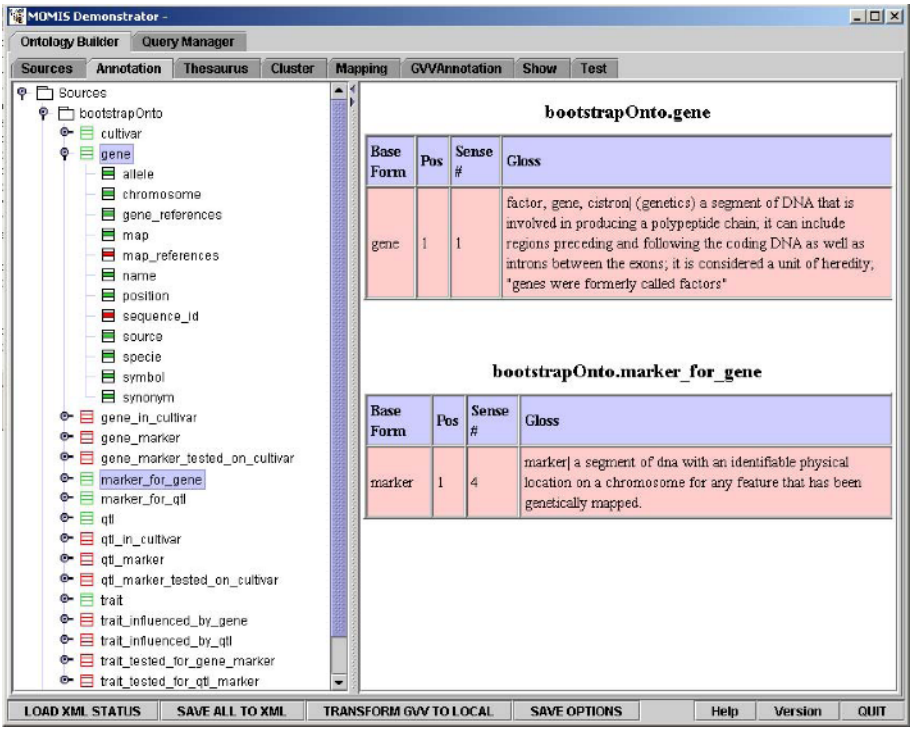


Fig. 4. The manual/automatic annotation for marker-for-gene/gene

As in our case the annotation is related to a very specific domain, WordNet lacks many of the terms involved. MOMIS provides the user with a WordNet Editor [3] to extend WordNet by adding new terms and synsets to the native elements of WordNet. Guided by the CEREALAB research group, we widely extended WordNet to face this new domain according to an existing technical glossary provided by the Gramene website. As an example, Fig.4 shows the automatic annotation for the "gene" attribute and the manual annotation, i.e. the definition we provided, for the "marker-for-gene" attribute. The annotation phase of the bootstrap ontology took quite a long time, but it resulted easier for the other two sources thanks to the extension provided to WordNet (as many of the terms involved in these two sources appear also in the bootstrap ontology) and to the capability of the WordNet Editor to cache the annoted terms and synsets. The advantage is that this extension step has to be performed just the first time a domain is handled.

2.4 Common Thesaurus Generation

Starting from the annotated local schemata, MOMIS constructs a Common Thesaurus describing intra and inter-schema knowledge in the form of SYN(synonyms), BT/NT(broader terms/narrower terms), and RT(meronymy/holonymy) relationships.

The Common Thesaurus is constructed through an incremental process in which the following relationships are added:

- schema-derived relationships: relationships holding at intra-schema level are automatically extracted by analyzing each schema separately. For example, MOMIS extracts intraschema RT relationships from foreign keys in relational source schemas. When a foreign key is also a primary key, in both the original and referenced relation, MOMIS extracts BT and NT relationships, which are derived from inheritance relationships in object-oriented schemas.
- lexicon-derived relationships: we exploit the annotation phase in order to translate relationships holding at the lexical level into relationships to be added to the Common Thesaurus.
- designer-supplied relationships: new relationships can be supplied directly by the designer, to capture specific domain knowledge.
- inferred relationships: Description Logics (DL) techniques of ODB-Tools [4],[6] are exploited to infer new relationships, by means of subsumption computation applied to a "virtual schema" obtained by interpreting BT/NT as subclass relationships and RT as domain attributes.

A detailed presentation of the methodology can be found in [2], [7].

Figure 5 shows some relationships automatically extracted by MOMIS for the `gene` classes and attributes. In our case, the relationships holding at lexical level are widely exploited to discover semantic relationships between local classes. For example, MOMIS sematically promotes the RT relationship between `gene` and `allele` (see Fig.4) into a BT relationship (see Fig.5, line 6). For the class `marker-for-gene` there is no evident schema relationship, but sematically it is identified as a NT of `gene` (see Fig.5, last line).

2.5 Global Virtual View (GVV) Generation

The MOMIS methodology allows us to identify similar ODL_{I^3} classes, that is, classes that describe the same or semantically related concepts in different sources, and mappings to connect the global attributes of each global class with the local sources' attributes. To this end, affinity coefficients are evaluated for all possible pairs of ODL_{I^3} classes, based on the relationships in the Common Thesaurus properly strengthened. Affinity coefficients determine the degree of matching of two classes based on their names (Name Affinity coefficient) and their attributes (Structural Affinity coefficient) and are fused into the Global Affinity coefficient, calculated by means of the linear combination of the two coefficients. Global affinity coefficients are then used by a hierarchical clustering algorithm, to include ODL_{I^3} classes in clusters according to their degree of affinity. The designer may interactively refine and complete the proposed integration results; in particular, the mappings which has been automatically created by the system can be fine tuned as discussed in Sect.2.6.

Being so specific the domain we are facing, the annotation phase gave rise to a large number of BT/NT relationships. This, together with a very different structure of the two sources, Gramene and Graingenes, from our bootstrap

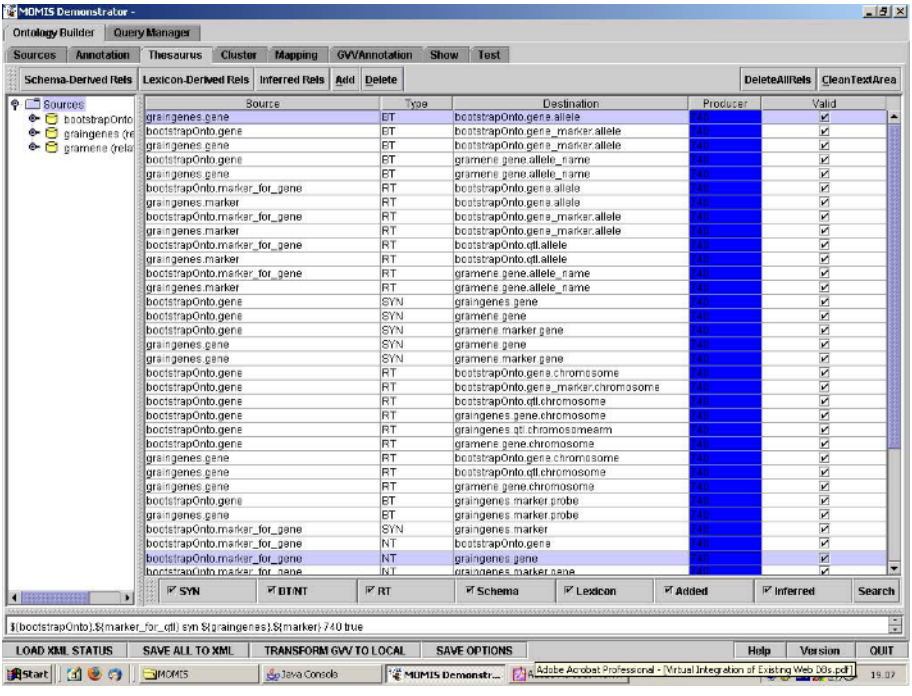


Fig. 5. The Common Thesaurus for the `gene` and `marker-for-gene` classes

Ontology, forced us to give less weight than usual to the BT/NT relationship when calculating the Name Affinity and the Structural Affinity coefficients. In particular, instead of the default weight to calculate the coefficients, SYN=1, BT/NT=0.8, we tuned the weights in order to consider SYN relationships as the most relevant. In this way the system gave rise to a set of significant Global Classes that reflected the entities we defined in Sect.1.1. No particular issues have been encountered during the generation of the GVV as all the data types appearing in the different sources are homogeneous for the same real world objects. These Global Classes have then been verified by the agrarian researchers to validate the Global Virtual View obtained and to indicate possible refinements. The main issues are identifying the Join Conditions and in few cases defining the Resolution Functions, which will be presented in Sect.2.6.

2.6 Mapping Refinement

The system automatically generates a Mapping Table (MT) for each global class C of a GVV , whose columns represent the local classes $L(C)$ belonging to C and whose rows represent the global attributes of C. An element $MT[GA][LC]$ represents the set of local attributes of LC which are mapped onto the global attribute GA. Figure 6 shows the MT of the `gene` Global Class.

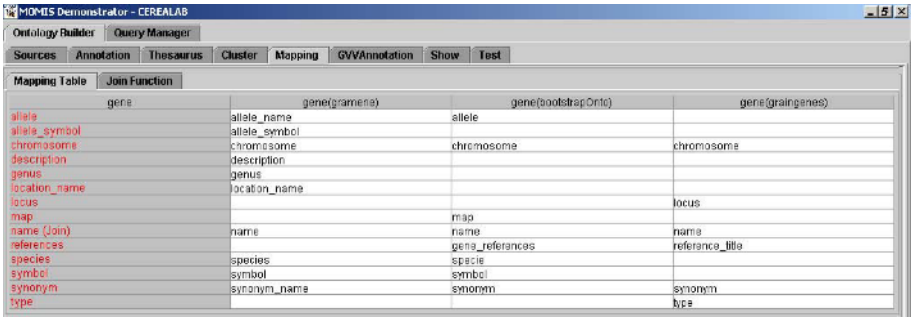


Fig. 6. The Mapping Table for the gene Global Class

More formally, we define an Integration System IS = (GVV, N, M) as constituted by:

- A GVV, which is a schema expressed in ODL_{I^3}.
- A set N of local sources; each local source has a schema also expressed in ODL_{I^3}.
- A set M of GAV mapping assertions between the GVV and N, where each assertion associates to an element g in GVV a query Q_N over the schemas of a set of local sources in N.

More precisely, for each global class C of the GVV we define:

- a (possibly empty) set of local classes, denoted by $L(C)$, belonging to the local sources in N
- a conjunctive query Q_N over $L(C)$.

Intuitively, the GVV is the intensional representation of the information provided by the Integration System, whereas the mapping assertions specify how such an intensional representation relates to the local sources managed by the Integration System. The query Q_N associated to a global class C is implicitly defined by the designer starting from the MT of C. The designer can extend the MT by adding:

- Data Conversion Functions from local to global attributes
- Join Conditions among pairs of local classes belonging to C
- Resolution Functions for global attributes to solve data conflicts of local attribute values.

On the basis of the resulting MT the system automatically generates a query Q_N associated to C, by extending the Full Disjunction operator [11], which is explained in the following.

Data Conversion Functions. The Ontology Designer can define, for each not null element $MT[GA][L]$, a Data Conversion Function, denoted by $MTF[GA][L]$, which represents the mapping of local attributes of L into the global attribute

GA. $MTF[GA][L]$ is a function that must be executable/supported by the class L local source. For example, for relational sources, $MTF[GA][L]$ is an SQL value expression. $T(L)$ denotes L transformed by the Data Conversion Functions.

Join Conditions. Merging data from different sources requires different instantiations of the same real world object to be identified; this process is called object identification [16], [18], [1], [10].

To identify instances of the same object and fuse them we introduce Join Conditions among pairs of local classes belonging to the same global class. Given two local classes L_1 and L_2 belonging to C, a Join Condition between L_1 and L_2, denoted with $JC(L_1, L_2)$, is an expression over $L_1.A_i$ and $L_2.A_j$ where A_i (A_j) are global attributes with a not null mapping in L_1 (L_2). As an example, the join condition for the `gene` Global Class is defined as follow:

```
((graingenes.gene.name) = (bootstrapOnto.gene.name)) AND
(((gramene.gene.name) = (bootstrapOnto.gene.name))
OR ((gramene.gene.name) = (graingenes.gene.name)))
```

Resolution Functions. In MOMIS the approach proposed in [16] has been adopted: a Resolution Function for solving data conflicts may be defined for each global attribute mapping onto local attributes coming from more than one local source; in this way we can define what value shall appear in the result. Our system provides some standard kinds of resolution functions (Random, Aggregation, Coalescence and others). In our domain we used the followings:

1. *Precedence function*: experimental results obtained by the CEREALAB research group regard mainly italian cultivars. These data could be different from data from the two existing sources, especially referring to phenotypic information since the Gramene and Graingenes are american databases. For this reason a precedence function has been used, to give priority to the CEREALAB informations as they are related to italian cultivars.
2. *All Values*: considering the integration viewpoint, the aim is to preserve all the information coming from the sources. For example, a "reference" attribute is often present for many entities to provide bibliographic references. Sometimes each source can cite different relevant references for the instance in exam. To let the user get all the data provided by the local sources as a result, we used the All Values function provided by MOMIS to return all the references present in the different sources.

Full Disjunction. Q_N is defined in such a way that it contains a unique tuple resulting from the merge of all the different tuples representing the same real world object. This problem is related to that of computing the natural outer-join of many relations in a way that preserves all possible connections among facts [17]. Such a computation has been termed as Full Disjunction (FD) by Galindo Legaria [11]. In our context: given a global class C composed of L_1,L_2, ..., L_n, we consider

$$FD(T(L_1), T(L_2), \ldots, T(L_n))$$

computed on the basis of the Join Conditions. With more than 2 local classes, the computation of FD is performed as follows. We assume that: (1) each L contains a key, (2) all the join conditions are on key attributes, and (3) all the join attributes are mapped into the same set of global attribute, say K. Then, it can be proved that: (1) K is a key of C, and (2) FD can be computed by means of the following expression:

$$(T(L_1) \text{ full join } T(L_2) \text{ on } JC(L_1, L_2)) \text{ full join } T(L_3)$$

$$\text{on } (JC(L_1, L_3) \text{ OR } JC(L_2, L_3)) \ldots \text{ full join } T(L_n) \text{ on } (JC(L_1, L_n)$$

$$\text{OR } JC(L_2, L_n) \text{ OR} \ldots \text{OR } JC(L_{n-1}, L_n))$$

Finally, Q_N is obtained by applying Resolution Functions to the attributes resulting from the above expression: for a global attribute GA we apply the related Resolution Function to $T(L_1).GA$, $T(L_2).GA$, ..., $T(L_k).GA$. As an example, Q_N for the `gene` Global Class is:

```
bootstrapOnto.gene full outer join graingenes.gene
on (((graingenes.gene.name) = (bootstrapOnto.gene.name)))
full outer join gramene.gene
on (((gramene.gene.name) = (bootstrapOnto.gene.name))
OR ((gramene.gene.name) = (graingenes.gene.name)))
```

3 The MOMIS Query Manager

The MOMIS Query Manager is the coordinated set of functions which takes an incoming query (say global query), defines a decomposition of the query according to the mapping of the GVV onto the local data sources, sends the subqueries to these data sources, collects their answers, fuse them (performing any residual filtering as necessary), and finally delivers the answer. Query processing consists of the following steps:

1. Query rewriting: to rewrite a global query as an equivalent set of queries expressed on the local sources (local queries)
2. Local queries execution: the local queries are sent and executed at local sources
3. Fusion and Reconciliation: the local answers are fused into the global answer.

Let us introduce a simple query in order to show the query processing steps:

```
select * from gene where name like '\%resistance\%'
```

The query retrieves all the genes that contain the word "resistance" in their name, i.e. the genes that express a kind of resistance. With Momis, this query allows the user to transparently retrieve information from Gramene and Graingenes with a single query (see Fig.7).

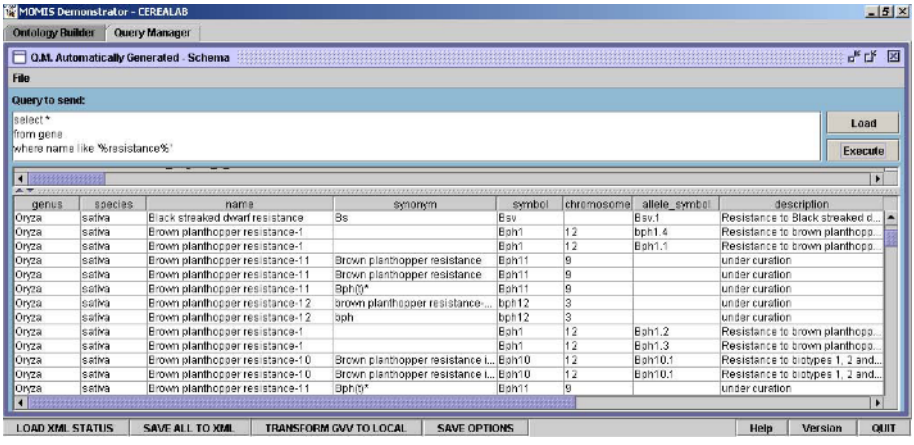


Fig. 7. A query on the CEREALAB GVV

3.1 Query Rewriting

MOMIS uses a global-as-view (GAV) approach [12] to model the mapping among the GVV and the local schemata. Then the global query is rewritten by means of unfolding, that is, by expanding each atom of the global query according to its definition in the mapping. A detailed description can be found in [8].

We consider a Global Query Q over a Global Class G:

$$Q = \text{select} < Q_{select-list} > \text{from G where} < Q_{condition} >$$

where $Q_{condition}$ is a Boolean expression of positive atomic constraints:

$$(GA\ op\ value)\text{or}(GA_1\ op\ GA_2)$$

where GA_1 and GA_2 are attributes of the Global Class G.

The query rewriting process is composed of the following steps:

1. Atomic constraint mapping
   In this step, each atomic constraint of a query Q is rewritten into one that can be supported by the local class. The atomic constraint mapping is performed on the basis of mapping functions defined in the Mapping Table.
2. Residual Constraints computation: Intuitively, residual constraints are the constraints of the global query that are not mapped in all local queries.
3. Local select-list computation: The select-list of a local query is a set of attributes, including the global query attributes, the join attributes, the residual constraints attributes, translated into the correspondent set of local attributes on the basis of the mapping table.

The output of the Query Rewriting process is a set of local queries; each local query Q_L over a local class L is in a form supported by the local source of the class L. For relational sources, a local query Q_L over L will be in the form:

$$Q_L = \text{select} < Q_{L_{select-list}} > \text{from L where} < Q_{Lcondition} >$$

In our example, the local queries Q_L over L are:

```
Source ‘‘bootstrapOnto’’
Query on Local Interface ‘‘bootstrapOnto.gene’’:
SELECT gene.name, gene.map, gene.symbol
FROM gene
WHERE (name) like (’\%resistance\%’)

Source ‘‘gramene’’
Query on Local Interface ‘‘gramene.gene’’:
SELECT gene.name, gene.species
FROM gene
WHERE (name) like (’\%resistance\%’)

Source ‘‘graingenes’’
Query on Local Interface ‘‘graingenes.gene’’:
SELECT gene.type, gene.locus, gene.name, gene.chromosome,
gene.fullname, gene.synonym, gene.reference-title
FROM gene
WHERE (name) like (’\%resistance\%’)
```

3.2 Local Queries Execution / Fusion and Reconciliation

A local query L_Q is sent to the source including the local class L; its answer is transformed by applying the mapping functions related to L: in this way, we perform the conversion of the local class instances into the GVV instances. The result of this conversion is materialized in a temporary table. No data conversion function is necessary for our domain.

Temporary tables are fused and reconciliated into the global answer. In our example, Q_N is:

```
select "Join_Eng_gene_graingenes_gene".type AS type_1,
"Join_Eng_gene_gramene_gene".genus AS genus_1,
"Join_Eng_gene_bootstrapOnto_gene".name AS name_1,
"Join_Eng_gene_graingenes_gene".name AS name_2,
"Join_Eng_gene_gramene_gene".name AS name_3,
"Join_Eng_gene_bootstrapOnto_gene".specie AS species_1,
"Join_Eng_gene_gramene_gene".species AS species_2,
"Join_Eng_gene_bootstrapOnto_gene".map AS map_1,
"Join_Eng_gene_gramene_gene".description AS description_1,
"Join_Eng_gene_graingenes_gene".locus AS locus_1,
"Join_Eng_gene_gramene_gene".location_name AS location_name_1,
"Join_Eng_gene_bootstrapOnto_gene".chromosome AS chromosome_1,
"Join_Eng_gene_graingenes_gene".chromosome AS chromosome_2,
"Join_Eng_gene_gramene_gene".chromosome AS chromosome_3,
```

```
"Join_Eng_gene_gramene_gene".allele_symbol AS allele_symbol_1,
"Join_Eng_gene_bootstrapOnto_gene".allele AS allele_1,
"Join_Eng_gene_gramene_gene".allele_name AS allele_2,
"Join_Eng_gene_bootstrapOnto_gene".synonym AS synonym_1,
"Join_Eng_gene_graingenes_gene".synonym AS synonym_2,
"Join_Eng_gene_gramene_gene".synonym_name AS synonym_3,
"Join_Eng_gene_bootstrapOnto_gene".symbol AS symbol_1,
"Join_Eng_gene_gramene_gene".symbol AS symbol_2,
"Join_Eng_gene_bootstrapOnto_gene".gene_references AS references_1,
"Join_Eng_gene_graingenes_gene".reference_title AS references_2
from "Join_Eng_gene_graingenes_gene" full outer join
"Join_Eng_gene_bootstrapOnto_gene"
on ((("Join_Eng_gene_bootstrapOnto_gene".name) =
("Join_Eng_gene_graingenes_gene".name)))
full outer join "Join_Eng_gene_gramene_gene"
on ((("Join_Eng_gene_gramene_gene".name) =
("Join_Eng_gene_graingenes_gene".name))
OR (("Join_Eng_gene_gramene_gene".name) =
("Join_Eng_gene_bootstrapOnto_gene".name)))
```

3.3 Query on Multiple Global Classes

Another example of query can be seen in Fig.8. The picture shows a query on multiple Global Classes. In particular this query retrieves all the genes, chromosome, position of the gene and the marker for that particular gene for the Triticum species. This query is performed on two Global Classes, gene and marker-for-gene.

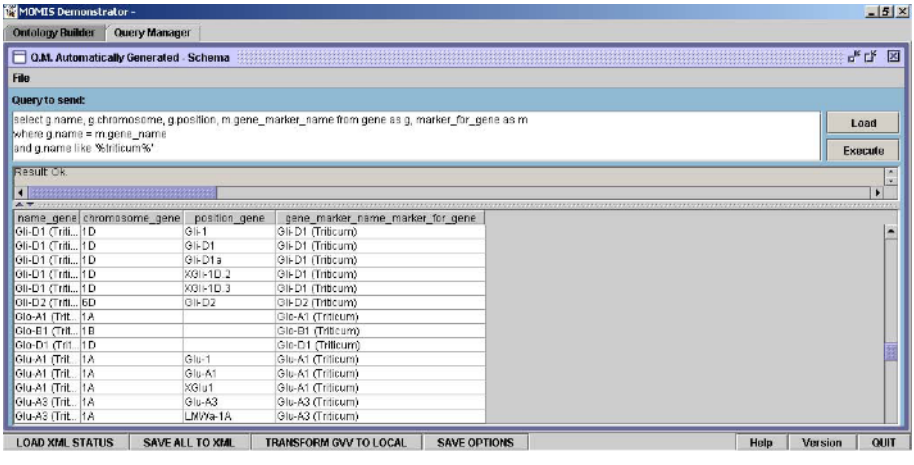


Fig. 8. A query on multiple Global Classes of the CEREALAB Global Virtual View

First, the query is decomposed into two queries on the single Global Classes:

```
Single Class Query_1 (JoinEngine JE001) :
select g.name , g.chromosome , g.position  from gene as g
where (name like  '%triticum%' )

Single Class Query_2 (JoinEngine JE002) :
select m.gene_name  from marker_for_gene as m
```

Each query is then further decomposed and rewritten as an equivalent set of queries expressed on the local sources.

```
Source "bootstrapOnto"
Query on Local Interface "bootstrapOnto.gene":
SELECT gene.name
FROM gene
WHERE (name) like ('%triticum%')

Source "gramene"
Query on Local Interface "gramene.gene":
SELECT gene.name
FROM gene
WHERE (name) like ('%triticum%')

Source "graingenes"
Query on Local Interface "graingenes.gene":
SELECT gene.locus, gene.name, gene.chromosome
FROM gene WHERE (name) like ('%triticum%')

Source "bootstrapOnto"
Query on Local Interface "bootstrapOnto.marker_for_gene":
SELECT marker_for_gene.gene_name
FROM marker_for_gene
```

Finally, the results of the Local Queries execution are then fused together according to the residual clause:

```
select g.name , g.chromosome , g.position ,
m.gene_name  from gene as g , marker_for_gene as m
where (g.name = m.gene_name )
```

4 Summary and Discussions

We described the design and realization of the CEREALAB database, a support for the research activity about cereal cultivars. This database has been developed as a Virtual View of two existing web databases, Gramene and Graingenes, integrated with another relation source designed to store the information achieved

by the research group of the CEREALAB project. The CEREALAB database integrates information from existing databases according to a common ontology providing a unique interface to query different sources.

A possible improvement of the usage of the MOMIS system in this domain would be the use of existing biological ontologies, for example the Open Biological Ontologies (OBO) (www.obo.org), as an ontological support to perform the integration process. The primary area related to our work is the area of heterogeneous information integration. Many projects based on mediator architectures have been developed [19], [15], [13] . In this paper we described the application of the MOMIS system for both integrating data sources concerning the molecular biology domain and giving the possibility to querying them. The result of this work is the creation of a virtual database for the genotypic selection of cereal cultivars.

A completely different approach that can be promising in the biology domain is presented in [20], where the dynamic query translation task is addressed. The idea is to develop a light-weight domain based form assistant which can handle alternative sources in the same domain to help the user query across dinamically selected web sources. This approach is completely different from our system as it just suggests a possible translation of the user's query for different web data source, while MOMIS performs information integration and provides a unique interface to query multiple sources.

References

1. Ananthakrishna, R., Chaudhuri, S., & Ganti, V. , "Eliminating fuzzy duplicates in data warehouses", In VLDB Conference, (pp. 586597) (2002).
2. S. Bergamaschi, S. Castano, D. Beneventano, M. Vincini: "Semantic Integration of Heterogeneous Information Sources", Special Issue on Intelligent Information Integration, Data & Knowledge Engineering, Vol. 36, Num. 1, Pages 215-249, Elsevier Science B.V. 2001.
3. R. Benassi, S. Bergamaschi, A. Fergnani, D. Miselli: "Extending a Lexicon Ontology for Intelligent Information Integration", European Conference on Artificial Intelligence (ECAI2004). Valencia, Spain, 22-27 August 2004.
4. D. Beneventano, S. Bergamaschi, C. Sartori, M. Vincini "ODB-QOptimizer: a tool for semantic query optimization in OODB". ICDE'97, UK, April 1997.
5. D. Beneventano, S. Bergamaschi, F. Guerra, M. Vincini: "The MOMIS approach to Information Integration", IEEE and AAAI International Conference on Enterprise Information Systems (ICEIS01), Setbal, Portugal, 7-10 July, 2001.
6. D. Beneventano, S. Bergamaschi, F. Guerra, M. Vincini: "Synthesizing an Integrated Ontology". IEEE Internet Computing 7(5): 42-51 (2003).
7. D. Beneventano, S. Bergamaschi, C. Sartori: "Description Logics for Semantic Query Optimization in Object-Oriented Database Systems", ACM Transaction on Database Systems, Volume 28: 1-50 (2003).
8. D.Beneventano, S.Bergamaschi:"Semantic Search Engines based on Data Integration Systems". In Semantic Web: Theory, Tools and Applicantions (Ed. Jorge Cardoso), Idea Group Publishing, May 2006
9. R. G. G. Cattell, Douglas K. Barry: "The Object Data Standard: ODMG 3.0" Morgan Kaufmann 2000.

10. Chaudhuri, S., Ganjam, K., Ganti, V., & Motwani, R. . "Robust and efficient fuzzy match for online data cleaning". In ACM SIGMOD Conference (pp. 313324) (2003).
11. C. A. Galindo-Legaria, "Outerjoins as Disjunctions". SIGMOD Conference 1994, 348-358.
12. A. Halevy, A. Y. Halevy. "Answering queries using views: A survey". Very Large Database J., 10(4):270-294, 2001.
13. C. Li, R. Yerneni, V. Vassalos, H. Garcia-Molina, Y. Papakonstantinou, J. Ullman, M. Valiveti. "Capability Based Mediation in TSIMMIS", SIGMOD 98, Seattle, June 1998.
14. A.G. Miller. "A lexical database for English". Communications of the ACM, 38(11):39:41,1995.
15. R. J. Miller, M. A. Hernandez, L. M. Haas, L. Yan, C. T. H. Ho, L. Popa, and R. Fagin, "The Clio project: managing heterogeneity", ACM SIGMOD Record 30, 1 (March 2001), pp. 78-83.
16. F. Naumann, M. Haussler: "Declarative Data Merging with Conflict Resolution". International Conference on Information Quality (IQ 2002). 2002, pages 212-224.
17. A. Rajaraman , J. D. Ullman: "Integrating Information by Outerjoins and Full Disjunctions". PODS 1996, pages 238-248.
18. Tejada, S., Knoblock, C. A., & Minton, S. , "Learning object identification rules for information integration", Inf. Syst., 26 (8), 607633 (2001).
19. L. Yan, R. J. Miller, L. M. Haas, and R. Fagin, "Data-driven understanding and refinement of schema mappings", Proc. 2001 ACM SIGMOD Conference (SIGMOD '01), pp. 485-496.
20. Zhen Zhang, Bin He, Kevin Chen-Chuan Chang: Light-weight Domain-based Form Assistant: Querying Web Databases On the Fly. VLDB 2005: 97-108

SMOP: A Semantic Web and Service Driven Information Gathering Environment for Mobile Platforms

Özgür Gümüs[1], Geylani Kardas[2], Oguz Dikenelli[1], Riza Cenk Erdur[1], and Ata Önal[1]

[1] Ege University, Department of Computer Engineering, Bornova, 35100 Izmir, Turkey
{ozgur.gumus, oguz.dikenelli, cenk.erdur, ata.onal}@ege.edu.tr
[2] Ege University, International Computer Institute, Bornova, 35100 Izmir, Turkey
geylani.kardas@ege.edu.tr

Abstract. In this paper, we introduce a mobile services environment, namely SMOP, in which semantic web based service capability matching and location-aware information gathering are both used to develop mobile applications. Domain independency and support on semantic matching in mobile service capabilities are the innovative features of the proposed environment. Built-in semantic matching engine of the environment provides the addition of new service domain ontologies which is critical in terms of system extensibility. Therefore the environment is generic in terms of developing various mobile applications and provides most relevant services for mobile users by applying semantic capability matching in service lookups. GPS (Global Positioning System) and map service utilization cause to find near services in addition to capability relevancy. The software architecture and system extensibility support of the environment are discussed in the paper. The real life implementation of the environment for the estate domain is also given as a case study in the evaluation section of the paper.

1 Introduction

Location aware and personal interest based information gathering from mobile devices is very active application and research area. Different experimental applications with such capabilities have been introduced in the literature [1] [2] [3]. From our perspective, one of the most critical problems of such applications is the extensibility of the software architecture. In our context, extensibility of the architecture describes domain independency: the addition of the new service domains to the running mobile application.

In this paper, we will introduce a software environment to develop location aware and semantic web based mobile information gathering applications. From now on, we will call this environment as SMOP (A Semantic Web and Service Driven Information Gathering Environment for Mobile Platforms). The innovative feature of SMOP is the use of a semantic matching engine which can identify semantically related knowledge, based on user queries. This semantic matching engine supports the addition of new service domain ontologies and this capability is critical in terms of domain independency. Moreover, service oriented infrastructure of SMOP further

R. Meersman, Z. Tari et al. (Eds.): OTM 2006, LNCS 4275, pp. 927 – 940, 2006.

contributes to the extensibility, since it is possible to introduce new services without affecting the core of the architecture.

The paper is organized as follows: Section 2 introduces the system overview and architecture of the environment. System extensibility from domain independency perspective is discussed in section 3. Section 4 gives an example case study and evaluates the environment while a new domain is added to the system. Section 5 gives an overview of related works in the literature and compares SMOP with those works. Conclusion and future work are given in section 6.

2 System Overview

2.1 General Architecture

SMOP has a three tiered architecture. Mobile client, the server side and platform web services exist in the corresponding tiers. The components of each tier and interactions between those components are shown in Fig. 1. Each tier will be discussed in more detail in the following subsections.

2.2 Mobile Client

The client side is responsible for getting the GPS data, providing the interface for specifying the user requests and displaying the results, sending the requests to the server in XML format and parsing the results received in XML format.

The "GPS Data Parser" component is responsible for parsing the location data retrieved from the internal or external GPS receiver. To get the most current position of the client/user, it reads periodically raw GPS data, parses it and gets the Latitude and Longitude coordinates of the client/user.

Dynamic creation of visual interfaces at run-time is the responsibility of the "User Interface Generator" component. User screens are usually limited in mobile devices; hence, the user interfaces created are not so complex. The primary user interface window provides view of all data received along with user-selectable menu choices for controlling the application. When the mobile client connects to the server at first time, the domain names which are added to the platform up to that time are retrieved and shown to the user by the "User Interface Generator". Then, user selects a domain and the XML files containing the concepts belonging to selected domain's ontology are transferred from the server. After that, the transferred XML files are parsed and a visual interface is created to let the user specify his/her choices. Hence, a user interface where users can specify their choices is created independently at run-time for each different domain by the "User Interface Generator". In fact, the ontologies are represented in Web Ontology Language (OWL) in the knowledgebase of the Semantic Matching Service (SMS). However, since mobile devices are resource limited, we simplified and represented these ontologies in simple XML format to make the parsing process efficient in the mobile device. Otherwise, the mobile device should execute the code necessary to parse OWL documents. When the results of semantic match query are returned, the "User Interface Generator" lists the found domain instances and shows detailed information about these instances. User may

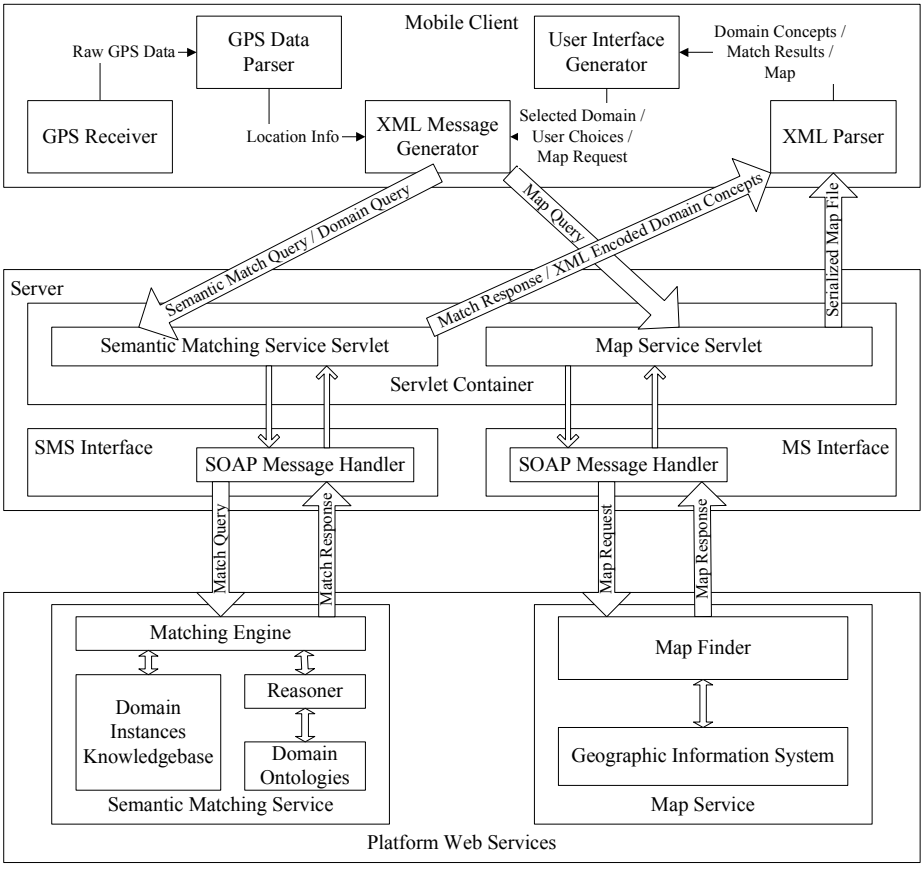


Fig. 1. SMOP's three tiered architecture

want to see one of those instances on the map. In this case, the "User Interface Generator" shows the map which is provided by the Map Service (MS) on the screen.

We preferred the requests and results to be transmitted in XML format, since it is a well-known web standard. So, the "XML Message Generator" component of the mobile client converts the (1) request of selected domain concepts, (2) request of semantic match according to user choices and (3) request of a map into the XML format. It also inserts the GPS location data of the client/user into last two requests. Then, it sends the requests in XML format to the server using a GPRS Http network connection.

The "XML Parser" component parses the XML response document received from the server. So, depending on the given request, this XML document may include (1) concepts of a selected domain or (2) domain instances satisfying user choices in the form of a collection or (3) a serialized map showing one of found instances. Mobile devices have a limited memory. For this reason, the "XML Parser" component has been designed to be small and light. The pull parser technique, in which the software drives the parsing, has been used. In this technique, only some part of a XML

document is read at once; hence, it does not need a large memory size. The application drives the parser through the document by repeatedly requesting the next piece. Our mobile client can process and display information as it is parsed after being downloaded from the server. In this case the "XML Parser" component basically iterates over the XML tree and finds the items. The parsed data is then passed to the "User Interface Generator" component to be printed on the screen of the mobile device.

2.3 The Server

The server side has the components that are responsible for meeting the client requests, fulfilling these requests via web services and returning the results to the client: servlet components interact with the client, interface components interact with web services and corresponding servlet and interface components interact with each other.

The "SMS Servlet" component takes the XML encoded match request, decomposes it and prepares the inputs of the SMS. These inputs are the domain concept (including its some properties that are chosen by the user) to be discovered and the required "degree of match" value. They are sent to the "SMS Interface" component. This component invokes the SMS and sends the outputs to the corresponding servlet. These outputs are the discovered domain instances that are matched with request sorted by the "degree of match" values. The "SMS Servlet" takes these instances, re-sorts the ones which have equivalent "degree of match" value according to distance to the client/user using the GPS location data and finally it inserts them into a collection. It then converts this collection into an XML message and sends the formed XML message to the client as an Http response.

The "MS Servlet" and "MS Interface" components work similar to the corresponding SMS components explained above. But, the inputs of the MS are the GPS location data of the mobile client/user and the instance that will be shown on the map. The "MS Servlet" takes the returned map from the "MS Interface" and serializes it into an XML message and sends it to the client.

2.4 Platform Services

2.4.1 Semantic Matching Service

The basic idea behind the matching process is to find the advertised concepts that are identical to the requested one. However, the advertised and requested concepts can be semantically related with each other but are not directly identical. In this case, a semantic matching process is required. Semantic matching process is a matching process that can identify the semantic relationships between the advertised and requested concepts. SMS executes this process. It has a registry to keep records of knowledge about advertised domain instances. It can be searched for the semantically most suitable instances using specific domain concepts.

The "Domain Instances Knowledgebase" component stores the instances of all domains. In this study, we have defined an abstract concept named as *Domain*. To add a new domain to the platform, a new concept that is specific type of *Domain* is defined. This new domain concept has its own data type properties and object type

properties. Instances of this concept are created using different predefined domain ontologies for the object type properties and stored in the knowledgebase.

The "Matching Engine" component realizes matching of requested domain concept with advertised domain instances and produces the list of suitable instances sorted by "degree of match" values. It uses the "Reasoner" component to determine subsumption relation between ontological concepts. It uses an algorithm similar to one that is especially for discovery of semantic web services proposed in Paolucci et al's study [6] and it has explained in detail in our previous work [7].

2.4.2 Map Service

MS provides a satellite map enclosing specified locations in an area. The "Geographic Information System" (GIS) component is a software package named as Mapxtreme Java Edition of Mapinfo Corporation. It stores the information about geographic objects on the earth including their attributes, positions and shapes. It provides an API named MapJ to form a map image and make some operations and analysis on this image. The "Map Finder" component interacts with the GIS using this API. It sends the position of objects that must be enclosed in the map. GIS forms a map image on which the given objects (in this case the client and selected domain instance) are marked, and returns it to the "Map Finder" in binary format.

3 System Extensibility from Domain Independency Perspective

SMOP provides a mobile services environment in which capabilities of both semantic web and location-aware information gathering are utilized to develop mobile applications for various business domains. Considering system extensibility aspect, domain independency support is an important feature and should be provided within the platform via software reusability. In this section, domain independency support in SMOP is discussed.

When service capability matching is required for a new domain, it is enough to initialize knowledgebase of the internal semantic service matching engine of the platform with this new domain ontology without any need of software architecture modification. Only domain concept and related semantic service capability advertisement ontologies are needed to be added into the knowledgebase of the engine. Communication between the engine and the outer environment is realized over standard OWL messages: Match requests and responses are composed of RDF (Resource Description Framework) triples. Hence, change in domain only effects the communication content, neither structure nor software architecture. More about the internal execution and capability matching algorithm of our matching engine are beyond the scope of this paper. However, they have been discussed in [7] and [8].

Taking into consideration of Model-View-Controller system pattern [9] decomposition of the SMOP's architecture; it can be said that domain ontology and related knowledgebase represent *model*, mobile GUI components residing on cell phones represent *view* and finally servlet container and related web services stand for the *controller* layer within the system. We applied an abstract domain model into the controller layer of the SMOP to support various business domains without any code modification (Fig. 2).

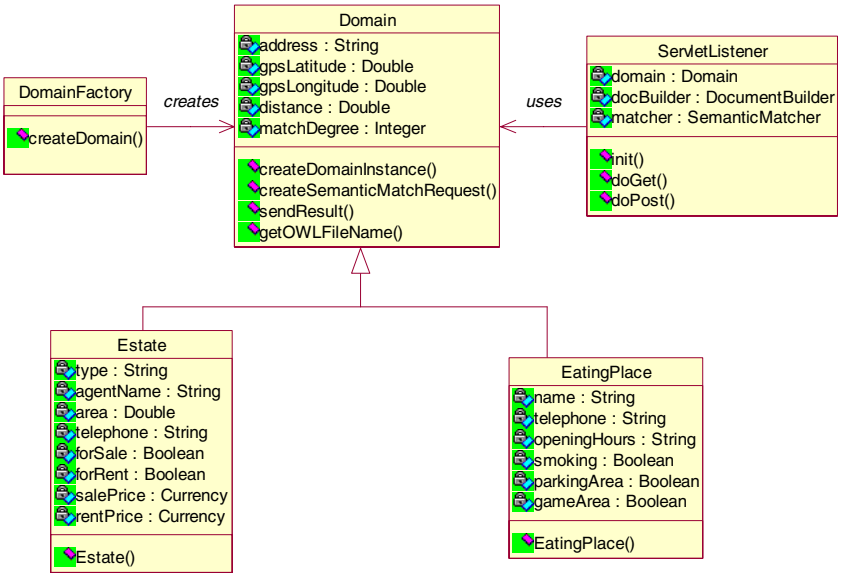


Fig. 2. Domain model of the SMOP to support extensibility in domain perspective

Domain is an abstract class which is extended by domain dependent wrapper classes to utilize the SMOP for new domains. For example, in Fig. 2, two subclasses of the *Domain* are given: *Estate* and *EatingPlace*. *Estate* is used for a "Real Estate Discovery System" in which mobile clients may search for geographically near real estates those match with clients' preferences – semantically appropriate with client's needs. On the other hand, *EatingPlace* instances represent eating places (like restaurants, cafes, patisseries, etc.) within an "Eating Place Discovery and Reservation" system in which semantic matching of eating place services with mobile clients' eating preferences is realized.

Apparently, it is enough to write such domain-dependent wrapper classes (subclasses of `Domain`) when a new domain is needed to be added into SMOP-based environment. Three abstract methods of *Domain*, called *createSemanticMatchRequest*, *createDomainInstance* and *sendResult* should be implemented within wrappers to handle domain knowledge.

createSemanticMatchRequest method receives related domain instance and domain type as input and returns its semantic match request counterpart which is processed by the engine during semantic match process. For example, eating place preferences of a mobile client are encapsulated within an *EatingPlace* object in a system. By calling this object's *createSemanticMatchRequest* method, controller servlet retrieves those preferences in RDF triples to form service match request and hence, they are compared with the advertised service capabilities by the built-in semantic matching engine.

Implementation of the *createDomainInstance* provides reverse information flow inside the system. As it is discussed in the architecture section, SMOP's semantic service capability matching engine –called Semantic Matching Engine- returns

semantic match results in a collection of *SemanticMatchResult* objects. Those objects store matched service advertisements as OWL individuals with their match degrees and GPS data. Those ontological result properties should be converted into the domain-specific attributes of the wrapper class. So, they can be processed by controller servlets and transferred into the mobile clients. This match result – domain class conversion is realized by calling *createDomainInstance* method of the related domain object.

sendResult method implementation provides transmission of domain instance content into mobile clients in XML format. Controller servlet sends those XML representations of the semantic query results to the view components of SMOP residing on the mobile phone. Those software components also don't need to be re-written in case of a domain change. Because, as it is mentioned in the previous section, "User Interface Generator" module of the mobile software initially retrieves concepts belonging to a specific domain's ontology from controller servlet also as XML data –they are not hard coded in GUI components- and it dynamically creates the visual interface. During semantic query communication, SMOP's mobile GUI components only need to parse received XML data and print out the content into the phone's screen in appropriate to the domain's desired format.

On the other hand, the constructor method of the wrapper class should be implemented in a way that attributes are set with the values which are retrieved from request XML document. This request document is received from the mobile client.

Runtime employment of wrapper classes is realized by applying *Factory* creational design pattern [10]. Initially, *DomainFactory* processes a document in which various domain definitions are specified and it creates desired domain-related class instance to be used within the controller servlet during system interactions. Notice that application of the Factory pattern and `Domain` class abstraction in software design provide the representation and use of the above mentioned domain-related wrapper classes in the system in a completely domain independent way; because wrappers are always referenced over their superclass (*Domain*) inside the software.

4 Case Study and Evaluation

As a case study, a mobile services application for the real estate domain has been designed and deployed on SMOP. In order to realize such an application environment, we have first defined a concept named as "Estate" to advertise the places for sale and/or rent to the internal matching engine of the SMS. One of the properties of this concept is the "type" which takes value from an OWL ontology shown in Fig. 3. This property is used during semantic matching process to find out semantically related real estates with user's request. The other critical properties are "for sale" and "for rent" to define a real estate is for sale, for rent or both. These properties are the additional options that users can specify in addition to real estate type. "GPS latitude" and "GPS longitude" properties define the geographic position of the advertised real estate. The other properties are address, area, sale price, rent price, agent name and telephone.

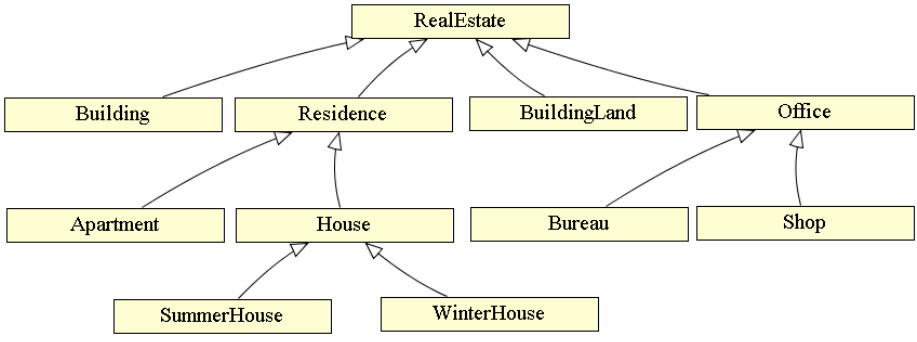


Fig. 3. An example *Real Estate Type* ontology to show the taxonomy of instances in a domain

We mapped a simplified version of *Real Estate Type* ontology to XML format and stored in the server. We also prepare an XML document representing of "for sale" and "for rent" properties of "Estate" concept and possible values of these properties. So, whenever the real estate domain is selected, these XML files are transferred to the mobile device for the creation of the visual interfaces at run-time. The XML document corresponding to the *Real Estate Type* ontology given in Fig. 3, is shown below:

```
<level1 type="Real Estate">
   <level2 type="Residence">
      <level3 type="House">
         <level4 type="Winter House"/>
         <level4 type="Summer House"/>
      </level3>
      <level3 type="Apartment"/>
   </level2>
   <level2 type="Building Land"/>
   <level2 type="Building"/>
   <level2 type="Office">
      <level3 type="Bureau"/>
      <level3 type="Shop"/>
   </level2>
</level1>
```

For a test scenario, we have created six instances of real estate concept and advertised to the SMS. Some important properties of these instances are shown in Table 1. Notice that, "type" property takes value from predefined *Real Estate Type* ontology.

Table 1. Instances of "Estate" concept that are advertised to SMS

No	Agent Name	Type	For Sale	For Rent
1	Green House	*Building*	Yes	Yes
2	Homecity	*SummerHouse*	No	Yes
3	House&House	*House*	Yes	No
4	RE/MAXX	*WinterHouse*	No	Yes
5	RE/MAXX	*Office*	No	Yes
6	RE/MAXX	*House*	Yes	Yes

Mobile client components of the application software have been deployed on a Siemens SXG75 cell phone with built-in GPS receiver and Java 2 Micro Edition (J2ME) 1.1, Mobile Information Device Profile (MIDP) 2.0 and Connected Limited Device Configuration (CLDC) 1.1 support.

When the client application on mobile phone is started by the user, first of all, it connects to the server and gets the available service domains. Let us assume that the user selected the real estate domain, then following the taxonomy of real estate types within this domain, he/she selected *House* concept to find instances of this concept. And he/she also specifies that he/she seeks real estates for rent. Screen snapshots showing user's selection of his/her request are given in Fig. 4.

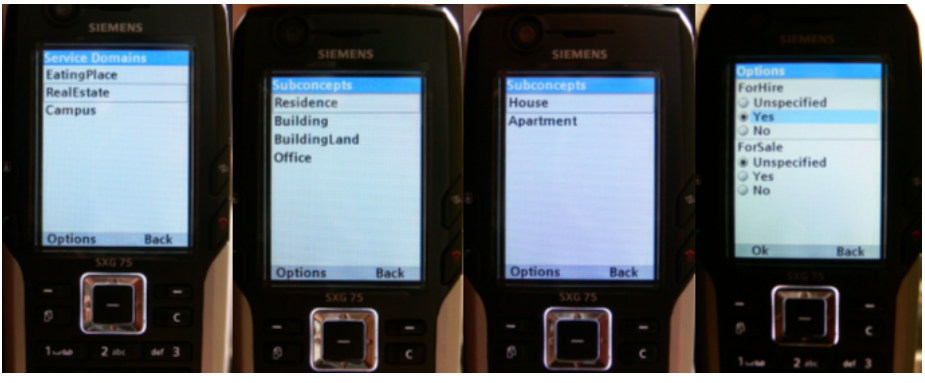


Fig. 4. Screen snapshots showing user's selection of his/her request

After user completes his/her request, the *House* concept, additional choices about the requested real estate and the position of the user are sent to the server. So, corresponding request XML document is shown below:

```
<request>
   <type>House</type>
   <forRent>Yes</forRent>
   <forSale>Unspecified</forSale>
   <gpsData>
      <latitude>22.333E</latitude>
      <longitude>52.444N</longitude>
   </gpsData>
</request>
```

Default value of "degree of match" parameter is given as *subsumes*. So, the server invokes the SMS with the given request. The SMS performs semantic matching in the following way: it excludes the first and fifth instances because their "degree of match" values are *fail*. It also excludes the third one because it isn't for rent. The sixth one has the *exact* "degree of match" value and second and fourth instances have the *subsumes* "degree of match" values. So, the SMS returns the matched real estate instances with their "degree of match" values in the following order: sixth, second, and fourth.

After the server receives match results from the SMS, it performs another sort operation on semantically equal match results regarding their distances to the user.

Although the second and fourth ones have the same “degree of match” values, the fourth one is closer than the second to the user. Hence, final list contains match results in the following order: sixth, fourth, second. Finally, the server creates an XML document that includes resultant domain instances and sends it to the mobile client. As an example, a part of the result XML document is shown below:

```
<matchResults>
   <result>
      <type>House</type>
      <degreeOfMatch>EXACT</degreeOfMatch>
      <distanceToClient>0,724km</distanceToClient>
      <agentName>RE/MAXX</agentName>
      <address>Bornova Street 1</address>
      <tel>2154585</tel>
      <forRent>Yes</forRent>
      <forSale>Yes</forSale>
      <area>200</area>
      <rentPrice>350€</rentPrice>
      <salePrice>25000€</salePrice>
      <gpsData>
         <latitude>27.229E</latitude>
         <longitude>38.455N</longitude>
      </gpsData>
   </result>
   <result>
      ...
   </result>
   ...
</matchResults>
```

The mobile device parses result XML document and creates a visual interface to show returned real estates to the user. The user can select one of the real estates from the list of match results to see detailed information about it. Snapshots showing list of match results and details of first two matched real estates are given in Fig. 5.

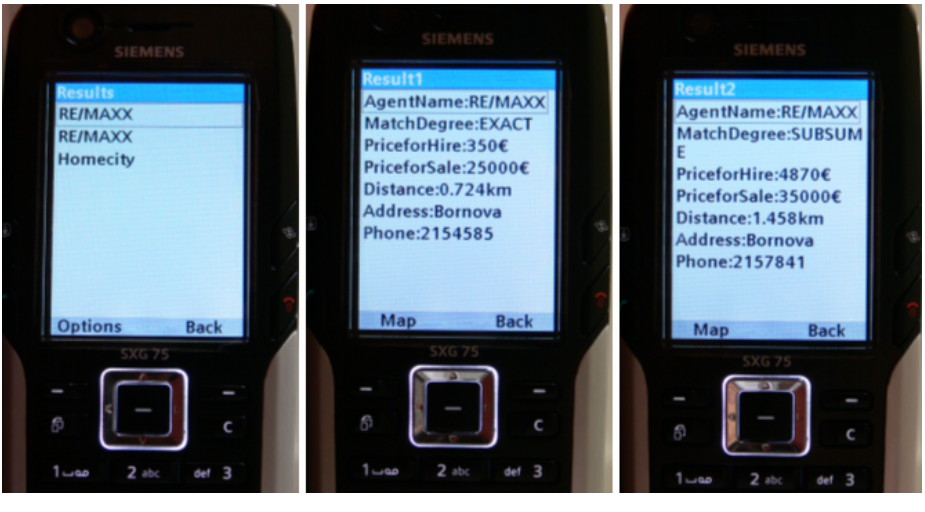


Fig. 5. Snapshots showing list of match results and details of first two matched real estates

The user may choose to see a satellite map which shows one of these locations' and his/her geographic position. In this case, the GPS data of both user and the real estate are sent to the server in XML format. As an example, corresponding XML document for the first result is shown below:

```
<locations>
   <user>
      <gpsData>
         <latitude>22.333E</latitude>
         <longitude>52.444N</longitude>
      </gpsData>
   </user>
   <domain instance>
      <gpsData>
         <latitude>27.229E</latitude>
         <longitude>38.455N</longitude>
      </gpsData>
   </domain instance>
</locations>
```

The server takes this XML document and invokes the MS. The MS marks the user and the real estate with different colors (blue for the user and red for the real estate) on a satellite map using its GIS. Then, this map is received by the mobile client through the server in binary format. Screen snapshot showing a satellite map enclosing first one of the matched real estates and the user is given in Fig. 6.

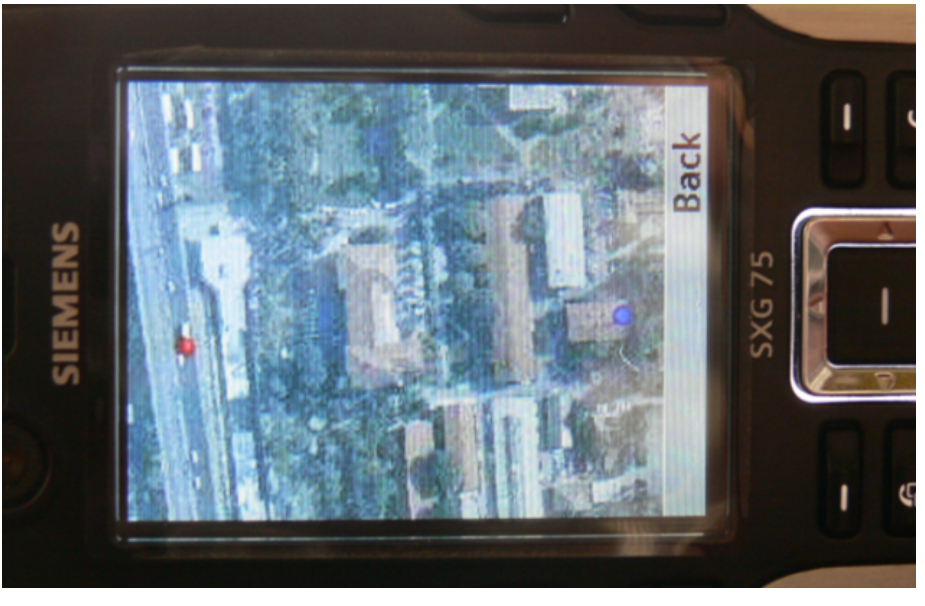


Fig. 6. Screen snapshot showing a satellite map enclosing first one of the matched real estates and the user. The blue point on the map marks current position of the mobile user while red point represents position of the real estate.

To demonstrate domain independency feature of SMOP, we have added a new domain, called eating place, to the above application environment. We didn't need to

change mobile client and platform web services tiers as we expected. On the other hand, we have naturally designed concept ontology of this new domain, added corresponding domain individuals into the SMS knowledgebase and prepared XML documents representing concepts of the domain ontology. Notice that, the wrapper class stands for this new domain has also been implemented by extending *Domain* abstract class as discussed in section 3. Hence the mobile client could interact with the SMS by means of this wrapper. Upon completion of above preparations, we examined that the environment successfully provides semantic web based information gathering on both domains for mobile clients.

5 Related Work

In the pervasive computing literature there are studies to develop software environments for location aware and context aware application development. Below, we will summarize some of these studies by comparing them with our system so that we can show in what ways SMOP is different from them.

Hessling et al. [1] devised an application, where the semantic user profiles which are stored in the mobile devices are matched against the semantic services broadcasted by stations. When the users with their mobile devices enter the range of a station, the station's services are matched against the semantic user profiles. Although the algorithm of how the user profiles are matched semantically is not given in detail in their paper, from the semantic matching point of view, the work of Hessling et al. [1] can be considered as the nearest one to our study. As we mentioned above, our aim is not to discover services, but to search the predefined semantic knowledge using semantic matching techniques so that mobile users can get the most relevant results for their queries. In addition, we focus on the extensibility of the architecture which is not considered in work of Hessling et al.

The Agents2Go [2] is an agent based distributed system that allows the creation of location dependent and service-based information systems. Although the proposed agent based architecture may allow the generation of new location dependent information systems, the extensibility perspective is not clear and not discussed in the paper. Also, a semantic matching engine, which makes it possible to extend the architecture by adding new service ontologies at run time, is not considered in Agents2Go.

Intelligent Computing group in University College Dublin developed some location dependent and context aware mobile applications like Gulliver's Genie [3] based on their Agent Factory framework. Their focus is in application development in space limited mobile devices and they have proposed an approach called as collaborative agent tuning [4] to incrementally develop such applications. Although Agent Factory framework and ACCESS context aware infrastructure [5] provide the necessary software architecture to implement agent based location dependent applications, extensibility in terms of adding new service domains at run time is not the focus of their works.

On the other hand, there are many classical location-based information search services in mobile environments commercialized by GSM operators. For example,

there are systems where users with mobile phones can be directed to the nearest local restaurants, shops, etc. These systems can be considered as standard information search services for mobile users. There are two features, which make SMOP different from them. The first feature is being domain independent based on an extensible software architecture. Supporting semantic matching is the second feature where the system that we have developed differs from them. Using semantic matching, a result list ranked by the degree of semantic match can be presented to the user in response to his/her request so that he/she can have the option of accessing to the most semantically related information. So, we take the previous works one step further by integrating semantic matching capability into the information gathering process in pervasive environments and by modeling the system in a way that it supports domain independency.

6 Conclusion and Future Work

We have introduced a mobile services environment in which semantic web based service capability matching and location-aware information gathering are used to develop mobile applications. The proposed architecture has been fully implemented and tested for different service domains.

The environment is adaptive for various mobile applications due to its domain independency and provides most relevant services for mobile users by applying semantic capability matching in service lookups. GPS and map service utilization cause to find near services in addition to capability relevancy and hence we believe this increases quality of the mobile services.

We currently work on to integrate mobile service execution into the environment. The system in use provides an enhanced service information gathering. However, invocation of remote services - except semantic service discovery and map services - by the mobile clients is not currently supported. Our aim is to provide an ultimate mobile services system in which semantic service discovery and execution are both fulfilled. In such a system, for example, a mobile client may first choose a relevant restaurant service and then reserve a table at this restaurant by only using his/her cell phone; or considering the real estate system given in this paper, the user may also arrange a meeting with the related estate agent after determination of the semantically most suitable and nearby estate.

Acknowledgements

We would like to thank members of Mobile Software Development Group in Ege University Computer Engineering Department (Seymen Ersen Diraman, Mehmet Niziplioglu, Oguz Karakus, Serkan Kaba, Murat Colak and Salim Asan) for their great effort in this study.

This study is partially funded by Ege University Scientific Research Projects Directorate with the project number 2003MUH039.

References

1. Hessling, A., Kleemann, T., Sinner, A.: Semantic User Profiles and their Applications in a Mobile Environment, In the Proc. of Artificial Intelligence in Mobile Systems 2004 (AIMS'04) In conjunction with UbiComp 2004, Nottingham, UK (2004)
2. Ratsiomor, O., Korolev, V., Joshi, A., Finin, T.: Agents2Go: An Infrastructure for Location-Dependent Service Discovery in the Mobile Electronic Commerce Environment, In ACM Mobile Commerce Workshop 2001, available at: http://research.ebiquity.org/v2.1/papers.
3. O'Grady, M. J., O'Hare, G. M. P., Sas, C.: Mobile agents for mobile tourists: a user evaluation of Gulliver's Genie, Interacting with Computers 17(4): 343-366 (2005)
4. Muldoon, C., O'Hare, G. M. P., O'Grady, M. J.: Collaborative Agent Tuning, ESAW'05, Kusadasi, Turkey (2005)
5. Muldoon, C., O'Hare, G. M. P., Phelan, D., Strahan, R., Collier, R. W.: ACCESS: An Agent Architecture for Ubiquitous Service Delivery, CIA 2003, pp. 1-15 (2003)
6. Paolucci, M., Kawamura, T., Payne, T. R., Sycara, K.: Semantic Matching of Web Services Capabilities, In the proc. of the first international semantic web conference (ISWC), Sardinia, Italy (2002)
7. Erdur, R. C., Dikeneli, O., Önal, A., Gümüs, Ö., Kardas, G., Bayrak, Ö., Tetik, Y. E.: "A Pervasive Environment for Location-Aware and Semantic Matching Based Information Gathering", Computer and Information Sciences - ISCIS 2005, Lecture Notes in Computer Science, Springer-Verlag, Vol. 3733, pp. 352-361 (2005)
8. Kardas, G., Gümüs, Ö., Dikeneli, O.: "Applying Semantic Capability Matching into Directory Service Structures of Multi Agent Systems", Computer and Information Sciences - ISCIS 2005, Lecture Notes in Computer Science, Springer-Verlag, Vol. 3733, pp. 452-461 (2005)
9. Buschmann, F., Meunier, R., Rohnert, H., Sommerlad P., Stal, M.: "Pattern-Oriented Software Architecture, Volume 1: A System of Patterns", John Wiley & Son Ltd, New York USA (1996)
10. Gamma, E., Helm, R., Johnson R., Vlissides, J.: "Design Patterns - Elements of Reusable Object-Oriented Software", Addison-Wesley, Massachusetts USA (1994)

Integrating Data from the Web by Machine-Learning Tree-Pattern Queries

Benjamin Habegger[1] and Denis Debarbieux[2]

[1] Dipartimento di Informatica e Sistemistica
Università di Roma 1 – "La Sapienza" – 00198 Roma, Italy
habegger@dis.uniroma1.it
[2] LIFL, UMR 8022 CNRS, Lille University (France)
Mostrare project, RU INRIA Futurs
denis.debarbieux@lifl.fr

Abstract. Effienct and reliable integration of web data requires building programs called wrappers. Hand writting wrappers is tedious and error prone. Constant changes in the web, also implies that wrappers need to be constantly refactored. Machine learning has proven to be useful, but current techniques are either limited in expressivity, require non-intuitive user interaction or do not allow for n-ary extraction. We study using tree-patterns as an n-*ary* extraction language and propose an algorithm learning such queries. It calculates the most information-conservative tree-pattern which is a generalization of two input trees. A notable aspect is that the approach allows to learn queries containing both child and descendant relationships between nodes. More importantly, the proposed approach does not require any labeling other than the data which the user effectively wants to extract. The experiments reported show the effectiveness of the approach.

1 Introduction

Providing an automated access to web data requires building wrappers. This is know to be both tedious and error prone. Past research on semi-automated wrapper construction ([5,8,2,11,6]) has mostly used string representations for both documents and learned patterns [7,9,5]. Other work (eg. Stalker), only learn *unary* patterns. Composing unary pattern can lead to n-ary extraction, but by requiring *non evident* extra user input. Our goal is to limit user input to examples of his target relation. In this context, existing work have only proven to be efficient in extracting tabular data. Much data on the web does not follow a tabular format. The natural tree structure of web documents can be used. This work is among the first using machine learning for information extraction explicitly using a tree-based document representation [2,8].

Tree patterns allow to directly express n-ary queries and, being closely related to the widely used XPath [12] language, are more easily readable by human users than other formalisms (eg. tree transducers). How tree patterns are defined has important consequences on matching and learning efficiency as well as expressivity. We propose a weight-based algorithm capable of generating a tree pattern in different settings. We particularly study ordered and unordered injective tree patterns in which child and descendant relationships can be expressed and propose an algorithm capable of learning

R. Meersman, Z. Tari et al. (Eds.): OTM 2006, LNCS 4275, pp. 941–948, 2006.

such patterns. Given two ordered tree representations of documents, we propose an algorithm computing a tree pattern which is maximal w.r.t. our weighting. Learning an ordered or unordered pattern is left to the user. Both cases are handled similarly and only require local adaptations. Here, we only consider patterns with injective.

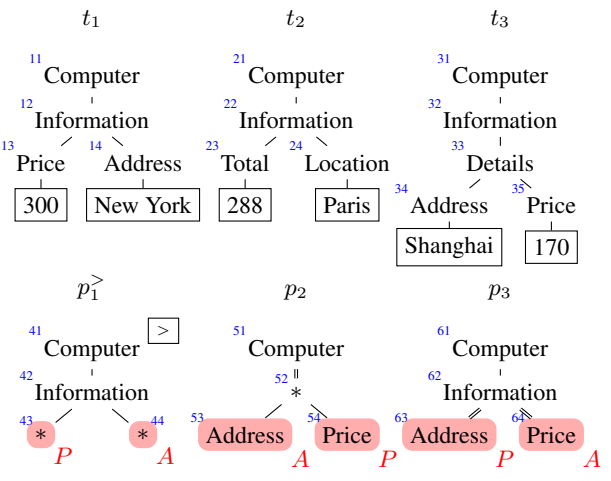


Fig. 1. Example trees and patterns

In figure 1, t_1, t_2 and t_3 represent a set of XML documents from which we wish to extract $(price, address)$ pairs ((P, A) for short). The ordered tree pattern $p_1^>$ extracts a couple (P, A) if P and A are two children of the same node and such that P is found before A independently of their labels. The relative positions allow to determine to which attribute the extracted values belong. However, $p_1^>$ is not adapted to extract information from tree t_3 address and price are inversed. Unordered tree patterns, such as p_1 and p_2 (a double line represents a descendant edge) allow to correctly extract from t_1 and t_3. To distinguish them we define a weighting over tree patterns based on three parameters : the number of *labels*, *descendant edges* and *child edges* in the pattern. Injectivity can also be used. The embedding from p_3 to t_3 is not injective since the least common ancestor (lca) of node 63 and 64 is 62 and 32 (the node where 62 is mapped to) is not the lca of 34 and 35 (where resp. 63 and 64 are mapped to). However, the embedding from p_2 to t_2 and from p_2 to t_3 are injective.

In this paper, we introduce (1) a flexible notion of weighted patterns and (2) an algorithm allowing to generalize n-ary extraction patterns adaptable to different settings. (3) We study learning tree patterns with our algorithm in different settings and evaluate it on different datasets. **No preprocessing** and **no intermediate labelling** is required. Also tree-patterns, since they allow for query rewriting, are more adapted to data integration than other wrapper languages.

This paper is organized as follows. Section 2 presents the tree pattern background. Section 3 details our generalization and its implementation. Its evaluation is presented in section 4. Section 5 presents related work and, finally we conclude in section 6.

2 Background

Definition 1 (Tree Pattern). *A* tree pattern p *(resp.* $p^>$*) is an* **unordered** *(resp.* **ordered***) unranked tree over alphabet* $\Sigma \cup \{*\}$ *with a distinguished subset of edges called* descendant edges*, and a* k*-tuple of nodes called* result tuple*, for some* $k \geq 0$.

Note that a document is an ordered tree pattern in which the set of *descendant edges* and the *result tuple* are empty. For an (unordered or ordered) pattern p, we denote by `d-edge` an edge of the set *descendant edges* and by `c-edge` (child-edge) the other edges. $CEDGES(p)$, $DEGEDES(p)$ and $NODES(p)$ resp. denote the sets of c-edges, d-edges and nodes of p. $ROOT(p)$ denotes the root of p. For a given node n, $LABEL(n)$, $PARENT(n)$ and $CHILDREN(n)$ resp. denote the label, parents and children of n. Furthermore, we will denote by $<$ the ordering obtained by walking thru the nodes of p in a depth-first manner. For any two nodes n and m, $n \prec m$ denotes that n is a strict ancestor of m. Finally, for a pattern p, each node of the *result tuple* is marked by a variable. $VARIABLES(p)$ denotes the set of variable nodes contained in p, and for a node n, $VAR(n)$ is the name of the attached variable, if any. A c-edge is drawn with a single line and a d-edge is drawn with a double line.

Consider the pattern p_2 of figure 1. Given that the numbers on the top left corner identify each node uniquely we have the following : $NODES(p_2) = \{51, 52, 53, 54\}$, $ROOT(p_1) = 51$, $CHILDREN(52) = \{53, 54\}$, $DEDGES(p_2) = \{(51, 52)\}$, $CEDGES(p_2) = \{(52, 53), (52, 54)\}$, $PARENT(54) = 52$, $LABEL(52) = *$, $LABEL(51) = \text{Computer}$, $VARIABLES(p_2) = \{53, 54\}$, $VAR(53) = A$.

Definition 2 (Embedding). *An function e from an unordered tree pattern p to a (ordered or unordered) tree pattern p' is an* unordered embedding *iff :*

(1) e is root preserving (ie. $ROOT(p') = e(ROOT(p))$)
(2) for all $n \in NODES(p)$, $LABEL(n) = *$ or $LABEL(n) = LABEL(e(n))$
(3) for each $n, m \in NODES(p)$:
- *if (n, m) is a c-edge in p then $(e(n), e(m))$ is an c-edge in p'*
- *if (n, m) is a d-edge in p then $e(m)$ is a proper descendant of $e(n)$ in p'*

(4) for each $n \in VARIABLES(p)$, $VAR(h(n))$ is defined and $VAR(h(n)) = VAR(n)$

Definition 3 (Ordered Embedding). *An function e from an ordered tree pattern $p^>$ to an ordered tree pattern $p'^>$ is an* ordered embedding *iff :*

(5) e is an unordered embedding
(6) for each $n, m \in NODES(p^>)$: if $n < m$ then $e(n) < e(m)$

Definition 4 (Extracted tuples). *Given an unordered (ordered) tree pattern p and a tree t. Let $(v_1, \ldots, v_n)$ denote the variables of p. The extracted relation from t given p is the set $R_p(t) = \{(e(v_i), \ldots, e(v_n))$ where e is an (ordered) embedding from p to t $\}$*

Definition 5 (Injective embedding). *An embedding e from a pattern p to a tree t is* injective *iff it also satisfies the following requirements :*

(7) $\forall n, m \in NODES(p), n \neq m \Rightarrow e(n) \neq e(m)$
(8) $\forall n, m \in NODES(p), e(lca(n, m)) = lca(e(n), e(m))$

where $lca(x, y)$ is the lowest common ancestor of x and y.

Proposition 1. *Let p and p' be two patterns such that there exists an injective (ordered or unordered) embedding e from p to p'. Then $|CHILDREN(ROOT(p))| \leq |CHILDREN(ROOT(p'))|$.*

Definition 6 (Least general generalization). *Given two patterns p_1 and p_2, a least general generalization (lgg) of p_1 and p_2 is a pattern p for which there exists an embedding from p to p_1 and another from p to p_2. Moreover, there exists no other pattern p' respecting the previous condition and such that an embedding from p to p' exists.*

3 Maximal Weight Generalization

As we will see here two tree patterns may have many different generalizations. We introduce a notion of weighted pattern to distinguish between them allowing to partly handle the non uniqueness of an lgg in the case of injective embeddings. It also allows for some preference on the type information to be used when characterizing the information to be extracted. The measure we propose will help our algorithm to distinguish between patterns but is not total since two different patterns can have the same weight.

Definition 7 (Weighting function). *Let Δ be the set of patterns. A* weighting *is a function $\mathcal{W} : \Delta \to \mathbb{R}^+$ which associates a positive real number to each pattern in Δ.*

The weight $w(n)$ of a node n of a pattern p is the weight of the subtree rooted at this node. In this article, we work with a specific $\mathcal{W}$ defined as $\mathcal{W}(p) = w_{ch}.|CEDGES(p)| + w_{dc}.|DEDGES(p)| + w_{ex}.|var(p)| + \sum_{a \in \Sigma} w_{lbl_a}.|nodes(p)|_a$. w_{ch}, w_{dc}, $\{w_{lbl_a}, a \in \Sigma\}$ and w_{ex} are parameters of the problem. These are parameters to our algorithm which can be set by using predefined default settings or changed by the user. $VAR(p)$, $CEDGES(p)$, $DEDGES(p)$ have already been defined (see page 943) and $|NODES(p)|_a$ denotes the number of nodes of p labelled by a. In our examples, all the labels have the same weight w_{label}. Formally, tree pattern generalization can be defined as follows :

Definition 8 (Tree pattern generalization (TPG) problem)

Input *$\mathcal{W}$ a weighting function , p_1 and p_2 two tree patterns and an integer k.*

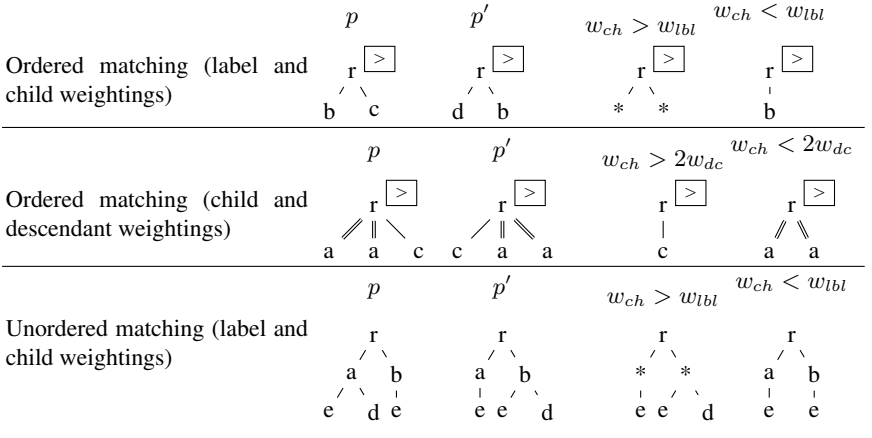


Fig. 2. Examples of the influence of the weights

Input: A parent candidate $s(x, y)$ and a candidate slot (x', y')
Output: A candidate child $s(x'', y'')$ which maximizes the score of $s(x, y)$ for slot (x, y)

Let X and Y be resp. the indexes of the non-child descendants of x and y
Let $(x'', y'') \leftarrow (x, y)$
Let $bs \leftarrow weight(s(x, y))$ % bs as best score
if (x, x') and (y, y') are both child edges in their respective patterns **then**
  $bs \leftarrow bs + w(child)$
else
  $bs \leftarrow bs + w(descendant)$
end if
for all $k \in X$ **do**
  if $weight(k, y') + w(descendant) > bs$ **then**
    Let $(x'', l'') \leftarrow (k, y')$ and $bs \leftarrow weight(k, y') + w_d$
  end if
end for
for all $l \in Y$ **do**
  if $weight(x', l) + w(descendant) > bs$ **then**
    Let $(x'', y'') \leftarrow (x', l)$ and $bs \leftarrow weight(x', l) + w(descendant)$
  end if
end for
return $s(x'', y'')$

Algorithm 1. Calculate the best child candidate candidate for a slot

Output *true iff there exists an unordered tree pattern p s.t. there exists an injective embedding from p_1 to p, an injective embedding from p_2 to p and $\mathcal{W}(p) \geq k$.*

As shown by the following examples, this problem is not trivial. Let us consider, on one hand trees t_1, t_2 and t_3, and on the other, patterns p_1, p_2 and p_3 from figure 2. First, (a) shows that, depending on the respective weights associated with child edges and to labels, the generalization of p and p' is either a pattern with two unlabeled children or a pattern with a child labeled by b. Similarly, (b) shows the influence of the relationship between weight of child edges and those of descendant edges. Finally, (c) extends example (a) to illustrate similar difficulties in unordered pattern. Not very surprisingly, we show that this problem is NP-hard by reduction from the maximum common embedded sub-tree problem. However, we propose an algorithm which solves the tree pattern generalization problem, and in a restricted case, does so in polynomial time.

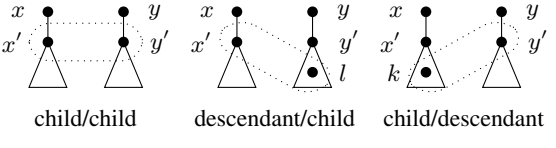


Fig. 3. Choosing the best candidate $s(x'', y'')$ as a child of $s(x, y)$ for slot (x', y').

Our generalization algorithm, can be split into two steps recursively referring to each other : (i) calculate the maximal weight generalizations of the sub-patterns and (ii) find

the best combination of sub-patterns respecting injectivity constraints (as well as order constraints when required). Given x a node of p_1 and y a node of p_2 the problem is to compute a new pattern $p(x, y)$ rooted at a node called $s(x, y)$ s.t. $p(x, y)$ is the maximal weight generalization of pattern p_1 rooted at x and pattern p_2 rooted at y. Of course, the maximal weight generalization p of two patterns p_1 and p_2 is the pattern $p(root(p_1), root(p_2))$. If x and y are leaves, the problem is quite easy. If x or y is an internal, we have to use weight of its subtrees. As shown by fig 3, only tree configurations are possible. Many cases have to be study (in particular for the case 'child/descendant' and for the case 'descendant/child'). So we use algorithm 1 to calculate the best choice by using dynamic programming. As proposition 1 holds, for each couple of nodes (x, y) we know the number of children of $s(x, y)$. However, for efficiency reasons, a maximal number of children c may have been set as a parameter. If it happens to be lower, than the number of children the pattern would effectively have, then the number of kept children is limited to c. By classic method of dynamic programming, it is easy to build the maximal weight pattern from the dynamic programming table. When $LABEL(x) = LABEL(y) = a$, $s(x, y)$ is labeled with a, or else it is labeled $*$. Similarly, when $VAR(x) = VAR(y) = X$ then $s(x, y)$ is attached X as variable and has no variable otherwise. Remember that one of the important contributions of our approach, is that descendant edges may appear in the learned patterns. This happens when the sub-pattern conserved by introducing a descendant edge is more interesting than keeping a child edge which generates many miss-matches in the underlying subtrees.

4 Evaluation

We have implemented our algorithm in Ocaml and evaluated it on different datasets. We proceeded to an evaluation under both unordered injective embeddings and ordered injective embeddings. In each evaluation, we did a 5-fold cross validation taking one set of documents as example set and the remaining for testing.

Table 1. Experimental results with unordered and ordered embeddings

	Unordered					Ordered				
Source	**Good**	**Wrong**	**Miss.**	**Rec.**	**Prec.**	**Good**	**Wrong**	**Miss.**	**Rec.**	**Prec.**
bigbook	3468	0	0	1.	1.	3444	0	0	1.	1.
okra	2598	0	0	1.	1.	2691	0	0	1.	1.
s20	253	9052	0	1.	0.03	247	4307	0	1.	0.05
L0-0	147	0	0	1.	1.	149	0	0	1.	1.
L3-0	139	0	0	1.	1.	150	0	0	1.	1.
L8-0	142	1232	0	1.	0.10	144	641	0	1.	0.18
L9-0	142	0	0	1.	1.	0	137	137	0.	0.
pagesjaunes.fr	101	0	9	0.92	1.	101	0	9	0.92	1.
amazon.com	79	0	0	1.	1.	79	0	0	1.	1.

The first three datasets Bigbook, Okra and S20 come from the RISE information extraction repository. These two datasets are the most referenced sets in the information

extraction community. They are however, known to be easy. The target relations we have used for our experiments were $bigbook(name, address)$, $okra(name, mail, score)$ and $s20(file, score, size, type)$. The L0-0, L3-0, L8-0, and L9-0 datasets, are artificial datasets made available by Marty[1]. All these datasets are constructed over the same data (a 13-ary relation) but with varying layouts trying to cover multiple complex presentations. We only present results for the datasets for which tree patterns over the relations considered in this paper have sufficient expressivity. While we plan in future work to add other relations, the expressivity required to cover all those case will likely require an unreasonable number of examples. Finally we evaluated our approach on two real world dataset : Amazon DVD listings and Pagesjaunes address entries where the target relations are $pagesjaunes(name, address, city)$ and $amazon(title, price)$. .

Table 1 presents the results obtained in both unordered and ordered injective settings. In all cases we have a very high if not perfect recall. The results for Amazon and Pagesjaunes are particularly encouraging, since these datasets come from existing web sites. The bad precision shows that in some cases erroneous extractions occurred. In all these cases it appears that the data have a linear format (ie. all the tuples follow each other under a single node). In such cases, no suitable pattern can be defined over child and descendant relations alone. Working with such relations alone is therefore not sufficient or that disjunctions of tree patterns (and therefore negative examples) might be required. Many approaches to learning disjunctions of conjunctions exist and we are looking into adapting our algorithm to add other relations (eg. next-sibling). The similarity between the results considering order or not suggest that respecting ordering does not provide much information on these sources. It should be noted that the ordering considered is similar to linking siblings together with a following-sibling relation. We believe that this ordering may not be strong enough to be informative and that a next-sibling relation might be an interesting replacement. In all our experiments we used the same fixed set of weights for the patterns and the same child limit number. The quality of the results suggest that defaults could easily be set and therefore the user would not require setting the parameters. The results given in this section show the effectiveness of our approach. It also shows that the extra labelling required by other approaches is not necessary to obtain similar performance.

5 Related Work

Much research on wrapper learning has been lead in the past years and many systems [7,5,8,4,10,3] have been proposed. Few systems allow to build n-ary extraction patterns by requiring *only* to label the target values. In Lixto[1], Stalker[7] or Squirrel, where patterns are *monadic*, the user is required to label *intermediate* nodes and building a pattern for them.

To our knowledge, only IERel [5] and WIEN [9] are capable of directly learning n-ary extraction patterns. They are string-based and rely on the strong hypothesis that the data to be extracted is tabular (ie. the tuples to be extracted do not overlap). However, the currently mostly referenced datasets (RISE) are either two simple for web

[1] http://www.grappa.univ-lille3.fr/ marty/

information extraction or require natural language information extraction techniques. Therefore there is no real basis for comparison.

6 Conclusion

We presented a tree-based approach to n-ary wrapper generation using tree patterns as the extraction language. Both the theoretical and practical aspect of our approach have been presented. An implementation allowed us to evaluate the approach and show that it is useful in many cases. In particular taking a tree view has allowed to overcome some limits of string-based approaches. Only examples are required as user input with similar performance as existing approaches.

We plan to extend the algorithm to include other types of relational constraints (eg. next-sibling and following-sibling) allowing to improve expressivity. We also plan to introduce negative examples to limit over-generalizations. Also less restrictive heuristics, but keeping the problem tractable, will be studied.

References

1. R. Baumgartner, S. Flesca, and G. Gottlob. Declarative Information Extraction, Web Crawling, and Recursive Wrapping with Lixto. In *Proc. of the 6th Int. Conf. on Logic Programming and Nonmonotonic Reasoning*, LNCS/LNAI, pages 21–40, 2001. Springer Verlag.
2. J. Carme, A. Lemay, and J. Niehren. Learning Node Selecting Tree Transducer from Completely Annotated Examples. In *Int. Conf. on Grammar Induction*, pages 29–102, 2004.
3. R. Gilleron, P. Marty, M. Tommasi, and F. Torre. Adaptive Relation Extraction from Semi-Structured Data. In *6èmes Journées Francophones "Extraction et Gestion des Connaissances"*, 2006.
4. S. Gupta, G. Kaiser, D. Neistadt, and P. Grimm. DOM-based Content Extraction of HTML Documents. In *Proc. of the 12th WWW Conference*, 2003. Elsevier Science.
5. Benjamin Habegger and Mohamed Quafafou. Context generalization for information extraction from the web. In Proc. of the ACM/IEEE Web Intelligence Conference, 2004.
6. C. Hsu and M. Dung. Generating Finite-State Transducers for Semi-Structured Data Extraction from the Web. *Information Systems*, 23(8), 1998.
7. C. Knoblock, K. Lerman, S. Minton, and I. Muslea. Accurately and Reliably Extracting Data from the Web : A Machine Learning Approach. *Data Engineering Bulletin*, 23(4), 2003.
8. R. Kosala, M. Bruynooghe, J. V. den Bussche, and H. Blockeel. Information Extraction from web documents based on local unranked tree automaton inference. In *Proc. of the 18th Int. Joint Conf. on Artificial Intelligence (IJCAI-2003)*, pages 403–408, 2003.
9. Nicholas Kushmerick. Wrapper induction: Efficiency and expressiveness. *Artificial Intelligence*, 2000.
10. K. Lerman, C. Knoblock, and S. Minton. Automatic Data Extraction from Lists and Tables in Web Sources. In *IJCAI-2001 Workshop on Adaptive Text Extraction and Mining*, Seattle, Washington, August 2001.
11. I. Muslea, S. Minton, and C. A. Knoblock. Hierarchical Wrapper Induction for Semistructured Information Sources. *Autonomous Agents and Multi-Agent System*, 4(1-2), March 2001.
12. XML Path Language (XPath), 1999. Available at http://www.w3.org/TR/xpath.

HISENE2: A Reputation-Based Protocol for Supporting Semantic Negotiation

Salvatore Garruzzo and Domenico Rosaci

DIMET, Università Mediterranea di Reggio Calabria
Via Graziella, Località Feo di Vito
89060 Reggio Calabria, Italy
{salvatore.garruzzo, domenico.rosaci}@unirc.it

Abstract. A key issue in open multiagent systems is that of solving the difficulty of an agent to understand messages coming from other agents having different ontologies. *Semantic negotiation* is a new way of facing this issue, by exploiting techniques that allow the agents of a MAS to reach mutually acceptable agreements on the exchanged terms. The produced scenario is similar to that of human discussions, where human beings try to solve those situations in which the involved terms are not mutually understandable, by *negotiating* the semantics of these terms. The HISENE approach is a recent JADE-based protocol effectively supporting semantic negotiation. It is based on the idea that an agent that does not understand a term can automatically require the help of other agents that it considers particularly reliable. However, HISENE does not take in account either the possibility of wrong answers coming from the requested agents or the fact that a term can have different meanings. In order to cover these two important issues, in this paper we present an extension of HISENE, called HISENE2, and we show experimentally that it performs better than HISENE with respect both to the quality and the efficiency of the semantic negotiation.

1 Introduction

The widespread diffusion of information agent technology and the exponential growth of different approaches in the field of open multiagent systems have produced a particular interest towards those communication problems caused by the heterogeneity of the different agent's knowledge models. These models, usually called *ontologies*, are often advocate as a complete solution for knowledge sharing between agents, giving the possibility to assign a *meaning* to terms contained in the exchanged messages. However, such a possibility exists only in the case each agent of the system knows the ontology of other agents; on the contrary, an agent that receives a message from another agent having a different ontology is not able to understand the content of the message, therefore the communication fails. A straightforward solution to such a problem is represented by the use of a unique common ontology, shared by all the agents of the system; however, this is a solution that appears unlikely in open multiagent systems, since it would imply that all the agents agree to adopt a standard ontology about which it is

R. Meersman, Z. Tari et al. (Eds.): OTM 2006, LNCS 4275, pp. 949–966, 2006.

necessary to reach consensus. Heterogeneity is due to the fact that each agent has its particular viewpoint of the world in which it performs its activity, and it does not necessarily desire to adopt a standard viewpoint that is not specifically designed for it. Another solution could be that of *ontology alignment*, that consists in providing each agent with a mapping between the other different ontologies. Unfortunately, in an open multiagent system, ontologies change continuously and it is not possible to define the mapping a priori, before the beginning of the agent interactions.

A possible way of facing the difficulties of an agent in understanding the messages coming from other agents having different ontologies is offered by the *semantic negotiation*. This is a process by which the agents of a MAS try to reach acceptable definitions mutually (i.e., mutual acceptable agreements on terms). The produced scenario is similar to that of human discussions, where human beings try to solve those situations in which the involved terms are not mutually understandable, by *negotiating* the semantics of these terms. Such a negotiation implies several operations performed by the two interlocutors as, for instance, a query that one of them could pose to have a description of a not understood term, a response provided by the other interlocutor containing the requested description, etc.

In [6] we introduced the idea that two agents involved in a communication process can require the help of other agents in order to solve possible understanding problems. In this context, the notion of *expertise* of an agent represents a measure of the capability of the agent to explain not understood terms to every other agent. Moreover, we also defined the notion of *understanding capability* of an agent a with respect to another agent b, that measures the capability of a to explain terms that b does not understand. Therefore, the expertise of an agent a is the capability of a to explain not understood terms to the whole community effectively, while the understanding capability with respect to b is relative only to the agent b. These two notions allow the possibility to introduce the synthetic measure of *negotiation degree*, defining the potential capability of a to negotiate the semantic of terms belonging to b. Therefore, in that framework, an agent can ask help to other agents to understand a term on the basis of their negotiation degree; for this purpose, it groups the agents in different *partitions* $p_1, p_2, \ldots, p_n$, ordered by a decreasing level of negotiation degree. On the basis of this idea, we proposed a semantic negotiation protocol, called *HIerarchical SEmantic NEgotiation* (HISENE), that is suitable to be applied for implementing such a semantic negotiation in the standard Java Agent DEvelopment Framework (JADE) [8]. An important advantage that this protocol introduces is that each agent can contact the other agents in different stages, by following the rational criteria of first negotiating with the agents belonging to the partition p_1, contacting the agents of the partition p_2 only if none of the agents in p_1 is able to answer positively, then contacting agents of the partition p_3 only if none of the agents in p_2 succeeds, and so on. Moreover, in order to understand unknown terms, each contacted agent can start another semantic negotiation in its turn; however, in order

to avoid the presence of a loop, each term is processed only once by each agent. This leads to the use of network communication resources in an efficient way.

It is worth to point out that in HISENE, both the expertise and the understanding coefficient of an agent, representing a sort of agent's *reputation*, are built on the basis only of the capability of the agent to explain unknown terms to other agents. The approach does not take in account the possibility that the *explanation* of a term provided by an agent a to another agent b could be uneffective, in the sense that, when b uses that explanation later in interacting with another agent, say c, it could be misunderstood. This possibility can arise for two different reasons, namely: *(i)* the explanation provided by a can be wrong, since a's ontology can contain mistakes; *(ii)* the explanation can be relative to a meaning of the involved term that is commonly used by a but that is unknown to c. The HISENE approach previously described does not deal with the issue *(i)*, since it assumes that each explanation provided is intrinsically correct; therefore HISENE is limited to cope only with this scenario and it is less effective in the more general case of possible incorrectness in the ontologies. Moreover, HISENE does not even consider the issue *(ii)*, and this is a very important limitation since a term can generally have different meanings in a multiagent system.

In this paper, we propose HISENE2, an evolution of the protocol HISENE. This new protocol presents a new mechanism to determine the reputation of an agent, based on the following idea. Suppose that an agent a explains the meaning of the term "volume" to another agent b, stating that a "volume" is a "book". Suppose that b understands the term "book" and therefore it is now able to understand "volume". Consequently, b includes the explanation of the term, that we here denote simply as $volume^a = book$, in its own ontology, by labelling the term with the identifier of the agent that provides the explanation, i.e. a. Moreover, b assigns a *context* to $volume^a = book$, defined as the set of agents that agree on this explanation; initially this context is, naturally, {a,b}. Finally, b assigns a *confidence* coefficient (belonging to the interval [0,1]) to $volume^a = book$, initially set to 1, meaning that b believes this explanation completely. Now, suppose that later b interacts with another agent c by using the explanation $volume^a = book$ provided by a. It is possible that c has the same explanation in its ontology, therefore it understands the meaning of the term and, consequently, b adds c to the context of the explanation. It is also possible that c does not understand the explanation $volume^a = book$, since it has, for instance, only another explanation for volume, that is $volume^a = quantity$. In this case, c can run a semantic negotiation with the other agents that it knows to acquire, if possible, the explanation $volume^a = book$. If the semantic negotiation succeeds, c understands $volume^a = book$ and communicates this fact to b that adds c to the context of the explanation. But it is also possible that the semantic negotiation performed by c fails. In the latter case, c is not able to understand the explanation c, in spite of its effort to find it in other agent ontologies, and it communicates this fact to b. As a consequence, b does not add c to the context of $volume^a = book$, and also decreases the confidence coefficient of the explanation, since it believes less in its reliability.

This simple situation shows that the confidence assigned to an explanation is dynamically determined in HISENE2 by observing the success of the explanation in different interactions. The *reputation* of an agent is computed on the basis of the quality of all the explanations it provided in the past. With respect to HISENE, HISENE2 improves the capability of an agent to determine good partitions of reliable agents to perform an effective semantic negotiation. Moreover, the introduction of the notion of *context* for an explanation allows an agent to register different meanings of the same term and also to remember what other agents agree with each meaning. This leads the agent interacting with another agent to use the more suitable meaning of a term.

We tested experimentally the effectiveness of HISENE2 by comparing it with HISENE in the scenario of a wide multiagent environment implemented by the JADE framework. These experiments show that the understanding degree of the whole agent community improves drastically by using HISENE2. Instead another experiment shows that the heterogeneity of the agent ontologies of the community decreases in time with the increase of the exchanged messages. Finally, by a third experiment, we show that the number of semantic negotiation messages is significantly smaller with respect to the case of HISENE, as a consequence of the more appropriate agent partitions built by using HISENE2.

The plan of the paper is the following: Section 2 describes some related work; Section 3 gives some preliminary notions on the JADE framework, while Sect. 4 describes the reputation-based partitioning; Section 5 deals in detail with the HISENE2 protocol, while Sect. 6 describes a simple example of how HISENE2 works; Section 7 shows the evaluation experiments, and finally, Sect. 8 draws some final conclusions.

2 Related Work

Semantic negotiation is a relatively recent research field. It involves some approaches known as *ontology negotiation* [11,13,14], a term coined in [1], and several other models and protocols dealing with the problem of negotiating the semantics of the terms contained in agents' messages [2,4,7,12,14]. However, most of these approaches are not able to support semantic negotiation without requiring agents either to share knowledge or to use a global common ontology, and none of them provides a semantic negotiation protocol that allows the whole agent community to contribute to the semantic understanding process between each agent pair. For example, the *common shared ontology*approach described in [12] provides the agents with a set of shared concepts, in which they can express their private knowledge. The communication vocabulary is formalized as an ontology, shared by the entire MAS, and in which every private concept of each individual agent can eventually be defined. Concept names used in an agent's private ontology, are not understandable to other agents. However, their definitions in terms of ground concepts are understandable. The use of definition terms, instead of concepts, enables optimal communications between agents. Moreover,

the approach presented in [2] introduces a computational framework for the detection of ontological discrepancies between two agents in multiagent systems. In this method, presuppositions expressed in a common vocabulary are extracted from the sender's messages, and compared with the recipient's ontology, which is expressed in type theory. Discrepancies are detected by the receiving agent if it notices type conflicts, particular inconsistencies or ontological gaps. Depending on the kind of discrepancy, the agent generates a feedback message in order to establish alignment of its private ontology with the ontology of the sender. The dialogue framework is based on a simple model of interaction. Another approach using a common knowledge is that presented in [14], where authors introduce a machine learning methodology and algorithms for multiagent knowledge sharing and learning in a peer-to-peer setting. Agents can use a set of shared concepts in which they can express their private knowledge. The work [7] proposes to consider the use of shared keys to solve the problem of using different names for the same object; in particular, a probabilistic matching approach is introduced. Semantic negotiation is described as a process by which a client and a service can negotiate mutually shared references. There are some other approaches that do not require the use of a shared ontology. As an example, in [4], to allow agents to interoperate, authors have developed a matchmaking system that, rather than requiring agents to share ontologies, exploits an agent-independent, domain-specific ontology, called a *global ontology*. When an agent joins the platform, the proposed system applies an information-extraction engine to the agent's code to extract useful information, that includes recognized names of concepts the agent uses (e.g. class names, parameter names, etc.). Instead of having a shared ontology, the proposed system maintains a mapping of the local ontologies of all agents to the independent global ontology. The main difference between this approach and a shared ontology approach is that an agent's programmer does not need to know anything about any other agent's local ontology, nor does he need to know about the global ontology, but it is the system that does the necessary mapping. In [6], agents do not need to share either a common ontology or to maintain a global ontology, in order to understand each other, but they try to solve their understanding problems availing the help of other agents that are considered experts in the involved domain and that have similar ontologies. Obviously, by using this approach, the understanding can be obtained only by waiting that the agent community evolves in time, allowing the formation of expert agents and understanding relationships among agents, due to the continuous interaction. The main advantage that this method presents is that mutual understanding among agents is not statically related to a global ontology, but it can dynamically improve by following agent interactions and monitoring agent communications. An alternative approach exploiting semantic negotiation is presented in [3], where the problems brought by the schema heterogeneity in Digital Libraries are discussed. The proposed architecture integrates the ontology, agent and P2P technologies together to support the schema mapping. The goal is to allow agents embedded in different libraries to communicate semantically. Another proposal is that contained in [10]. In this work, agents in an open agent

system jointly agree on an axiomatic semantics for the agent communications language utterances they will use to communicate. This work assumes that all the agents involved start with a common semantic space, and then together assign particular locutions to specific points in this space. Such a structure would not appear to permit an incremental construction of the semantic space itself.

3 Preliminaries

3.1 JADE (Java Agent DEvelopment Framework)

JADE is a software framework fully implemented in Java language to realize distributed multiagent systems complied with the FIPA [5] specifications. JADE offers a number of advantages such as: *(i)* each agent "lives" in a runtime environment on a given host; *(ii)* communications are held by means of ACL messages; *(iii)* information can be represented as an instance of an application-specific class (a Java object). Moreover, the support for content languages and ontologies provided by JADE is designed to perform automatically all the above conversion operations, thus allowing developers manipulating information within their agents as Java objects. In order for JADE to perform the proper semantic checks on a given content expression it is necessary to classify all possible elements in the domain of discourse (i.e. elements that can appear within the content of an ACL message) according to their generic semantic characteristics. This classification is derived from the ACL language defined in FIPA which requires that the content of each ACLMessage must have a proper semantics according to the performative of the ACLMessage. The JADE *content reference model* considers only four types of elements which can be used as meaningful content of an ACL message, namely:

Predicates, are boolean expressions saying something about the status of the world, e.g. $(studies - in\ (Student : name\ Jim)\ (University : name\ MIT))$ states that "the student Jim studies in the University MIT". Generally, there are some expressions referenced inside predicates, called **Concepts**, that indicate entities with a complex structure e.g. $(Student : name\ Jim : age\ 21)$.
Agent Actions, indicating actions that can be performed by some agents, e.g. $(sell\ (Book : title\ AnnaKarenina)\ (Person : name\ Jim))$ states that the person Jim sells the book "Anna Karenina".
Identifying Relational Expressions (IRE), are expressions that identify the entities for which a given predicate is true, e.g. $(all\ ?x\ (studies - in\ ?x\ (University : nameMIT)$ identifies all the students for which the predicate $(studies - in\ (Student : name\ x)(University : name\ MIT))$ is true.
ContentElement Lists, are lists of elements of the above three types.

3.2 Extended Ontology

In JADE, an ontology is a set of schemas defining the structure of concepts, predicates and agent actions that are pertinent to the domain of interest. In the

following, we refer to the JADE ontology to define our *extended ontology*, that is formed by a set of *elements*, where each element can be either a concept, a predicate, an agent action (as in the JADE ontology) or an *explained element*, where an explained element is a set of *explanations*. In particular, the new notion of *explanation* that we introduce can be considered as a description of an element based on other elements. In other words, we introduce a different way of representing the reality of an agent with respect to a classical JADE ontology. In fact, a JADE ontology contains only unexplained elements (classes, object schemas), where each element has a single meaning expressed both by its class name (the lexical component) and its class structure (the structural component) and thus does not need any further explanation. Differently, an extended ontology contains also explained elements, that can have several meanings. Moreover, we consider that in the ontology of an agent i, each explanation is associated with a set of agents, that we call *context*, for which this explanation is understandable. Furthermore, we associate with each explanation the *explainer* agent that has provided it to i and the *confidence* that i assigns to the explanation.

Definition 1 (Explanation). Let MAS be a multiagent system and $\mathcal{O}$ an extended ontology. An *explanation* e is a tuple $\langle E, C, ea, c\rangle$ where $E \subseteq \mathcal{O}$ is the set of extended ontology elements constituting e, $C \subseteq MAS$ is the context, $ea \in MAS$ is the explainer agent that provided e, and $c \in [0, 1]$ is the explanation confidence.

Definition 2 (Extended Ontology Element). An *extended ontology element* can be a concept, a predicate, an agent action or an explained element.

Definition 3 (Explained Element). An *explained element* is a set of explanations.

Example 1. Consider the portion of an agent ontology represented in Table 1. The elements *Book* and *Quantity* are JADE concepts having respectively (*title, author, price*) and (*unit, value*) as class members. The element $Volume$ is an explained element composed of three explanations, meaning that it has three possible meanings: *(i)* $\{Book, Seller(Bookshop)\}$, with a confidence 0.9, provided by the agent b and understandable by agents a and b, *(ii)* $\{Book, Editor\}$, with a confidence of 0.8, provided by the agent a and understandable by the agents a and d, and *(iii)* $\{Quantity, Amount\}$, with a confidence of 0.5, provided by the agent e and understandable by the agents d and e. □

4 Reputation-Based Partitioning

Communications between agents in JADE are held by means of messages having a format specified by the ACL language, defined by the FIPA international standard [5]. This format includes a certain number of fields such as: *(i)* the name of the **sender** agent; *(ii)* the **performative** indicating the aim of the message (e.g. `INFORM`, `REQUEST`); *(iii)* the **content** that is the actual information included in the message and is composed of a list of content elements; *(iv)* the **language** and the **ontology** used to express the content.

Table 1. Excerpt of an agent extended ontology

Element type	Element name	Class members
Concept	Book	(title, author, price)
Concept	Quantity	(unit, value)
Explained element	Volume	({Book, Seller(Bookshop)}, {a, b}, b, 0.9) ({Book, Editor}, {a, d}, a, 0.8) ({Quantity, Amount}, {d, e}, e, 0.5)

4.1 Semantic Message

In our framework, agents performing semantic negotiation activities need to *(i)* express the content of the message using the extended ontology, and *(ii)* exchange more information (e.g. the list of the unknown elements, the timeout determined for the understanding process).

In order to satisfy the issue *(i)*, we introduce the concept of *semantic ordinary message* that extends the message format described below, by using only explained elements in its content. This is due to the fact that in our framework it is necessary to express the *semantics* of an element, while the traditional JADE concepts, predicates and agent actions are only structural elements. Moreover, to satisfy also the issue *(ii)*, we define the *semantic negotiation message* that further extends the semantic ordinary message.

Definition 4 (Semantic negotiation message). Let MAS be a multiagent system. A *semantic negotiation message* is a tuple $\langle i, j, ia, p, T, C\rangle$ where $i, j \in MAS$ are the sender and receiver agents respectively, $ia \in MAS$ is the agent interested in the understanding process, p is the performative, T is the understanding timeout determined by ia, and C is the content of the message.

The performative p indicates what the agent i wants to obtain from the agent j; as a consequence, for each performative there corresponds a different content. Our protocol is composed of six performatives, namely:

1. `SN_QUERY` : the agent i requires the help of the agent j to understand some unknown elements. For this purpose i specifies in the content C a list *ununderstood* of explained elements $(el_1, el_2, \ldots)$.
2. `SN_RESPONSE` : after receiving an `SN_QUERY` message from the agent j, the agent i replies giving some explanations. In this performative, the content C is a list of explained elements containing their explanations.
3. `SN_ACCEPT` : after receiving an `SN_RESPONSE` message from the agent j, the agent i indicates the understood explanations. In this performative, the content C is a list of explained elements containing the accepted explanations.
4. `SN_UNKNOWN` : the agent i is unable to give an answer to a previous `SN_QUERY` message sent by the agent j. The message's content is void.
5. `SN_ALREADY_ANSWERED` : as previously described in Sect. 1, an agent receiving an `SN_QUERY` message can start, in its turn, another semantic negotiation; as a consequence, an agent can receive the same `SN_QUERY` message from

different agents. In this scenario, after receiving an `SN_QUERY` message from the agent j, the agent i replies that it has already answered to the same request previously received. The message's content is void.

6. `SN_FEEDBACK` : the agent i is unable to understand a semantic ordinary message even after a semantic negotiation. In this scenario, i replies indicating the not understood explanations specified in the content C.

4.2 Agent Reputation

During the evolution of the community, agents have continuous interactions between them giving rise to different semantic negotiations; as a consequence, each agent can learn both new unknown explained elements and new explanations of already known explained elements. Unfortunately, in a scenario of uncertain trustworthiness of the agents, it is possible that an agent might learn something wrong. To avoid this problem, an agent can learn to identify "unreliable" agents marking each one with a reputation coefficient, belonging to the interval $[0, 1]$, where 1 means complete reliability. As previously described, an agent can receive a negative feedback in relation to a not understood explained element el. In particular suppose that, during a semantic negotiation, i understands the explanation e_{el} provided by the agent j for the explained element el. Suppose also that e_{el} is a wrong explanation for el. Probably, several times that i will use e_{el}, it will receive a negative feedback. An agent can assign to each explanation a confidence value, considering the frequency with which it is used in a profitable way. However, until the explanation is unused, the agent can arbitrarily fix a confidence value representing the reliability the agent has for this explanation.

Definition 5 (Explanation confidence). Let e be an explanation for an element learnt via semantic negotiation by the agent i from the agent j. Given s_{ije} be the number of times that i uses the explanation e in a semantic ordinary message, f_{ije} be the number of feedbacks received by i, containing the explanation e, and $\mathcal{R}_i$ be the reliability value the agent i assigns to new agents. The *explanation confidence* the agent i assigns to the explanation e is defined as

$$c_{ije} = \begin{cases} 1 - \frac{f_{ije}}{s_{ije}} \text{ , if } s_{ije} \neq 0 \\ \mathcal{R}_i \text{ , if } s_{ije} = 0 \end{cases}$$

Notice that, even though the explanation confidence is a characteristic of the explanation, it is strictly related to the agent that has explained it. As a consequence, an agent i can consider all the explanation confidences c_{ije} (if they exist) of the explanations learnt from the agent j to assign it a reputation coefficient.

Definition 6 (Reputation coefficient). Let $E_{ij} = \{e_1, e_2, \ldots, e_k\}$ be the set of all the explanations learnt by the agent i from the agent j, c_{ije} be the explanation confidence of $e \in E_{ij}$, and $\mathcal{R}_i$ be the reliability value the agent i assigns to new agents. The *reputation coefficient* of j with respect to i is defined as

$$r_{ij} = \begin{cases} \frac{1}{|E_{ij}|} \sum_{e \in E_{ij}} c_{ije} \text{ , if } E_{ij} \neq \emptyset \\ \mathcal{R}_i \text{ , if } E_{ij} = \emptyset \end{cases}$$

Each agent in a MAS can contribute to the community giving the information about the reputation of agents. As a consequence, the MAS is able to create an expertise coefficient on the base of all the reputation information collected by the single agents. However, when a new agent joins with the MAS, it does not have any reputation; therefore, the community can fix a value $\mathcal{R}$ depending on the reliability policy of the MAS.

Definition 7 (Expertise coefficient). Let s_{ij} be the number of times that i uses an explanation learnt from the agent j in a semantic ordinary message, r_{ij} be the agent reputation of j with respect to i, and $\mathcal{R}$ be the reliability value the community assigns to new agents. The *expertise coefficient* of j is defined as:

$$x_j = \begin{cases} \frac{\sum_{i \neq j} r_{ij} \cdot s_{ij}}{\sum_{i \neq j} s_{ij}} \text{ , if } \sum_{i \neq j} s_{ij} \neq 0 \\ \mathcal{R} \text{ , if } \sum_{i \neq j} s_{ij} = 0 \end{cases}$$

All the expertise coefficients are stored, by means of a yellow pages service, into a global database centralized on the server machine where the JADE management system runs.

Both the agent reputation and expertise coefficients are used by each agent i of the MAS to determine a partitioning in the set of the agents belonging to the MAS. We call AS_i the set of all the agents belonging to the MAS, except the agent i, and AS_i^k, $k = 1, 2, .., p_i$, the k-th partition determined by the agent i in the agent set AS_i. The agent i decides how many partitions p_i have to be considered; moreover, the criterium for assigning each agent j, belonging to AS_i, to a partition AS_i^k, is represented by a function $p(j)$ that receives the agent j as input and yields as output the number of the partition which j has to be assigned to, on the basis both of reputation and expertise of j. More in particular, the agent i assigns a weight $w_r^i \in [0, 1]$ to the reputation coefficient, representing the importance the agent i gives to the reputation; thus, it is also easy to assign a weight to the expertise coefficient as $w_x^i = 1 - w_r^i$. Finally, i computes the *negotiation degree* of j as $n_{ij} = w_x^i \cdot x_j + w_r^i \cdot r_{ij}$, and then, the function $p(j)$ is calculated as: $p(j) = z$ if $t_{z+1} \leq n_{ij} < t_z$.

5 The HISENE2 Protocol

The HISENE2 protocol is composed of the semantic negotiation protocol using reputation-based partitioning (see Fig. 1). An agent supporting the semantic negotiation can perform three different behaviours: *(i)* requiring the help of other agents to understand unknown elements, *(ii)* answering to a request, and *(iii)* sending a feedback indicating not understood explanations.

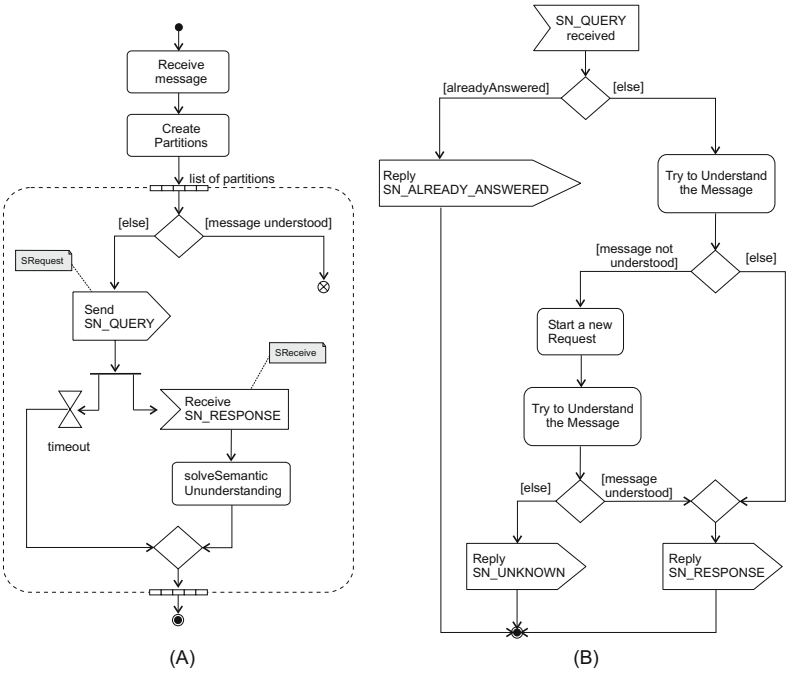


Fig. 1. The (A) Request and (B) Answer behaviours

5.1 Request Behaviour

A request behaviour is started when an agent i needs to understand unknown explained elements. This happens when i receives:

- a semantic ordinary message that i does not understand. The agent i creates the message $\langle i, j, i, SN\_QUERY, T, C\rangle$, where j is the generic receiver agent, i itself is the interested agent and C is the list *ununderstood* of unknown explained elements (see Sect. 4.1).
- an `SN_QUERY` message having in its content C some explained elements that i is unable to explain. Thus, i creates the message $\langle i, j, ia, SN\_QUERY, T, C^*\rangle$ where j is the generic receiver agent, the interested agent is the same of the received message, and $C^* \subseteq C$ is the list *ununderstood*.

A function `createPartitions` reads the expertise and reputation coefficients and, on the basis of the partition weights set by the agent owner, determines the agent partitions. Then, `SRequest` and `SReceive` behaviours are executed. `SRequest` is a OneShotBehaviour that, for each partition level k, sends an `SN_QUERY` message to each agent contained in the k-th partition, until either the list *ununderstood* becomes empty or the message timeout is reached. `SReceive` is a CyclicBehaviour in which i waits for answers from the contacted agents.

Each received `SN_RESPONSE` message has as content a list of explained elements $(el_1, el_2, \ldots, el_h)$ where el_m contains the explanations $\{e_m^1, e_m^2, \ldots, e_m^l\}$.

Therefore, the function `solveSemanticUnunderstanding`(e_m^g), $g = 1, 2, \ldots, l$ is called for each received explanation e_m^g. This function performs a schema matching between the i's ontology and the set of elements contained in e_m^g, and can be implemented by using aone of the several schema matching methods existing in literature, such as [9]. For each accepted explanation e relative to an explanation element el, the agent i stores e inside the element el in its ontology and replies an `SN_ACCEPT` message indicating the understood explanation.

5.2 Answer Behaviour

An answer behaviour starts when an agent i receives a request (`SN_QUERY`). There are three possibilities:

- i has previously answered to the same message. In this case, i replies with an `SN_ALREADY_ANSWERED` message.
- i understands the message. In this case i replies with an `SN_RESPONSE` message containing the list of explained elements as described above.
- i does not understand the whole message. In this case, i starts in its turn a request behaviour (i.e. a new semantic negotiation). After that, if the message is partially or completely understood (resp. not understood), i replies with an `SN_RESPONSE` (resp. `SN_UNKNOWN`) message.

5.3 Feedback Behaviour

A feedback behaviour starts when an agent k receives a semantic ordinary message containing some unknown explained elements from an agent i. The agent k begins a semantic negotiation in order to understand all the unknown explanations. After having concluded this negotiation, k sends an `SN_FEEDBACK` message to i, containing the list of all the explanations that remained already unknown. For each explanation e learnt from the agent j that is contained in an `SN_FEEDBACK` message, the agent i updates the explanation confidence as follows:

$$\hat{c}_{ije} = 1 - \frac{(f_{ije} + 1)}{s_{ije}} = c_{ije} - \frac{1}{s_{ije}}$$

The decrease of the explanation confidence is reflected on both reputation and expertise coefficients of j, in the following way:

$$\hat{r}_{ij} = r_{ij} - \frac{1}{|E_{ij}| \cdot s_{ije}} \qquad \hat{x}_j = x_j - \frac{\sum_{i \neq j} \frac{s_{ij}}{|E_{ij}| \cdot s_{ije}}}{\sum_{i \neq j} s_{ij}}$$

5.4 Exploitation of the Context

Thanks to semantic negotiation, an agent can learn both new unknown explained elements and new explanations of already known explained elements, but this is not the only benefit. As previously described in Sect. 3.2, each explanation is a different meaning of the relative element and is also associated with a context (i.e.

a set of agents for which this explanation is understandable). As a consequence, an agent can learn both to understand incoming messages and how to make its own messages understandable.

Example 2. Consider the agent i whose ontology is represented in Table 1. Suppose that i wants to make an offer to the agent d for the $Volume$: (Book :title "Les Fleurs du Mal" :author "Beaudelaire") and can use two different meanings for $Volume$, namely $e_1 = (\{Book, Editor\}, \{a, d\}, a, 0.8)$ and $e_2 = (\{Book, Seller(Bookshop)\}, \{a, b\}, b, 0.9)$. The agent i chooses to use e_1 in its message, since the receiver agent d is in the context of e_1. □

6 Example

In this section, we present an application of semantic negotiation in order to compare the HISENE2 protocol with the previous version of HISENE. Consider a small e-commerce agent community composed of a client agent, denoted by $a1$, operating on behalf of a human owner, and three server agents, denoted by $a2$, $a3$ and $a4$, these latter providing e-commerce web services.

Now we describe the two different evolutions of the agent community in presence of HISENE and HISENE2 separately.

6.1 HISENE

Initially, both the understanding and expertise coefficients are set to 0. Suppose that the agent $a2$ assigns the values $w_u = 0.6$ and $w_e = 0.4$.

In Fig. 2.A, we see that the agent $a1$ sends a `PROPOSE` message to $a2$, containing a predicate that says it desires to buy by auction a book having the title "Anna Karenina", under the condition that the item is located in Italy and that it can pay by a money order. The agent $a2$ receives the message, but it is unable to understand the elements *country* and *payment*, since they are not present in his ontology. Then, it decides to exploit the semantic negotiation protocol and, since both the understanding and expertise coefficients are equal to 0, then all the negotiation degrees are 0. As a consequence, the only agent partition that $a2$ can build is $AS^0_{a2} = \{a1, a3, a4\}$. Suppose that $a3$ proposes the element *music* as a synonym of *country* and the element *sum* as a synonym of payment, and that $a4$ provides the element *payment_method* to explain the element *payment*. Now, suppose the ontology of $a2$ contains both *music*, *sum* and *payment_method*: in this case, it is now able to understand the message of $a1$ completely. Moreover, both the understanding and expertise coefficients are increased of one unit for each provided synonym.

Notice that the element *music* is not a synonym of *country*, and that the element *sum* can be considered as a synonym of *payment*, but not in the meaning of the agent $a1$. As a consequence, they are wrong explanations.

In Fig. 2.B, the agent $a2$ desires to sell a camera to the agent $a4$ at a price of 150, under the condition that it wants to receive the payment by cash. It bears in mind that *sum* is a synonym of *payment* and decides to use it in this

PROPOSE message. Now, suppose the ontology of $a4$ contains all the terms used in the message; nevertheless, it does not understand the use of the element *sum* in the message. As a consequence, the agent $a4$ replies negatively. In this case, both the understanding and expertise coefficients will remain unchanged.

In Fig. 2.C is depicted the following situation, in which the agent $a1$ sends a PROPOSE message to $a2$, saying that he desires to buy a book with the title "Les Fleurs du Mal", with author "Beaudelaire" and edition 1914. Since $a2$ does not understand the element *edition*, he decides to exploit the semantic negotiation protocol first by computing the negotiation degrees. As a result, while the agent $a4$ has the value $n_{a2,a4} = 1$, the agent $a3$ has the highest negotiation degree $n_{a2,a3} = 2$. This is due to the fact that the agent $a3$ has explained more elements than other agents, independently from the quality of the descriptions.

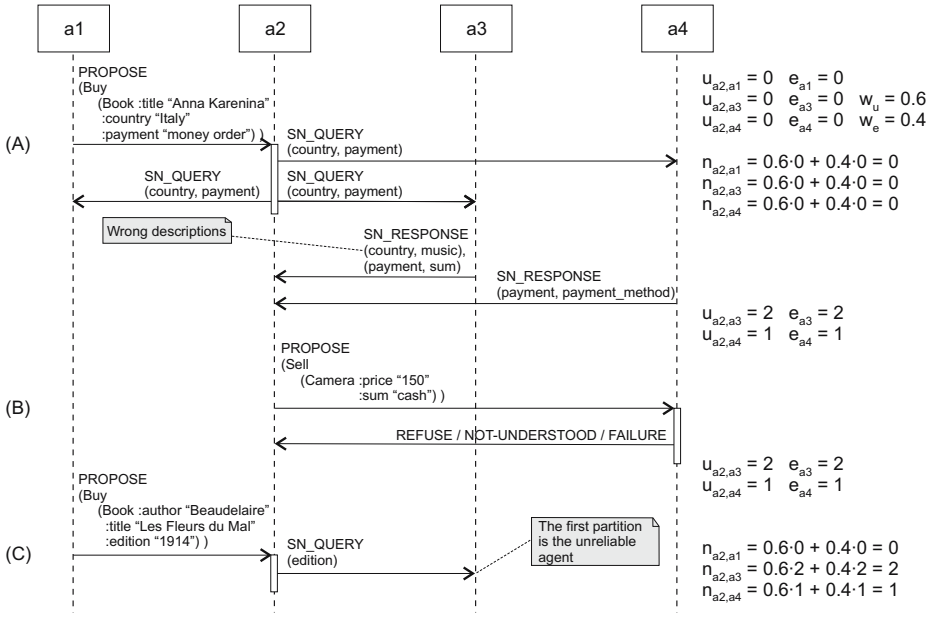


Fig. 2. Example of a community evolution supported by HISENE

6.2 HISENE2

Consider that the agent $a2$ assigns the values $w_r = 0.6$, $w_x = 0.4$, and $\mathcal{R}_{a2} = 0.7$, and that the community gives the reliability value $\mathcal{R} = 0.2$ to new agents.

In Fig. 3.A, the agent $a2$ computes the negotiation degrees, by using the coefficients $\mathcal{R}$ and $\mathcal{R}_{a2}$. Initially, none of the agents ever explained any element; as a consequence, the only agent partition that $a2$ can build is $AS^0_{a2} = \{a1, a3, a4\}$. The agents $a3$ and $a4$ provide their explanations for *country* and *payment*. The agent $a2$ accepts these explanations and assigns them the value $\mathcal{R}_{a2} = 0.7$.

In Fig. 3.B, the agent $a2$ uses the explained element *sum* for the first time as an explanation of *payment*. As a consequence, the explanation confidence of *sum* become $c_{a2,a3,\text{“}sum\text{”}} = 1 - \frac{0}{1} = 1$. Suppose that the receiver agent does not understand the meaning of *sum* in the message, and thus it performs a semantic negotiation; suppose also that the semantic negotiation is not able to make *sum* understandable. As a consequence, the agent $a4$ replies with an `SN_FEEDBACK` message. In this case, the first time *sum* has been used it received a negative feedback, thus the confidence of *sum* becomes $c_{a2,a3,\text{“}sum\text{”}} = 1 - \frac{1}{1} = 0$. Note that the fact that the confidence of *sum* has been decreased is not caused only by the unability of $a4$ to explain it; in fact, $a4$ has performed a semantic negotiation that did not give any result, therefore we can affirm that the community in its whole was unable to explain the element.

In Fig. 3.C a third situation is depicted, in which we can appreciate the HISENE2 reputation-based partitioning. The agent $a2$ does not understand *edition* and decides to exploit the semantic negotiation protocol; thus it computes the negotiation degrees: $n_{a2,a4} = 0.50$ and $n_{a2,a3} = 0.35$. This is due to the fact that $a2$ realized that *sum* is a wrong explanation for the term *payment*.

In conclusion, we observe that the reputation of an agent decreases in time with the number of wrong explanations.

7 Evaluation of the Approach

In order to assess the effectiveness of our protocol, we compared the evolution of a community composed of JADE agents having heterogeneous ontologies, both in presence and in absence of the HISENE2 protocol. We have performed our experiments on different communities of agents, having different degrees of heterogeneity. In order to compute the heterogeneity of an agent community, we note that the semantic difference between two ontology elements can be measured by using a fixed schema matching algorithm. In this experiment we have used the well-known SDR algorithm [9], that performs a schema matching both textual and structural and computes a *semantic distance* coefficient between two elements. The SDR algorithm can be applied directly if the two elements are concepts, predicates or agent actions. In the case of a pair of explained elements A and B, we compute the semantic distance between A and B as the average of the semantic distances between all the possible pairs (A_i, B_j), where $A_i \in A$ (resp. $B_j \in B$) is an explanation for the element A (resp. B). Instead, the semantic distance between two explanations A_i and B_j is computed as the average of all the pairs (A_i^h, B_j^k), where A_i^h (resp. B_j^k) is an element belonging to the extended ontology element set of A_i (resp. B_j). We call Semantic Heterogeneity Degree $SHD_{\mathcal{O}^1,\mathcal{O}^2}$ between two ontologies $\mathcal{O}^1$ and $\mathcal{O}^2$ the average of all the semantic distances between each pair $(\mathcal{O}_i^1, \mathcal{O}_j^2)$, where $\mathcal{O}_i^1$ (resp. $\mathcal{O}_j^2$) is an element of $\mathcal{O}^1$ (resp. $\mathcal{O}^2$). Finally, we call Semantic Heterogeneity Degree SHD of the whole community of agents the average of all the $SHD_{\mathcal{O}^1,\mathcal{O}^2}$. Moreover,

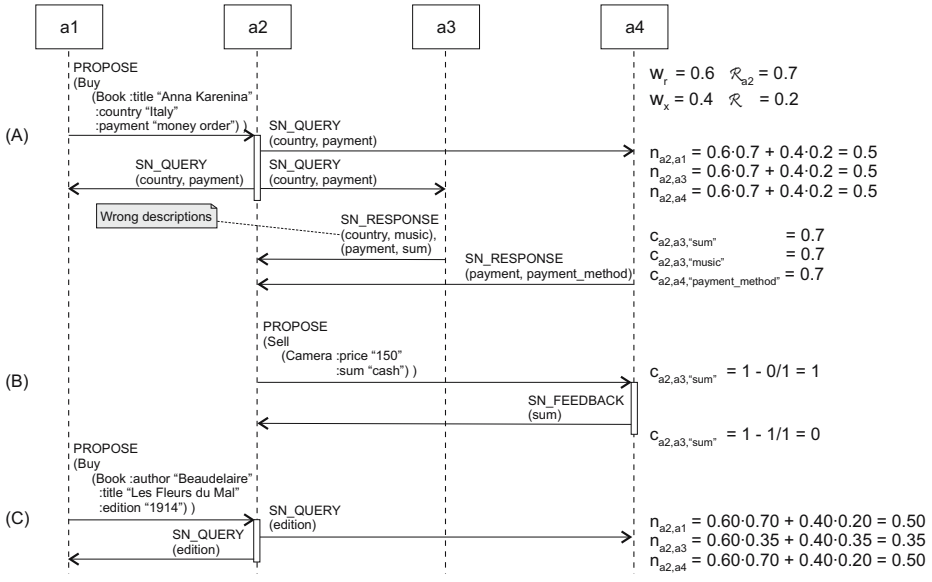


Fig. 3. Example of a community evolution supported by HISENE2

we call Understanding Rate UR the fraction of semantic ordinary messages that do not give rise to `SN_FEEDBACK` messages.

In order to obtain an estimation of our approach, we built six collections of ontologies having SHD equal to $\{0.3, 0.4, 0.5, 0.6, 0.7, 0.8\}$. The understanding rate obtained in presence and in absence of the HISENE protocol, for the different SHD, is shown in Fig. 4.a. We can observe that high values of SHD imply low UR and vice versa; this is due to the high probability of exchanging messages between agents with different ontologies. The community takes advantage on the use of HISENE that improves the UR; in particular, the higher SHD is, the higher the improvement of UR is. In fact, high level of SHD means high probability for a message to give rise to a semantic negotiation; the higher the number of semantic negotiations is, the higher the number of explanations learnt is, and the smaller the number of `SN_FEEDBACK` messages will be. In other words, semantic negotiations modify the original ontologies decreasing the SHD in time. This is well shown in Fig. 4.b, where we considered the set of ontologies having $SHD = 0.7$; this value does not change without the presence of HISENE. Otherwise, agents performing semantic negotiations learn new explained elements and explanations from other agents, adding them to their ontologies; thus, the SHD decreases in time. In Fig. 4.c we plotted the number of semantic negotiations generated by the HISENE2 protocol with respect to the previous version HISENE. We note that HISENE2 needs a smaller number of negotiations than HISENE, thus showing a better efficiency.

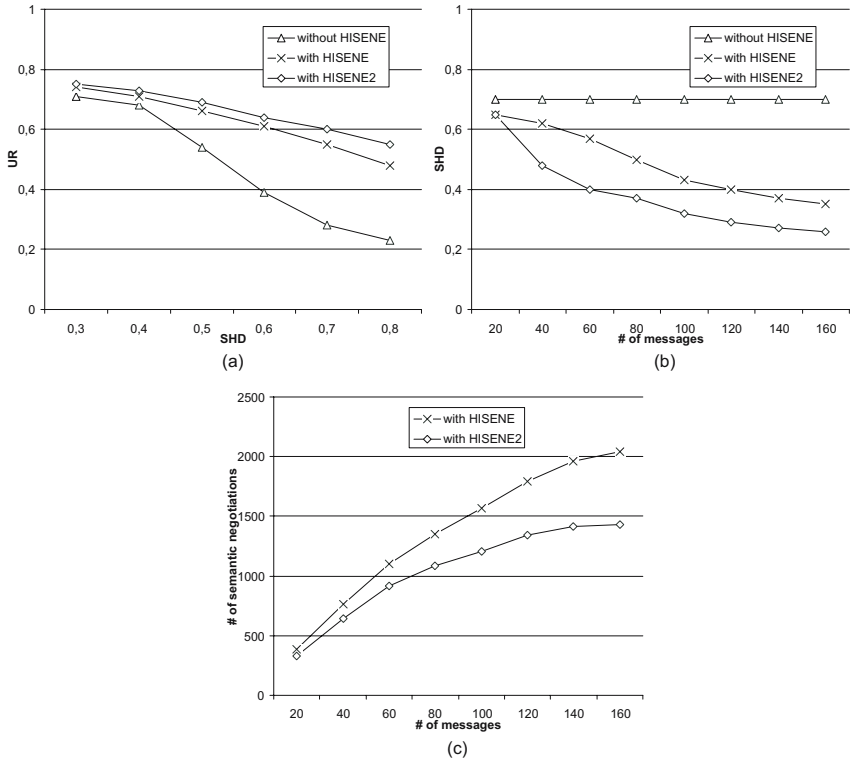


Fig. 4. Experimental results

8 Conclusions

A key issue in developing multiagent systems to support users in their Web activities is that of solving understanding problems between agents having personal ontologies that are not completely homogeneous. Semantic negotiation is a promising framework for facing this issue, and HISENE is a new semantic negotiation protocol recently introduced, that makes the process of selecting the most suitable negotiation partners effective. However, an important limitation of HISENE is that the reputation of an agent is built only on the basis of the capability to explain unknown terms to other agents, without considering the possibility that the *explanation* of a term cannot be effective. Moreover, HISENE does not admit the possibility that a term can have different meanings in different contexts. In this paper, we propose an extension of the protocol HISENE, called HISENE2, that exploits a new mechanism to determine the reputation of an agent. First of all, the ontology elements of HISENE can have different meanings, and can be used to form explanations of other elements. Moreover, the effectiveness of an explanation provided by an agent is evaluated when the explanation itself is used in agent discussions. The reputation of an agent is therefore determined on the basis of the effectiveness of the explanations that it

provided in the past to the other agents of the community. We performed some experiments to evaluate the advantages introduced by HISENE2 with respect to HISENE, and we show that the new protocol improves both the quality and the efficiency of the semantic negotiation process.

References

1. S.C. Bailin and W. Truszkowski. Ontology negotiation between intelligent information agents. *Knowl. Eng. Rev.*, 17(1):7–19, 2002.
2. R.-J. Beun, R.M. van Eijk, and H. Prust. Ontological Feedback in Multiagent Systems. In *AAMAS '04: Proc. of the Third Int. Conf. on Autonomous Agents and Multiagent Systems*, pages 110–117. IEEE, 2004.
3. H. Ding and I. Sølvberg. Towards the schema heterogeneity in distributed digital libraries. In *ICEIS (5)*, pages 307–312, 2004.
4. D.W. Embley. Toward Semantic Understanding: An Approach Based on Information Extraction Ontologies. In *CRPIT '04: Proc. of the Conf. on Australasian Database*, pages 3–12. Australian Computer Society, 2004.
5. http://www.fipa.org, 2005.
6. S. Garruzzo and D. Rosaci. Information agents that learn to understand each other via semantic negotiation. In *6th Int. Conf. on Distributed Applications and Interoperable Systems (DAIS 2006)*, pages 199–212. Springer, 2006.
7. R.V. Guha. Semantic Negotiation: Co-identifying objects across data sources. In *Proc. of the Semantic Web Services Conf.*, March 2004.
8. http://jade.tilab.com, 2005.
9. L. Palopoli, D. Rosaci, G. Terracina, and D. Ursino. A graph-based approach for extracting terminological properties from information sources with heterogeneous formats. *KAIS*, 8(4), 2005.
10. C. Reed, T.J. Norman, and N.R. Jennings. Negotiating the Semantics of Agent Communication Languages. *Computational Intelligence*, 18(2):229–25, 2002.
11. L. Soh and C. Chen. Balancing ontological and operational factors in refining multiagent neighborhoods. In *AAMAS '05: Proc. of the Int. Conf. on Autonomous agents and multiagent systems*, pages 745–752. ACM Press, 2005.
12. J. van Diggelen, R.-J. Beun, F. Dignum, R.M. van Eijk, and J.-J.Ch. Meyer. Optimal communication vocabularies and heterogeneous ontologies. In *Developments in Agent Communication*, LNAI 3396. Springer, 2004.
13. J. van Diggelen, R.-J. Beun, F. Dignum, R.M. van Eijk, and J.-J.Ch. Meyer. An effective minimal ontology negotiation environment. In *AAMAS '06: Proc. of the 5th Int. Conf. on Autonomous Agents and Multiagent Systems*. ACM Press, 2006.
14. A.B. Williams. Learning to Share Meaning in a Multi-Agent System. *Autonomous Agents and Multi-Agent Systems*, 8(2):165–193, 2004.

An HL7-Aware Multi-agent System for Efficiently Handling Query Answering in an e-Health Context

Pasquale De Meo, Gabriele Di Quarto, Giovanni Quattrone, and Domenico Ursino

DIMET, Università Mediterranea di Reggio Calabria,
Via Graziella, Località Feo di Vito, 89060 Reggio Calabria, Italy
demeo@unirc.it, dqlele@iol.it, quattrone@unirc.it, ursino@unirc.it

Abstract. In this paper we present a multi-agent system aiming at supporting patients to search health care services of their interest in an e-health scenario. The proposed system is HL7-aware in that it represents both patient and service information according to the directives of HL7, the information management standard adopted in medical context. In this paper we illustrate the technical characteristics of our system and we present a comparison between it and other related systems already proposed in the literature.

1 Introduction

Most industrialized countries are shifting toward a knowledge-based economy in which knowledge and technology play a key role for supporting both productivity and economic growth. This transition is characterized by deep changes affecting the individual quality of life and requires that economic development keeps pace with social progress. In this scenario, it is possible to foresee a rising demand of ad-hoc social services shaped around citizen needs [6].

The application of Information and Communication Technologies on the whole range of health sector activities (also known as *e-health*) can simplify the access to health care services and can boost both their quality and their effectiveness. E-Health tools allow the construction of *patient-centric* Health Care Service Providers (hereafter, $HCSP$s), aiming at supporting patients to access health-related information, to prevent their possible diseases and to monitor their health status.

These considerations justify the large amount of health-related information disseminated over the Web; as an example, European Commission has recently activated the EU-health portal that supplies information on 47 health topics and allows the access to more than 40000 trustworthy data sources [7].

At the same time, individuals are showing a growing interest to health-related Web sites; as an example, a European Commission study observed that, in 2005, at least 33% of European adult population browsed the Web to search health-related information [7].

R. Meersman, Z. Tari et al. (Eds.): OTM 2006, LNCS 4275, pp. 967–974, 2006.

Despite the abundance of available proposals, the retrieve of interesting services is not always easy. In fact, existing *HCSP*s often use proprietary formats for representing data and services; as a consequence, their interoperability and comparison are generally difficult. In addition, the vocabulary exploited by a patient for composing his queries is often limited and consists of quite generic terms; on the contrary, medical resources and services are often described by means of specialistic terms. As a consequence, terms composing patient queries usually fail to match with documents describing medical resources and services. This might cause two, generally co-occurring, consequences. The former is that a query answer might contain a large number of useless proposals that fail to fulfill patient needs (false positives). The latter is that a query answer could ignore a large number of proposals that might be useful to the patient (false negatives).

This paper focuses on these research issues and aims at providing a contribution in this setting; in fact, it proposes a multi-agent system for effectively supporting patients to detect health care services of their interest; the proposed system aims at coping with *HCSP*s interoperability and comparison problems, as well as at returning precise and complete results.

Our system is HL7 aware; in fact, it uses the HL7 (Health Level Seven) standard [1] for effectively handling both service and patient information. HL7 provides several functionalities for the exchange, the management and the integration of data regarding both patients and health care services. It is strictly related with XML, since HL7-based documents can be easily coded in this language. Moreover, it is a widely accepted standard in the marketplace [8]; specifically, *(i)* a large number of commercial products implement and support it; *(ii)* several research projects adopt it as the reference format for representing clinical documents; *(iii)* various industrial and academical efforts have been performed for harmonizing it with other standards for the electronic representation of health-related data. HL7 plays a key role in our system since it allows interoperability and comparison problems to be successfully faced.

Our system consists of five main components, namely: *(i)* a *Patient Agent - PA*, allowing a patient to submit queries for detecting services of his interest; *(ii)* a *Health Care Service Provider Agent - SPA*, supporting a *HCSP* manager to maintain the corresponding service database up-to-date; *(iii)* a *Coordinator Agent - CA*, cooperating with *PA*s and *SPA*s to detect those services appearing to be the closest to patients' queries and profiles; *(iv)* a *Health Care Service Provider Database - SD*, that is associated with a *HCSP* and manage information about services delivered by it; *(v)* a *Patient Profile Database - PD*, storing and handling patient profiles.

Each time a patient submits a query, our system "intelligently" forwards it only to the most promising *HCSP*s, i.e., to those *HCSP*s providing services that are likely to best match both the submitted query and the patient profile. In order to guarantee this important feature, it implements three ad-hoc strategies, tailored to the peculiarities of our reference context. As it will be clear in the following, these strategies avoid a patient to manually contact and query a large number of *HCSP*s for retrieving services of his interest (this last activity is

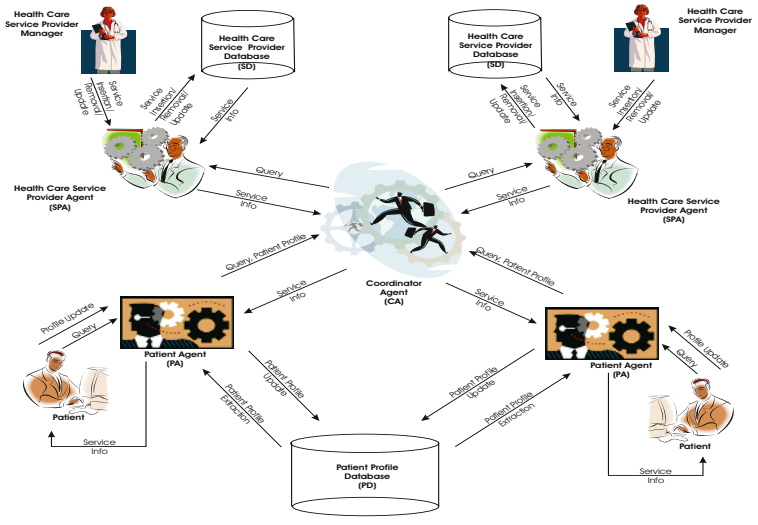


Fig. 1. General architecture of our system

usually called Brute Force search in the literature) and allows those $HCSP$s that will more likely provide useful results to be preventively identified. As a consequence, patient queries are evaluated against a small number of $HCSP$s. This allows more precise and sound results to be achieved, as well as query execution time to be reduced, $HCSP$ resource management to be improved and, finally, network performance to be increased.

2 Description of the Proposed System

The general architecture of our system is shown in Figure 1. From this figure it is possible to observe that our system is characterized by three typologies of agents, namely: *(i)* a *Patient Agent* (hereafter, PA), that supports a patient to search services of his interest; *(ii)* a *Health Care Service Provider Agent* (hereafter, SPA), that supports a $HCSP$ manager to maintain the corresponding database up-to-date; *(iii)* a *Coordinator Agent* (hereafter, CA), that cooperates with PAs and SPAs to detect those services appearing to be the closest to patients' exigencies and queries. In our system a $HCSP$ is provided with a suitable database (hereafter, SD), storing and managing information about services delivered by it. Our system is also provided with a *Patient Profile Database* (hereafter, PD), storing and handling patient profiles. In the following we call $SDSet$ (resp., $SPASet$) the set of SDs (resp., SPAs) associated with all involved $HCSP$s.

2.1 Health Care Service Provider Database (SD)

A $HCSP$ Database is associated with a $HCSP$ and stores information about the health care services delivered by it. The profile SP_j of a health care service S_j consists of a tuple $\langle SId_j, SName_j, SDescr_j, SURL_j, SFS_j, SCS_j \rangle$, where:

- SId_j, $SName_j$, $SDescr_j$, $SURL_j$ represent the code, the name, the description and the URL of S_j.
- $SFS_j = \{F_j^1, F_j^2, \ldots, F_j^p\}$ represents a set of features associated with S_j; each feature is a keyword describing a peculiarity of S_j (e.g., an illness faced by it). As an example, consider the service "chest radiography"; a possible set of features describing it is {"pneumonia", "heart failure", "emphysema", "lung cancer", "check", "radiography"}.
- $SCS_j = \{SC_j^1, SC_j^2, \ldots, SC_j^t\}$ is a set of constraints associated with S_j. A constraint SC_j^l consists of a pair $\langle SCName_j^l, SCValues_j^l \rangle$, where $SCName_j^l$ represents its name and $SCValues_j^l$ indicates the corresponding admissible values. A constraint could represent the requisites a patient must have for accessing a certain service; another constraint could define the activation or the expiration date of a service, and so on. For instance, the constraint ⟨"free", "Heart Problems"⟩, associated with the service "Holter Monitoring", might specify that if a patient has heart problems, then he is eligible for receiving Holter Monitoring service for free.

SFS_j and SCS_j are coded according to the rules specified for representing the component "entry" of the various sections of the Body of the Clinical Document Architecture (CDA), release 2. CDA is the HL7-based standard for representing clinical documents. These rules specify that section entries are represented with the support of specific dictionaries. CDA sections of interest for SFS_j are "Problem list", "History of past illness", "History of medication use", "History of present illness", "Family history", "Social history", "Immunization", "Past surgical history". CDA sections of interest for SCS_j are: "Problem list", "History of allergies", "Past surgical history", "Family history", "Social history", "Immunization", "History of past illness", "History of medication use", "History of present illness". All these sections refer to the LOINC [2] dictionary for their definition and to the SNOMED CT [3] dictionary for their content; however, if necessary, HL7 allows other dictionaries to be exploited. According to HL7 suggestions, for those features and/or constraints that cannot be coded by means of the already defined sections and the already existing dictionaries, we have defined new specific sections and a new specific dictionary. SD is constructed and handled by means of the XML technology.

2.2 Patient Profile Database (PD)

PD stores the profiles of the patients involved in our system. A Patient Profile PP_i, associated with a patient P_i, consists of a triplet $\langle DS_i, MP_i, PCS_i \rangle$, where:

- DS_i stores both the demographic and the social data of P_i.
- $MP_i = \{K_i^1, K_i^2, \ldots, K_i^m\}$ represents the Medical Profile of P_i; it consists of a set of keywords capable of representing both his pathologies and his possible needs in the health care domain.

  It is worth pointing out that MP_i is much more than a simple clinical document recording patient diseases. In fact, due to its mission, our system

should not restrict itself to record patient diseases but it should offer tools for providing patients and health care professionals with a profile-guided access to a large variety of health care related services, as well as for raising both the quality and the efficacy of the Public Health Care System. These goals can be achieved by managing patient information at a broader level than that guaranteed by a simple patient's pathologies list. This choice agrees with the ideas outlined in [9,13], where it is pointed out that, besides medical information, the profile of a patient in e-health systems should include his tasks and goals, his cognitive and psychological peculiarities, and so on.

- $PCS_i = \{PC_i^1, PC_i^2, \ldots, PC_i^l\}$ is a set of constraints associated with P_i. A constraint PC_i^h is a pair $\langle PCName_i^h, PCValues_i^h \rangle$, where $PCName_i^h$ represents its name and $PCValues_i^h$ indicates the corresponding admissible values. The exploitation of constraints allows a wide range of health-related episodes to be correctly managed. As an example, the constraint ⟨"allergen", "penicillin"⟩, associated with P_i, specifies that penicillin is an allergen for him. The relevance of this constraint is clear: it might prevent P_i from undergoing a clinical treatment involving penicillin.

DS_i is coded according to the rules specified for representing the component "Record Target" of the Header of the CDA, release 2. The fields of "Record Target" taken into account in DS_i are: "Id", "Name", "Addr", "Telecom", "Administrative Gender Code", "Birth Time", "Marital Status" and "Living Arrangement". According to HL7 philosophy, whenever necessary, we have defined and exploited other fields not already defined in HL7 standard (think, for example, to the field "patient yearly income"). MP_i and PCS_i representations conform to the rules specified for representing the component "entry" of the various sections of the Body of the CDA, release 2. Reference sections and dictionaries are the same as those considered for SFS_j and SCS_j. For those features and constraints that cannot be coded by means of the sections and the dictionaries already proposed for CDA and HL7 we have adopted the same sections and the same dictionary defined for SFS_j and SCS_j. Analogously to SD, also PD is constructed and managed by means of the XML technology.

2.3 Patient Agent (PA)

A Patient Agent PA_i is associated with a patient P_i. Its support data structure stores the profile of P_i. PA_i is activated by P_i when he wants to know if there exist new services of interest for him and, in the affirmative case, for accessing them. First PA_i retrieves PP_i from PD and stores it in its support data structure. Then, it allows P_i to submit a query Q_{i_k} specifying the service typology he presently desires. Q_{i_k} consists of both a set of keywords $QKS_{i_k} = \{Q_{i_k}^1, Q_{i_k}^2, \ldots, Q_{i_k}^r\}$ and a stop condition, stating when it can be considered satisfied (for instance, a stop condition might state that Q_{i_k} can be considered satisfied if at least 10 services have been retrieved). After this, PA_i forwards the pair $\langle Q_{i_k}, PP_i \rangle$ to CA which is in charge of processing it. When CA returns the corresponding results, PA_i presents them to P_i.

PA_i can be activated by P_i also when he wants to update his profile PP_i. In this case it retrieves the current PP_i from PD and shows it to P_i by means of a graphical interface. At this point P_i can modify it in a friendly and guided fashion. At the end of this activity PA_i stores the updated PP_i into PD.

2.4 Health Care Service Provider Agent (SPA)

SPA is an Interface Agent, analogous to that described in [5]. A SPA is associated with a $HCSP$ that uses it for adding, modifying or removing information about the services it provides. SPA allows our system to uniformly manage possibly highly heterogeneous services. It is also in charge of processing queries submitted by CA; in this case, it retrieves the necessary information from its SD and processes it in such a way to construct the corresponding answers.

2.5 Coordinator Agent (CA)

CA receives a pair $\langle Q_{i_k}, PP_i \rangle$ from a Patient Agent PA_i and processes Q_{i_k} taking also into account information stored in PP_i. First it determines those SPAs appearing to be the most adequate for answering Q_{i_k}; then, it requires each of them to process Q_{i_k}; after this, it merges provided results for constructing the final answer; finally, it sends this answer to PA_i that presents it to P_i. For selecting the most adequate SPAs for a query, we have implemented three ad-hoc algorithms, called *Patient Profile Based* (PPB), *Database Similarity & Patient Profile Based* (DS-PPB) and A^*-*Based* (AB).

PPB relies on the general observation that combining query processing techniques with user profile information into a single framework allows the most adequate answers to user queries to be found [10].

DS-PPB considers not only information stored in patient profiles but also semantic similarities possibly existing among SDs. In our opinion, the knowledge of these similarities allows the enhancement of query processing activities and, consequently, the increase of the number of relevant services that the system can propose to a patient.

The philosophy underlying AB is quite different from that underlying PPB and DS-PPB. In fact, these last can be considered "query-centric", since they process *a query at a time* and determine the SPAs best satisfying it. On the contrary, AB can be considered "provider-centric", since it considers a *set of queries* $QSet$ at a time and partitions it into disjoint subsets, each associated with a SPA of $SPASet$; this enhances the overall "quality" of answers.

3 Related Works

In this section we compare our system with other ones aiming at supporting a user in his access to health care services.

In [11] a system supporting a user to retrieve information of his interest in a medical context is proposed. It is possible to detect some similarities between our system and that described in [11]. Specifically, both of them: *(i)* provide a

suitable formalism for representing medical resources; *(ii)* provide a technique for determining the importance of each keyword composing a query. As for the main differences between them we observe that: *(i)* our system has been conceived to support mainly patients; on the contrary, the approach of [11] aims at supporting a wider range of users (e.g., patients, health professionals, etc.); *(ii)* the system of [11] does not take patient profiles into account; *(iii)* for performing its tasks, the system of [11] exploits a semantic network of medical terms whereas our system uses patient and service profiles.

In [4] a multi-agent system aiming at supporting physicians to retrieve medical information is described. As for the main similarities between the system of [4] and ours, we observe that both of them: *(i)* are provided with a suitable data structure for modelling patient needs; *(ii)* are multi-agent; *(iii)* consider an intermediate agent capable of gathering results coming from different medical databases; *(iv)* manage a scenario in which autonomous and heterogeneous medical databases coexist. As for the main differences between them, we observe that: *(i)* the system of [4] has been conceived for supporting mainly physicians; *(ii)* the system of [4] requires a medical ontology for processing queries; on the contrary, our system exploits both patient and service profiles.

In [12] *PERSIVAL*, a system for supporting physicians to search documents about patient care, is presented. *PERSIVAL* and our system are similar in that: *(i)* both of them use keywords for representing patient and service profiles; *(ii)* in both of them retrieval activity is driven by patient profiles. The main differences between them are the following: *(i)* the end users of *PERSIVAL* are physicians whereas the end users of our system are patients; *(ii) PERSIVAL* is provided with a summarization functionality; this is not present in our system; *(iii)* in *PERSIVAL* no notion someway analogous to our "Affinity" concept has been defined.

In [9] *MMA*, a system for delivering medical information to an heterogeneous audience of users, is proposed. Our system and *MMA* share some similarities; specifically, both of them: *(i)* associate a suitable profile with each user; *(ii)* consider not only strictly medical data but also other information, such as demographic and social data. As for the main differences between them, we observe that: *(i)* in *MMA* user profiles are based on stereotypes whereas, in our system, each profile is specific for a patient; *(ii) MMA* takes a great care to the graphical interface characteristics desired by a user; this aspect is not considered by our system; *(iii) MMA* has been conceived for handling various user typologies whereas our system is specifically devoted to support patients.

4 Conclusions

In this paper we have presented an HL7-aware multi-agent system supporting patients in their access to services delivered by *HCSP*s. The proposed system combines submitted queries with the corresponding patient profiles for identifying those services that are likely to satisfy patient needs and desires.

Various extensions and improvements might be thought for our system in the future. Among them we consider particularly challenging to investigate the

possibility to integrate our system with a Decision Support one; in this way, patient behaviour could be analyzed (for instance, by means of a Data Mining tool) for determining the key features of the most appreciated services. This information might be particularly useful for $HCSP$s managers when they must decide the new services to propose.

References

1. Health Level Seven (HL7). `http://www.hl7.org`.
2. Logical Observation Identifiers Names and Codes (LOINC). `http://www.regenstrief.org/loinc/`.
3. Systematized NOmenclature of MEDicine Clinical Terms (SNOMED CT). `http://www.snomed.org`.
4. L. Braun, F. Wiesman, J. van den Herik, and A. Hasman. Agent support in medical information retrieval. In *Proc. of the IJCAI International Workshop on Agents Applied in Health Care*, pages 16–25, Edinburgh, UK, 2005.
5. A. Cesta and D. D'Aloisi. Building interfaces as personal agents: a case study. *ACM SIGCHI Bullettin*, 28(3):108–113, 1996.
6. European Commission. e-Health - making healthcare better for European citizens: An action plan for a European e-Health Area. Technical report, Available at `http://europa.eu.int/information_society/doc/qualif/health/`, 2004.
7. European Commission. Reliable health information at the click of a mouse European Commission launches new Health Portal. Technical report, Available at `http://ec.europa.eu/health-eu/index_en.htm`, 2006.
8. M. Eichelberg, T. Aden, J. Riesmeier, A. Dogac, and G.B. Laleci. A survey and analysis of electronic healthcare record standards. *ACM Computing Surveys*, 37(4):277–315, 2005.
9. L. Francisco-Revilla and F.M. Shipman III. Adaptive medical information delivery combining user, task and situation models. In *Proc. of the ACM International Conference on Intelligent User Interfaces (IUI'00)*, pages 94–97, New Orleans, Louisiana, USA, 2000. ACM Press.
10. G. Koutrika and Y. Ioannidis. Personalized queries under a generalized preference model. In *Proc. of the IEEE International Conference on Data Engineering (ICDE 2005)*, pages 841–852, Tokyo, Japan, 2005. IEEE Computer Society Press.
11. Z. Liu and W.W. Chu. Knowledge-based query expansion to support scenario-specific retrieval of medical free text. In *Proc. of the ACM Symposium on Applied Computing (SAC '05)*, pages 1076–1083, Santa Fe, New Mexico, USA, 2005. ACM Press.
12. K.R. McKeown, N. Elhadad, and V. Hatzivassiloglou. Leveraging a common representation for personalized search and summarization in a medical digital library. In *Proc. of the ACM/IEEE Joint Conference on Digital Libraries (JCDL '03)*, pages 159–170, Houston, Texas, U.S.A., 2003. IEEE Computer Society.
13. E. Vasilyeva, M. Pechenizkiy, and S. Puuronen. Towards the framework of adaptive user interfaces for eHealth. In *Proc. of the IEEE Symposium on Computer-Based Medical Systems (CBMS'05)*, pages 139–144, Dublin, Ireland, 2005. IEEE Computer Society.

PersoNews: A Personalized News Reader Enhanced by Machine Learning and Semantic Filtering

E. Banos, I. Katakis, N. Bassiliades, G. Tsoumakas, and I. Vlahavas

Department of Informatics, Aristotle University of Thessaloniki,
54124, Thessaloniki, Greece
{vmpanos, katak, nbassili, greg, vlahavas}@csd.auth.gr

Abstract. In this paper, we present a web-based, machine-learning enhanced news reader (PersoNews). The main advantages of PersoNews are the aggregation of many different news sources, machine learning filtering offering personalization not only per user but also for every feed a user is subscribed to, and finally the ability for every user to watch a more abstracted topic of interest by employing a simple form of semantic filtering through a taxonomy of topics.

1 Introduction

The explosive growth of the WWW has brought essential changes in everyday life. Maybe the most determining contribution was the boundless, instantaneous and costless offering of information. Recently, the rate of available information became gigantic, making the discrimination of useful information out of tons of worthless data a tedious task. This phenomenon is commonly named "Information Overload" and comprises a main issue impeding the user finding the needed information in time.

Machine Learning (ML) and especially Text Classification (TC) is a promising field that has the potential to contribute to the solution of the problem. In TC, a classifier is trained to separate interesting messages on behalf of the user. Much work has been done to this direction [8], but, unfortunately, not many applications were widely used.

We have implemented a system (PersoNews) in order to fill this gap. PersoNews is a web-based machine learning enhanced RSS reader. It utilizes an incremental Naïve Bayes classifier in order to filter uninteresting news for the user. The web application PersoNews has a twofold functionality. Firstly, it operates as a typical RSS reader, and secondly, the user can choose from a thematic ontology a *topic of interest.* The distinctiveness of PersoNews is that both the above functionalities are enhanced by the Machine Learning Framework described in [5].

In the rest of the paper, we cover related work on News Classification, in section 2; section 3 describes the ML framework chosen for the application and section 4 presents the PersoNews system. Our paper concludes with discussion and plans for future work.

2 Related Work

The problem of News Filtering is to effectively separate interesting news articles for the user from a large amount of documents. A typical application of this task would

R. Meersman, Z. Tari et al. (Eds.): OTM 2006, LNCS 4275, pp. 975 – 982, 2006.

be to create a personalized on-line newspaper for each user. The role of machine learning in such problems was early recognized.

In [1], for example, a personalized on-line newspaper is created for every user, based on user feedback. The approach in that paper was to convert each article into a word/feature vector. Having the user profile also as a feature vector, all articles could be ranked according to their similarity with this vector.

In [3] a special purpose news browser for PalmOS-based PDAs is implemented. The authors use a Bayesian Classifier in order to calculate the probability that a specific article would be interesting for the user. An interesting part of this paper is the fact that the system does not take direct feedback by making the user evaluate every article. Instead, the news browser takes advantage of some other characteristics like total reading time, total number of lines, number of lines read by the user, and a constant denoting the user's average line reading time. All those metrics are utilized in order to automatically infer how interesting a particular user found a news article.

Billsus and Pazzani [2] implemented a Java Applet that uses Microsoft's Agent library to display an animated character, named News Dude, which reads news stories to the user. The system supports various feedback options like "interesting", "not interesting", "I already know this" and "tell me more about this". After an initial training phase, the user can ask the agent to compile a personalized news program.

Finally in [4] the user specifies his/her own categories of interest by entering keywords manually. These keywords are used in order to search for relevant articles in the world wide web. A classifier is used in order to filter uninteresting news. The classifier accepts feedback from the user who rates each article's relevancy.

Unfortunately, the aforementioned systems haven't been widely used, mainly because the problem was confronted as a personalization problem and not as an information overload problem, which appeared more recently. Moreover, most of these systems are constrained on specific sources of news articles. With the appearance of the RSS and OPML standards, it is far more straightforward to aggregate many different news sources. In addition, the user has his own personalized classifier per feed. That feature is crucial in dealing with information overload because an RSS feed can have many articles per day and the user will probably be interest for a small portion of those. Finally, in all the above systems, the user cannot declare general topics of interests like "databases" or "text classification".

In PersoNews, the user chooses a topic of interest from a thematic hierarchy. That functionality is of vital importance because a topic of interest can be covered more effectively by multiple news sources. Take as an example a user interested in a topic like "ComputerScience/DataBases". Interesting articles for this user could appear in many sources. For example, some articles may appear in a general Computer Science Journal, more relevant documents are going to be found in a more specialized source, like the proceedings of a database conference. At the same time, the user might be interested in commercial database management system like ORACLE. Therefore some interesting articles can be found in ORACLE's web site RSS feeds. In PersoNews all those sources are monitored under the same topic of interest, and additionally a classifier personalizes this monitoring for each user.

Contemporary popular Web RSS Readers like NewsGator (www.newsgator.com) and Google Reader (www.google.com/reader) can indeed aggregate and manage

many RSS feeds but they luck of an abstracted thematic ontology and there is no machine learning filtering which will abate information overload.

3 Machine Learning Framework

In order to strengthen our system with an adaptive filter that will utilize user feedback a proper Machine Learning Classification system has to be chosen. Our problem (News Classification) can be classified as a Text Stream Classification problem with the (probable) occurrence of concept drift. Concept drift is the potential change of the target-class' concept in a classification problem. Therefore, we have the following requirements for our classifier:

1. An evidently good classifier for text categorization tasks.
2. An incremental classifier is required in order to constantly update knowledge when user sends feedback.
3. Because we intend to have a personal classifier for every user and for every feed or topic of interest the user subscribes, we needed a classifier with the minimum computational cost.
4. In a Text Streaming application there is no prior knowledge of the features/words that might appear and the use of a global vocabulary of hundreds of thousands of words is simply inefficient. Therefore, we need a classifier that has the ability to build dynamically the feature space as more documents/articles arrive. We call this space a "dynamic feature space".

Feature selection is of vital importance for text classification. Our two additional requirements for feature selection are:

1. It should be incremental and able to execute in a dynamic feature space.
2. An evidently good feature evaluation metric for text classification.

Classifiers that fulfil the first and third requirement are the Naïve Bayes (NB) classifier and Support Vector Machines (SVMs). We had to reject the use of SVM classifier due to complexity reasons and the fact that up to our knowledge there is no related work that describes how to execute SVMs in a dynamic feature space. Therefore SVMs do not meet requirements 3 and 4. The Naïve Bayes classifier on the other hand has shown good performance in text classification tasks [6] and is widely used in similar problems because of its simplicity and flexibility. Moreover, it can be easily incremental and as we discuss in previous work of ours [5] it can be straightforwardly converted into a feature based classifier, meaning that it can execute in a dynamic feature space. For the above reasons, the framework proposed in [5] is selected for use in PersoNews.

4 System Implementation

PersoNews is a web application which provides users with the ability to monitor a large set of web sites (RSS feeds) and receive notifications about new publications regarding topics of their interest. The system consists of three modules which function

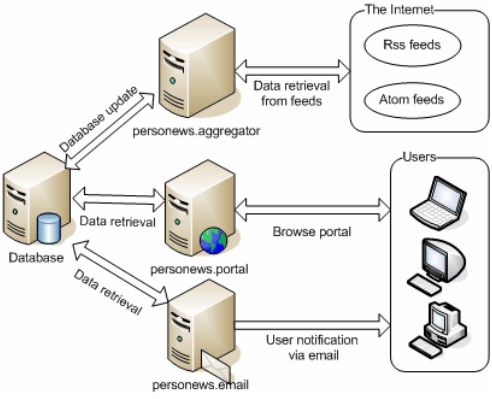


Fig. 1. PersoNews system architecture

in parallel using a common database to store information. PersoNews system architecture is shown in Fig. 1.

The main modules of PersoNews are:

Web site (PersoNews.portal). PersoNews.portal is responsible for the user interaction. Essentially, it is an open web application (news.csd.auth.gr) where everyone can register and have full access to PersoNews services using a web browser.

System update service (PersoNews.aggregator). PersoNews.aggregator is a server-side process which monitors RSS feeds in order to detect new publications and update PersoNews database. PersoNews.aggregator performs periodical polling of the news feed's URL in order to retrieve new publications which in turn are processed and stored in the database. Additionally, relevant topics are also updated.

Email notification service (PersoNews.email). PersoNews.email is responsible for notifying users by email about updates on feeds and topics they monitor. PersoNews.email is executed on a daily basis in order to check if there are any new publications in the feeds and topics monitored by each user. In that case, users are notified via email. PersoNews.email is fully customizable giving users the option to have email notifications for specific feeds and topics, change the email format or modify their email address.

4.1 Automatic News Classification

The most distinctive feature that adds up to the value of PersoNews is the integration of the Machine Learning framework discussed in the previous section We used Information Gain as a feature evaluation measure and Naive Bayes as a classifier. For the preprocessing step an implementation of Porter Stemmer [7] is used. Fig. 2 shows the preprocessing procedure for each publication.

While reading a publication from a feed or a topic, the user can perform some actions such as visit the publication's URL by clicking on its hyperlink. This action triggers a procedure that forces the PersoNews filter to update the classifier taking the

Fig. 2. PersoNews preprocessing steps

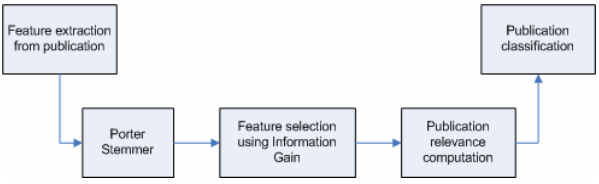


Fig. 3. PersoNews filter performs automatic classification of a new publication

specific document as an extra positive training example. Alternatively, the user can mark a publication as not interesting, forcing the PersoNews filter to utilize the specific publication as a negative training example and update the classifier. Users can also ignore new publications. In that case, the knowledge base is not updated at all.

PersoNews filter training is performed incrementally, resulting in the creation of a knowledge base which includes various publication features as well as how much do they interest each user. As a result, when PersoNews.aggregator retrieves a new publication and extracts its features, it can decide in which extend it interests the user or not based on previous user feedback. Publications classified as interesting are displayed to users and are also sent to them via email while uninteresting publications are suppressed, thus reducing information overload. Fig. 3 shows the automatic classification of a new publication. It must be noted that each user has his own classifier for each subscribed feed and topic.

4.2 Feed Manipulation and Monitoring

Users can start monitoring feeds by selecting them from the list provided by the system or by entering their own feed URL. PersoNews also supports the OPML Protocol in order to batch import any number of feeds. It must be noted that due to the large number of feeds, they are organized into abstract categories according to their topics to enable better selection and browsing for users.

Clicking on a feed title allows the user to view its publications. Fig. 4 shows a list of publications in a feed. On the top right of each publication there are four icons which correspond to the available user actions (mark as junk, forward to a friend and visit URL).

In case an article is marked by the user as not interesting, the document is forwarded to the classifier as a negative example in order to update its knowledge. When the user follows a URL, the system assumes that the user was interested in this publication and forwards the document to the classifier as an extra positive example. If the user follows the link and finds out that the article was not interesting, he/she can still mark the article as junk. In that case, the document is forwarded to the classifier as

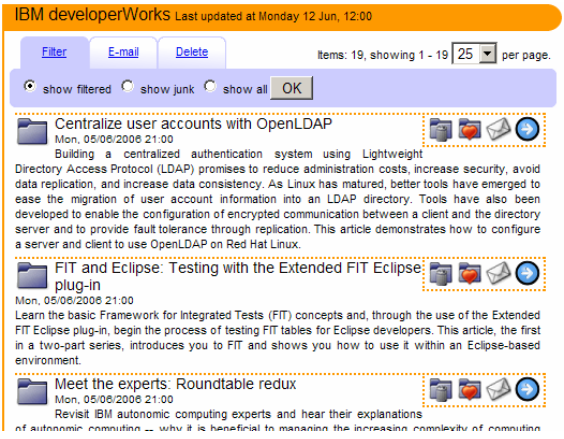


Fig. 4. Sample feed view

junk and we retrieve the previous positive example from the database. The user has also the ability to go into the "junk" folder and mark something as "not-junk" if he finds a misclassified article.

4.3 Semantic Filtering Through Topic Manipulation and Monitoring

Except from monitoring specific feeds, users are able to monitor publications regarding a special topic of interest, such as "Database Management", that belongs to the system's domain specific topic hierarchy, regardless the source feed of the publications. PersoNews has the ability to check all the feeds it is monitoring and locate new publications relevant to the topic. Currently, the topic selection list is a variant of the ACM Computing Classification System and it is organized in a tree structure featuring multiple levels of topic abstraction (Fig. 5). The topic hierarchy is implemented using an XML file to store topic descriptions as well as the associations between them. Notice that PersoNews is domain-independent, which means that it can operate under any topic hierarchy.

For each topic, the user can define one or more related keywords, which are essentially a set of words that act as an extra filter for new publications. If a publication from any feed contains any of these words then it is considered relevant to the topic. Initially, topic keywords derive from subtopics in the ACM topic hierarchy but users can also add their own custom keywords if they consider it appropriate (Fig. 6).

Under this operation, PersoNews employs a primitive form of semantic filtering, since each topic is accompanied by a number of user-defined keywords that supposedly describe the topic and can be considered as topic synonyms. Furthermore, the hierarchy of topics is also taken into account, since the keywords of all sub-topics are also considered to describe all their super-topics.

As soon as PersoNews.aggregator retrieves new publications from feeds, it performs filtering in two steps in order to detect if there are any relevant items for each topic. Initially, it scans each new publication to check if it matches any of the

Fig. 5. Form used to start monitoring a topic by selecting it from the ACM topic hierarchy

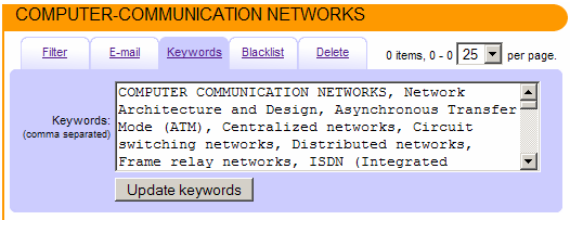


Fig. 6. Topic keywords

keywords of every user's topics and in that case the classifier of that topic examines if the publication is interesting or not for the user. It is obvious that keywords act as an additional filter which performs a selection among all new publications. The ones which are selected from the keywords filter will have to transcend the PersoNews classifier as well.

When the user starts monitoring a topic, PersoNews searches for subtopics and includes them in the keywords of the selected topic. For example, when choosing to monitor the topic "Database Systems", PersoNews automatically aggregates every subtopic keywords such as "Database Concurrency", "Distributed databases", "Multimedia databases", "Object oriented databases".

MyTopics publications are displayed like MyFeed publications but, in addition, users can visit the source feed or start monitoring the source feed of the publication, or they can add the source feed to the topic blacklist (and therefore not receive publications from these sources anymore).

5 Conclusions and Future Work

Observing the concurrent growth of the World Wide Web and Information Overload we hope that systems like PersoNews will be widely used. Unfortunately the system was up until currently in beta version and thus, we did not encourage users to register until recently. Although an exhaustive evaluation is in our immediate plans, we had a crude estimation of the system's performance taken from a small amount of registered

users. We had statistics showing that we had an average of 2.6% false negative rate (percentage of messages that the classifier marked as interesting but the users moved to the junk folder) and 5% false positive rate (percentage of messages that the classifier marked as junk but the users moved to the interesting folder), after a month of training. Although we do not claim statistical adequacy of this evaluation, we believe that these numbers are indeed encouraging.

Aside from an extensive evaluation of PersoNews, it is in our future plans to make the hierarchy offered by the system fully customizable, meaning that the user could add certain concepts in any level of the hierarchy. We also plan to investigate alternative machine learning frameworks which combine good scalability and performance. An essential important element we plan to add to our system is the aggregation of news sources that have no RSS feed available. For this purpose we investigate the potential of using Content Extraction techniques in order to extract text from simple html pages and create a corresponding RSS feed.

References

[1] Bharat, K., Kamba, T., and Albers, M., *Personalized, interactive news on the web.* Multimedia Systems, 1998. **6**(5): p. 349-358.

[2] Billsus, D. and Pazzani, M. *A Hybrid User Model for News Story Classification*. in *Seventh International Conference on User Modeling*. 1999. Banff, Canada: Springer-Verlag.

[3] Carreira, R., et al. *Evaluating adaptive user profiles for news classification*. in *9th International Conference on Intelligent user Interface*. 2004. Funchal. Madeira, Portugal: ACM Press.

[4] Chan, C.-H., Sun, A., and Lim, E.-P. *Automated Online News Classification with Personalization*. in *4th International Conference of Asian Digital Library (ICADL2001)*. 2001. Bangalore, India.

[5] Katakis, I., Tsoumakas, G., and Vlahavas, I. *On the Utility of Incremental Feature Selection for the Classification of Textual Data Streams*. in *10th Panhellenic Conference on Informatics (PCI 2005)*. 2005. Volos, Greece.: Springer-Verlag.

[6] McCallum, A. and Nigam, K., *A Comparison of Event Models for Naive Bayes Text Classification*, in *AAAI-98 Workshop on Learning for Text Categorization*. 1998.

[7] Porter, M.F., *An algorithm for suffix stripping.* Program, 1980. **14**(3): p. 130-137.

[8] Sebastiani, F., *Machine Learning in Automated Text Categorization.* ACM Computing Surveys, 2002. **34**(1): p. 1-47.

An Ontology-Based Approach for Managing and Maintaining Privacy in Information Systems

Dhiah el Diehn I. Abou-Tair and Stefan Berlik

Databases and Software Engineering Group
University of Siegen
{aboutair, berlik}@informatik.uni-siegen.de

Abstract. The use of ontologies in the fields of information retrieval and semantic web is well-known. Since long time researcher are trying to find ontological representations of the diverse laws to have a mechanism to retrieve fine granular legal information about diverse legal cases. However, one of the common problems software systems are faced with in constitutional states is the adapting of the diverse privacy directives. This is a very complex task due to lacks in current software solutions – especially from the architectural point of view. In fact, we miss software solutions that manage privacy directives in a central instance in a structured manner. Even more, such a solution should provide a fine granular access control mechanism on the data entities to ensure that every aspect of the privacy directives can be reflected. Moreover, the whole system should be transparent, comprehensible, and modifiable at runtime. This paper provides a novel solution for this by means of ontologies. The usage of ontologies in our approach differs from the conventional form in focusing on generating access control policies which are adapted from our software framework to provide fine granular access on the diverse data sources.

1 Introduction

In constitutional states, software applications in nearly every domain have to provide and to ensure security issues. In fact, security issues usually are derived from laws, e.g. data protection acts or general security rules stemming from the domain itself. In Article 8 of the 1950 European Convention of Human Rights and Fundamental Freedoms, privacy is declared as a fundamental human right. The EU Directive 95/46/EC of the European Parliament and of the Council of 24 October 1995 deal with the protection of individuals with regard to the processing and on the free movement of their personal data [1]. However, usually laws and rules are given as plain texts and lack a common formalism. During the last years ontologies have been used as a formalism to describe laws and rules on a common basis, where mainly their task is to retrieve fine granular legal information about diverse legal cases, i.e. as inference mechanism.

Actually, it is a big challenge to adapt such laws and regulation in a software system. On the one hand, it is a challenging task to include the formalized laws and rules in software applications and especially in distributed applications. In

R. Meersman, Z. Tari et al. (Eds.): OTM 2006, LNCS 4275, pp. 983–994, 2006.

general, more than one regulation affect the terms of privacy concerning one organization. In the case of German universities, the Federal Data Protection Act (Bundesdatenschutzgesetz), the Sate Data Protection Act (Landesdatenschutzgesetz) as well as university internal acts must be considered when accessing personal information [2]. This complexity may make it impossible to predict a clear defined rule type of privacy regulation and privacy guidelines. Therefore, in Germany every organization which belongs to the public bodies must have a data protection officer. One of his main tasks is to assure that used software systems conform to the diverse privacy regulations, see Figure 1. In fact, he instructs the programmer developing the systems and in the end controlls the developing process.

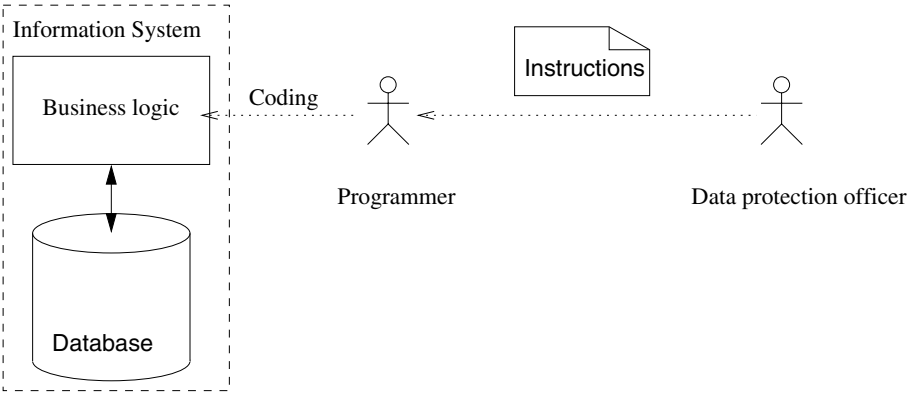


Fig. 1. Actual process to ensure privacy

On the other hand, we missed solutions that handle the problem from the architectural point of view by providing enterprise architectures and concepts that support fine granular access on the diverse data sources in software applications. In general, software developers and designers adapt the privacy regulations by coding them into the software, sometimes over access control policies and sometimes by creating functions in the application's business logic. By doing so, it is hard to trace whether a system is considering the privacy regulation as well as to modify the privacy policies. Moreover, implementing complex policies such as health care or government cannot be applied using these methods.

This paper provides a novel solution for this by means of ontologies. The usage of ontologies in our approach differs from the conventional form in focusing on generating access control policies which are adapted from our software framework to provide fine granular access on the diverse data sources.

2 Related Works

Privacy Enhancing Technologies (PETs) refer to a variety of technologies that protect personal data by minimizing or eliminating the collection of personal

data as well as the technologies to protect the confidentiality, integrity and availability of personal data [3]. The central target of PETs is to provide the technological infrastructure, implementation, and concept to protect the intimate right of individual's self determination of their personal data. Moreover, PETs provide an interface to the various privacy regulations to be adapted in the digital landscape. In other words, PETs ensure that these regulations will be considered when accessing personal data. In addition, PETs aim at making the adaption process of privacy regulation transparent so that it can be easily understood by the average user.

The researchers of the META GROUP [4] classify PETs in two main categories; privacy protection and privacy management. The first category comprises tools and technologies that are actively involved in protecting the privacy assigned. Such tools and technologies enable electronic transactions to take place without any need for private information; providing anonymous web services like browsing and email capability, i.e. without exposing the user's address and identity. Moreover, encryption, filtering, and tracking tools belong to this category. These tools protect emails, documents and transactions from being read by other parties; protect the user against unwanted email and web content by blocking them; and obliterate electronic traces of the user's activity. In contrast, the privacy management category includes tools and technologies that support the administration of privacy rules, such as tools for managing user identity and permissions as well as tools for creating and checking privacy policies [4].

Besides the concept of PETs a different effort can be identified in the work of the Transparent Accountable Datamining Initiative. The TAMI Project aims at creating technical, legal, and policy foundations for transparency and accountability in large-scale aggregation and inferencing across heterogeneous information systems [5].

In the following subsections an overview about a variety of technologies which can be used to ensure an information system's conformity with data protection regulations will be given.

2.1 Platform for Privacy Preferences

Platform for Privacy Preferences (P3P) is a specification developed by the World Wide Web Consortium (W3C) [6]. It enables Web sites to express their privacy practices in a standard XML-based format to be automatically retrieved and interpreted by user agents. In fact the implementation of P3P does not provide privacy protection, but it can greatly advance transparency and can thus be used to support efforts to improve privacy protection.

2.2 Resource Access Decision

Resource Access Decision (RAD) [7] constitutes an access right retrieval framework. Individual resources can be bound to names, and have policies and rules assigned to them. Based on the identity of the user and the applicable policies, a decision about the user's access rights is derived and returned to the business logic which subsequently handles this decision. In fact this approach can handle

all eventualities, but so far there is no commercial implementation. Moreover, one of the critical points is the administration process, which can become increasingly complicated unless a clear structure of privileges is created early on [8].

2.3 Access Control List

Access Control Lists (ACL) provide a general mechanism for assigning rights to individual users or roles. In the context of PET this is insufficient, as ACLs offer no means to fulfill certain aspects such as limitations to the duration for which private data may be stored. Furthermore, to grant a user access to his/her own private data but not to that of other users of the same class, each user would require an own role. This would render the entire role concept useless and cause an inflation of rights.

2.4 Digital/Private Rights Management

Digital Rights Management (DRM) and the derived Private Rights Management (PRM) are powerful frameworks for controlling various dimensions of content usage and distribution. DRM were originally developed for the purpose of controlling the usage of digital media such as video or music, but due to its nature the concept can be used to ensure the protection of other private data from unauthorized access and usage as well. PRM constitutes a variant of the DRM concept which is geared toward this purpose. They however require comparatively large architectural considerations in their host application, as these approaches basically are applications on their own. Moreover, there are considerable problems with scalability. The general concept was designed to control the infrequent access of a few users to lots of data from few sources, such as no more than a couple of accesses to media files of several megabytes each per minute. This is opposed to frequent access of many different users to comparatively little data from many sources [9].

2.5 Rights Management Languages

Rights Management Languages (RML) such as XrML [10] or ODRL [11] are a standardized format for storing access right information in external resources. Such standards are used e.g. in DRM. One major benefit is the separation of the individual access rights from the content to which they relate, which increases flexibility and transparency.

2.6 Web Service Policies in OWL-DL

Kolovski et al. proposed a mapping of WS-Policy to the description logic fragment species of the Web Ontology Language (OWL-DL) [12] and claim that OWL reasoners are ready to be used as policy processing tools. They describe how standard OWL-DL reasoners can be used to check policy conformance and perform several policy analysis tasks. Since OWL-DL is much more expressive

than WS-Policy it provides a framework for exploring richer policy languages. However, the focus of WS-Policy is on the aspects of a service required to establish a connection between endpoints. Thus it does not require a great expressiveness and its scope is limited with respect to other approaches as e.g. provided by the XACML model.

3 Our Approach

Figure 2 illustrates our conceptual architecture to adapt privacy directives. Generally the design consists of the application, the XACML Framework, the Privacy Manager, and the data protection officer. The application is a classical multi tier architecture. A change is made by the addition of the privacy layer, which is located between the business logic and the data model. The privacy layer's

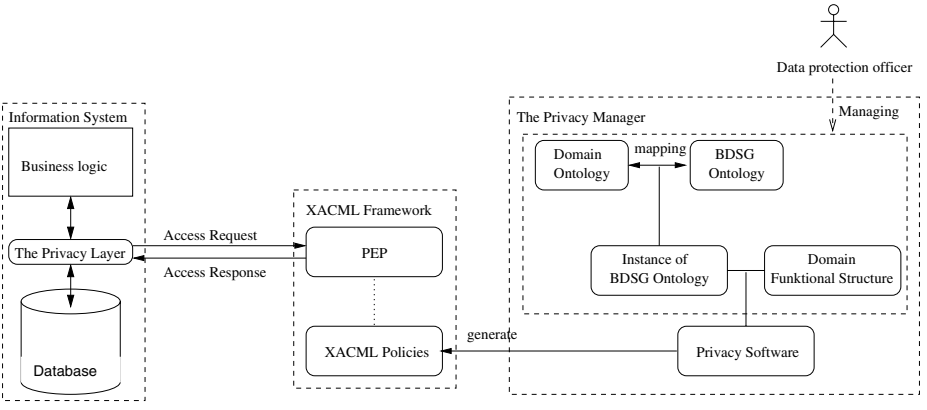


Fig. 2. A conceptual architecture to adapt privacy directives

behavior is controlled by the XACML Framework, which constitutes the interface between the Privacy Manager and the application. The Privacy Manager generates the XACML policies to be used by the XACML Framework and is managed by the data protection officer. In the following subsections we describe the components in detail.

3.1 Privacy Manager

The main task of the Privacy Manager, see Figure 2, is to generate the XACML policies to be used by the XACML Framework. It comprises two components. The first component consist of a ontological representation of the German Federal Data Protection Act (Bundesdatenschutzgesetz, BDSG) and the second one includes domain internal security restrictions and the domain ontology itself.

BDSG. It consists of an ontology that represents the German Federal Data Protection Act (Bundesdatenschutzgesetz). The decision to use an ontological

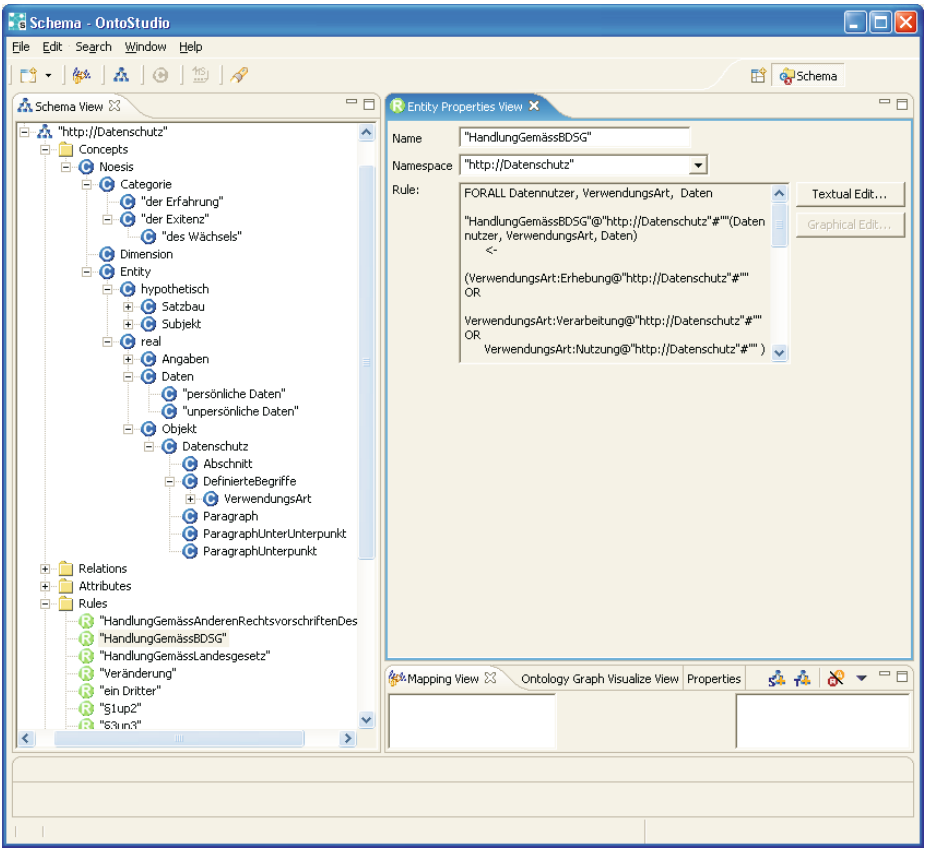


Fig. 3. BDSG Ontology in OntoStudio

technology was made because we have through this a mechanism to map law statements to a machine interpretable language without losing any aspect of its meaning. In fact, we apply the subject, predicate and object principle. As subjects we declared persons/individuals who are affected from the BDSG, the BDSG paragraphs are represented as predicates and objects as entities. By constructing the BDSG-Ontology we profit from well-known experience in the field of AI and law, like FOLaw's [13] and CAUSATIO^{NT} [14] ontologies.

We use OntoStudio[1] to model the concepts, relations, attributes, and rules of the BDSG-ontology. Figure 3 illustrates a screen shot of the BDSG-ontology in Ontostudio. The rules of the ontology which represent the diverse regulations rules are represented in F-Logic. Below is a clause from BDSG – paragraph 2 of section 1:

[1] A research license of OntoStudio which is a product of the Ontoprise GmbH – http://www.ontoprise.de

This Act shall apply to the collection, processing and use of personal data by

1. public bodies of the Federation,
2. public bodies of the Länder in so far as data protection is not governed by Land legislation and in so far as they
   (a) execute federal law or
   (b) act as bodies of the judicature and are not dealing
3. private bodies in so far as they process or use data in or from data files in the normal course of business or for professional or commercial purposes.

Fig. 4. BDSG: paragraph 2 of section 1

For a representation of this paragraph in a predicate language, here it is F-Logic, see Figure 5.

```
FORALL X(type_of_use), Y(user), Z(data)
X,Y,Z : BDSG_applies
<-
(X: collection OR X: processing OR X: use) AND
Z: personal_data  AND
(Y: public_bodies_of_the_Federation OR
(Y: public_bodies_of_the_Länder AND NOT
X,Y,Z: Länder_Act_applies AND (Y: execute_federal_law OR
(Y: act_as_bodies_of_the_judicature AND NOT Y,X :
dealing_with_administration))
OR
Y: private_bodies AND
(X: course_of_business OR X:professional_or_commercial_purposes))
```

Fig. 5. BDSG: paragraph 2 of section 1 in F-Logic

Domain Ontology and internal security restrictions. The domain ontology represents the diverse entities in the domain, one could say it is derived from the database scheme. The purpose of this is to map the diverse entities of a special domain to the abstract entities of the BDSG ontology. By doing this we instantiate a valid BDSG according to the domain of use. In most cases not only the BDSG determines the privacy regulation in a domain, but also a set of domain specific guidelines. For example, in the case of a university, students have not the right to overwrite some of their data. Therefore, we generate the domain security restrictions as a set of F-Logic terms, see Figure 5.

3.2 Data Protection Officer

As shown in [15], the normative knowledge of the law alone is of limited worth. Since legal sources contain and assume non-legal, common-sense domain knowledge, these also have to be supplied to the system. In our approach we have implemented the normative knowledge of the privacy law in a form of an ontology, see subsection 3.1. The missing meta and world knowledge is given by the data protection officer running the system. He also is responsible to incorporate special individual rights. In contrast to conventional systems, he thus always has the overview on the implemented privacy rules. Moreover, if the legal situation changes he can himself modify the affected entities. Another advantage is that the implemented rules are represented in a transparent and comprehensible form. This way, third persons have the opportunity to comprehend the system.

3.3 XACML

XACML is an initiative to provide a standard for access control and authorization systems which is generic, distributed, and powerful [16]. In contrast, most of the current systems implement access control and authorization in a proprietary manner.

XACML provides an XML based access control policy language together with an access control decision request/response language. Applications and systems can use these to fulfill their access control needs. The XACML specification just deals with the framework and leaves the exact implementation details of the access control engine for the actual implementation [17,18]. Besides the objective to create a portable and standard way of describing access control entities and their attributes, XACML aims at providing a mechanism that offers much finer granular access control than simply denying or granting access.

XACML is composed of the components depicted in Figure 6. The model operates by the following steps.

0. Policy Access Points (PAPs) write policies or policy sets and provide them to the Policy Decision Point. These policies consist of one or more rules and represent the complete policy for a specified target.
1. The access requester sends a request for access to the Policy Enforcement Point.
2. The Policy Enforcement Point (PEP) sends the request for access to the context handler in its native request format, which might optionally include attributes of the subjects, resource, action, and environment.
3. The context handler initiates an XACML request context and sends it to the Policy Decision Point.
4. The Policy Decision Point (PDP) requests any requisite subject, resource, action, and environment attributes from the context handler.
5. The context handler requests the attributes from a Policy Information Point.
6. The Policy Information Point (PIP) obtains the requested attributes.
7. The PIP returns the requested attributes to the context handler.
8. Optionally, the context handler includes the resource in the context.

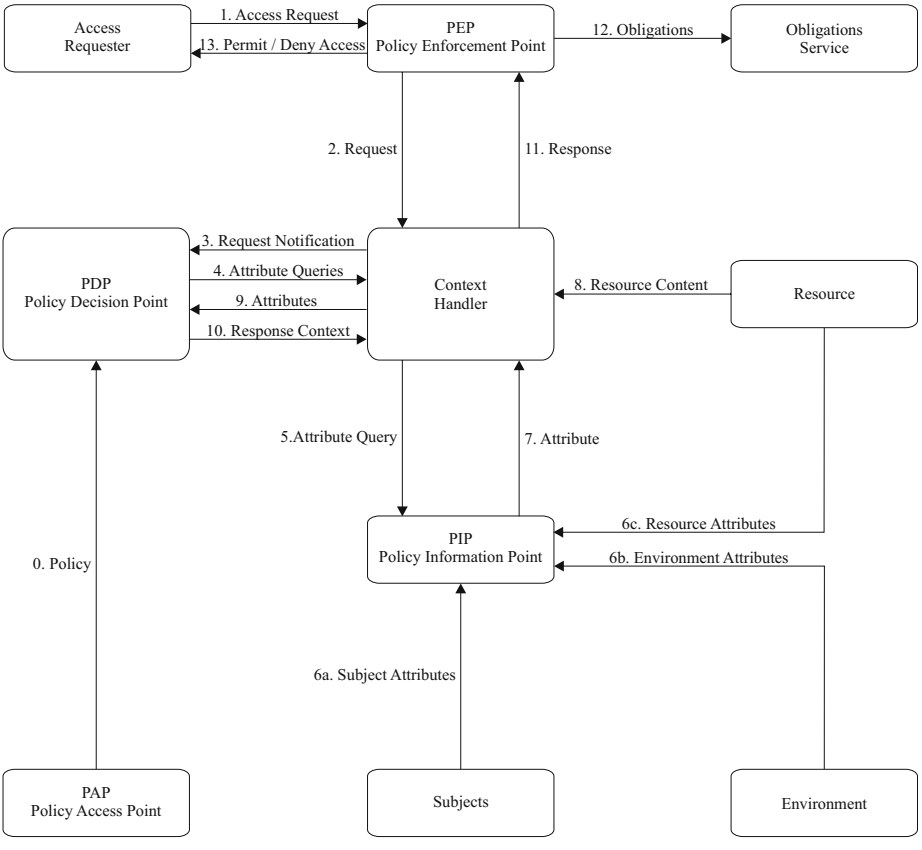


Fig. 6. XACML data-flow diagram

9. The context handler sends the requested attributes and – if given – the resource to the PDP which in turn evaluates the policy.
10. The PDP returns the response context including the authorization decision to the context handler.
11. The context handler translates the response context to the native response format of the PEP and returns it then to the PEP.
12. The PEP fulfills the obligations.
13. If access is permitted, then the PEP grants the requester access to the resource; otherwise, it denies access.

The policies provided by the Policy Access Point may in the simplest case just be coded by hand. However, even simple policies can result in some bulky XML documents hard to read and analyze directly again. As this method is cumbersome and error-prone the policies containing XML documents are usually generated from some database or software systems, see e.g. [19]. Nevertheless, also this approach does not solve the maintenance problem in an adequate

manner since the policy representing facts molder into unrelated trifles. Therefore policies in our approach originate from the Privacy Manager by means of an ontology which is used to maintain the access regulations. The ontology represents the collectivity of the policies to be derived as a central instance in a structured manner; in the example at hand privacy directives of different laws. The whole system is transparent, comprehensible, modifiable at runtime and even allows to do inference. The transformation from the ontology's native format into XACML conform policies can be done e.g. by the OntoBroker tool.

4 Implementation

The whole BDSG has been implemented in form of an ontology using OntoStudio, where the laws' clauses are modeled in F-Logic, c.f. Section 3.1. The main motivation to use OntoStudio was the fact that together with OntoBroker we have a software solution at hand that can easily be integrated with our implementation. Even if OntoBroker supports inferencing on the ontology we just use it to judge concrete access control requests. For demonstration purposes we generated a static case study. In this example we illustrated the case of a professor trying to read a student's address. The principally query 'ActionAllowed(professor, read, address, student)' is subsequently unfolded in the BDSG ontology till all clauses concerning the actor, access type, access target, and affected subject are evaluated and can be determined. The result of the query is then transformed into an XACML policy. The example (available in german only) can be called from our web site[2].

Through this example we demonstrated the proof-of-concept. By use of an ontology and XACML it is possible to provide fine granular access control on several data entities. At the moment we are working on the automatization of this process.

5 Conclusions

In this paper we have proposed an innovative framework that:

1. ensures privacy according to the diverse data privacy directives, and
2. shows how transparent and comprehensible this can be.

We presented a solution that manages privacy directives in a central instance in a structured manner. This solution provides a fine granular access control mechanism on the data entities using XACML to ensure that every aspect of the privacy directives can be reflected. Moreover, the whole system is transparent, comprehensible, and modifiable at runtime. This is achieved by means of ontologies, where the usage of ontologies in our case differs from the conventional form in focusing on generating access control policies which are adapted from our software framework to provide fine granular access on the diverse data sources.

[2] See http://pi.informatik.uni-siegen.de/whois/BDSG_Ontology_Demo/

Further we redefined the role of the data protection officer. He now always has the overview on the implemented privacy rules. Moreover, if the legal situation changes, he can himself modify the affected entities. Another advantage is that the implemented rules are represented in a transparent and comprehensible form. This way, third persons have the opportunity to comprehend the system. First results are obtained by a completely elaborated example which demonstrated the proof-of-concept.

Acknowledgement

The authors would like to thank M. Niemczyk for implementing the BDSG ontology.

References

1. European Parliament and Council, "Official journal l 281, 23/11/1995 p. 0031 - 0050."
2. M. Wettern and J. Von Knop, "Datenschutz im hochschulbereich," in *Jahrbuch der Heinrich-Heine-Universität Düsseldorf 2004*, 2005, pp. 575–589.
3. S. Fischer-Hübner, *IT-Security and Privacy - Design and Use of Privacy-Enhancing Security Mechanisms*, ser. Lecture Notes in Computer Science. Springer, 2001, vol. 1958.
4. META Group, "Privacy enhancing technologies," Danish Ministry of Science, Technology and Innovation, Tech. Rep., 2005.
5. Decentralized Information Group, "Transparent accountable datamining initiative," 2006. [Online]. Available: http://dig.csail.mit.edu/TAMI/
6. W3C, "Platform for privacy preferences (p3p) project." [Online]. Available: http://www.w3.org/P3P/
7. The Object Management Group (OMG), "Resource access decision." [Online]. Available: http://www.omg.org/technology/documents/formal/resource_access_decision.htm
8. W. Eberling, "Resource access decision - ein framework zur realisierung eines datenbasierten zugriffsschutzes," MATHEMA Software GmbH, Tech. Rep., 2003.
9. L. Korba and S. Kenny, "Towards meeting the privacy challenge: Adapting drm." in *Digital Rights Management Workshop*, 2002, pp. 118–136.
10. XrML, "Xrml - the digital rights language for trusted content and services." [Online]. Available: http://www.xrml.org/
11. ODRL, "ODRL - Open Digital Rights Language." [Online]. Available: http://odrl.net/
12. V. Kolovski, B. Parsia, Y. Katz, and J. A. Hendler, "Representing web service policies in owl-dl," in *International Semantic Web Conference*, 2005, pp. 461–475.
13. J. Breuker and R. Hoekstra, "Epistemology and ontology in core ontologies: FOLaw and LRI-Core, two core ontologies for law," in *Proceedings of EKAW Workshop on Core ontologies*. CEUR, 2004. [Online]. Available: http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/
14. J. Lehmann, J. Breuker, and B. Brouwer, "Causatio^{nt}: Modeling causation in ai&law." in *Law and the Semantic Web*, 2003, pp. 77–96.

15. J. Breuker, A. Valente, and R. Winkels, "Use and reuse of legal ontologies in knowledge engineering and information management." in *Law and the Semantic Web*, ser. Lecture Notes in Computer Science, vol. 3369. Springer, 2003, pp. 36–64.
16. OASIS, *eXtensible Access Control Markup Language (XACML)*, Feb 2005. [Online]. Available: http://www.oasis-open.org/committees
17. S. Microsystems, "Sun's XACML Implementation," 2006. [Online]. Available: http://sunxacml.sourceforge.net/
18. M. Verma, "XML Security: Control information access with XACML," 2004. [Online]. Available: http://www-128.ibm.com/developerworks/library/x-xacml/
19. L. Seitz, E. Rissanen, T. Sandholm, B. S. Firozabadi, and O. Mulmo, "Policy administration control and delegation using xacml and delegent," in *6th IEEE/ACM International Workshop on Grid Computing, Seattle, USA*. IEEE Press, 2005.

Ontology-Based User Context Management: The Challenges of Imperfection and Time-Dependence

Andreas Schmidt

FZI Research Center for Information Technologies
Information Process Engineering
Haid-und-Neu-Straße 10-14, 76131 Karlsruhe, Germany
Andreas.Schmidt@fzi.de

Abstract. Robust and scalable user context management is the key enabler for the emerging context- and situation-aware applications, and ontology-based approaches have shown their usefulness for capturing especially context information on a high level of abstraction. But so far the problem has not been approached as a data management problem, which is key to scalability and robustness. The specific challenges lie in the imperfection of high-level context information, its time-dependence and the variability in the dynamics between its different elements. The approach presented in this paper presents a layered data model which structures the problems and is geared towards flexible and efficient query processing in combination of relational database and logic-based techniques. The techniques have been successfully applied for context-aware corporate learning support.

1 Introduction

Situation (or context awareness) has become a major topic in a wide range of research areas – among them mobile information systems, ambient intelligence, adaptive e-learning and knowledge management systems. Especially in information systems research, this expresses the insight that after the quest for making available vast amounts of information and for doing that efficiently, it is now the user who is the bottleneck. In order to find relevant information, the user needs to specify more precisely what she actually needs. But in many cases, the user is either not capable of doing that, or it drastically reduces the usability of the system – or both. This usage efficiency dilemma between selectivity on the one side and ease of use on the other side can be overcome by the system's awareness of the situation of the user. The system can then add transparently implicit assumptions of the user to her explicit queries or actions.

This idea sounds compelling, but closer inspection reveals that it faces fundamental challenges. Most of them can be traced back to the problem that the system cannot sense the usage situation (as the subset of the state of the real world relevant to the interaction with the system) directly, but has to rely on the usage context as a model for that situation (see fig. 1). This model is the result of a mapping which is highly imperfect in its nature (see also [1]):

– The mapping is **incomplete** as the system will never be able to capture all of the different aspects of the situation.

R. Meersman, Z. Tari et al. (Eds.): OTM 2006, LNCS 4275, pp. 995–1011, 2006.

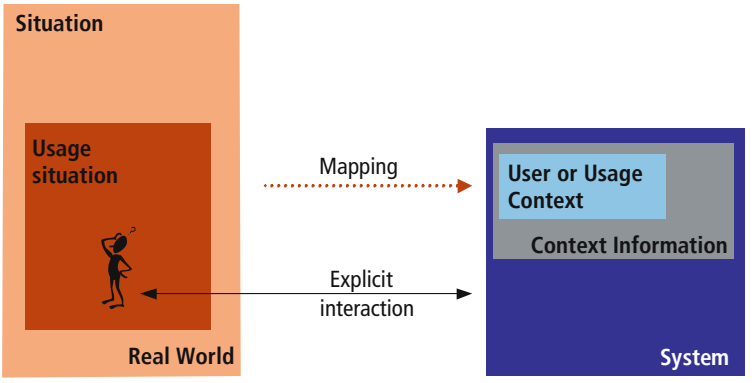


Fig. 1. Situation and Context: The origin of the problems

- The mapping is **uncertain** as the system has to rely on indirect methods and heuristics for eliciting context information from observable data.
- The mapping is **imprecise** as these methods yield only results with limited precision
- The mapping is **inconsistent** as a consequence of contradictions, resulting from different methods and their uncertainty and imprecision.

Especially for high-level context information [2], i.e. context information on a high level of abstraction as opposed to sensor-level information, this is aggravated by the problem of dynamics [3]. On the one side, it is often not possible to determine context on demand, but rather the system has to collect its pieces in advance *(asynchronicity of acquisition and usage)*. But on the other side, some parts of the context change often quite quickly, others are rather stable *(variability in the rate of change)*. Furthermore, efficient user context management requires a fairly deep *understanding of the context semantics*, especially when augmenting the collected context information, which is inevitable for high-quality context. This makes the provision of context information to applications a complex task that should be realized by a specialized user context management service – in analogy to other data management systems.

The requirement for understanding the context semantics calls for ontology-based approaches, which have proven their usefulness for incorporating a shared semantics in applications, but ontology management systems currently in use are not geared towards the pecularities of user context data, which is characterized by various forms of imperfection and a strong time-dependency both in terms of validity and reliability [4] combined with a high update frequency. In this paper, an approach is presented that is capable of representing and efficiently dealing with these challenges. The organization of the paper is as follows: in section 2, the requirements for user context data management will be analyzed and presented. In section 3 the layered data model is presented that is used to represent user context data. Section 4 will cover the issue of integrating ontologies into the data model. In section 5, some implementation issues will be discussed in the frame of a case study. The paper will close with a review of related work in section 6 and conclusions and outlook in section 7.

2 Usage Scenario

2.1 Scenario

The main guiding scenario for the user context management approach presented here was context-steered workplace learning [5]. In order to achieve the integration of work and learning processes, the learning support system does not rely on the user searching actively for appropriate learning material, but observes what the user is currently doing. Based on these observations and background knowledge on the competency requirements of elements of the user's situation like task, process, or role, the system then suggests appropriate learning programs which are compiled on demand from fine-grained learning objects. Additionally, the system can also suggest co-workers who are experts in a certain area, or who were in a similar situation recently. This approach of awareness of the work and learning situation can help to reduce the cognitive load of self-steered learning drastically. In this scenario, situation awareness has many different aspects which have been systematically analyzed in [6]) and can be divided along the different phases of the e-learning process: authoring, delivery (what, when, how), and execution.

2.2 Use Cases

In order to support the scenario sketched above on a technical level, a service-oriented architecture of learning support services has been conceived within the project *Learning in Process*. The need of these services has formed the basis of the following use cases from which the requirements for the user context management approach have been derived:

- **Retrieve feature values for the current context.** The standard use case for context-aware application is the retrieval of certain context feature values which are considered relevant for adapting the system behaviour. A learning system can adapt the presentation to the technical context (broadband access, loudspeakers available) or the selection based on the current project context or mid-term interests and goals.
- **Check for certain feature values.** In the presence of incompleteness, this use case is slightly different from the previous one as the query can have different modalities. It can take the form of: does the user have the value X for feature Y? This makes sense if we have an application that offers support only under special conditions. But it could also take the form of: Is it possible (with a certain error probability) that a user has the value X for feature Y. This is useful for strategies where some features are highly critical (e.g. emotional state, but also social relationships), but evidence is usually rather scarce.
- **Query for other users in a certain context.** Especially in the area of corporate learning where the social dimension plays an important role, it is highly desirable to establish contacts between employees with a similar context. The system can recommend others who are now or have been in a similar situation within a certain time frame in the past. This is a very different use case from the first one as it does not only cover the current context, but also previous contexts.

– **Trigger actions based on context changes.** Especially for proactive system behaviour, it is important that context-aware applications get a timely notification that the context has changed. One example are process or task changes that can initiate learning processes [7].

Typical context sources in this scenario either determine higher-level abstraction of the user's situation by analyzing user interface events (e.g. via Bayesian Networks[8] or rule-based formalisms), or by applying heuristics to data in existing sources like personal information managers (e.g. Microsoft Outlook) or documents.

2.3 Requirements

A closer inspection of the application scenario has revealed the following requirements:

– **Aging.** It should be obvious that collected context is not valid indefinitely. If the system gets to know about the current "task" of the user, this information will only be valid for a limited amount of time. As a consequence, the user context management system needs to have some aging mechanism.
– **Variability in dynamic behavior.** The closer inspection of the "aging" problem reveals that aging is not uniform across the different aspects of user context information. While information like name, birthdate changes infrequently to never, other aspects like personal skills, interests goals evolve over time, and tasks or location are highly volatile. So the aging support has to be specific for the different parts of the context.
– **Scalability.** If we want to materialize user context information, we have to select methods that are scalable with respect to large numbers of users and long time frames.
– **Push and pull interaction paradigm.** The system must support both the push and the pull paradigm, i.e. it must be able to answer interactive queries and allows for trigger-style notifications.
– **Open-world assumption.** As we have seen, the mapping between the situation is incomplete, and application may require to be able to query also for the possibility of facts or may exploit negative facts directly. The most trivial case is explicit negative user feedback on user context information, but there are many techniques that can generate negative facts in order to increase the quality of the context. So the context data model must build on the open world assumption.

3 Layered Context Model

3.1 General Considerations

For traditional database management systems, it has proven effective to divide the management functionality into different layers which are basically independent of the internal logic of the lower layers. In that spirit, we have grounded our work on a three layer model (an initial version of which has been presented in [9]) that allows for structuring the problem in a better way (see figure 1).

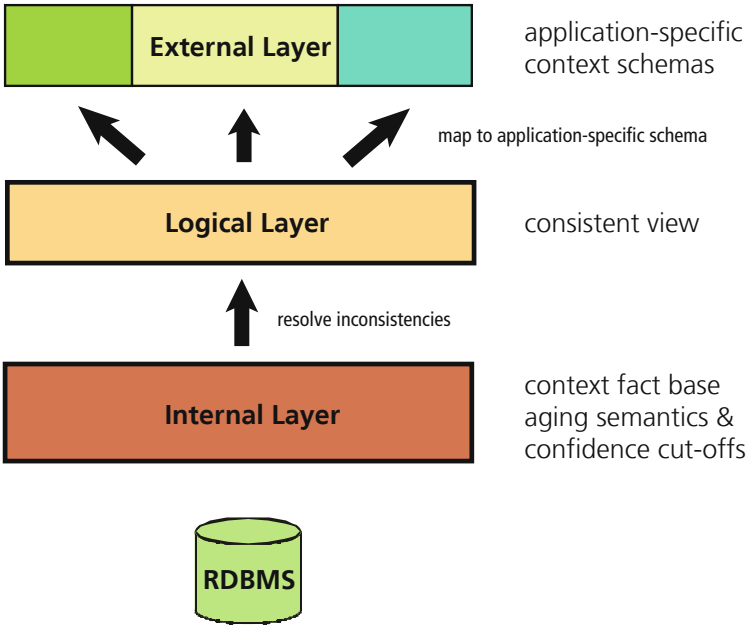


Fig. 2. Layers of the Context Data Model

- **Internal Layer.** The internal layer as the lowermost layer stores all collected information about users in a time-dependent way as so-called context facts. The context facts can be queried according to their timestamp or by using value-level operators (on different data types).
- **Logical Layer.** This layer provides a consistent view on the collected data, conforming to a (single) specified schema. For an specific instant in time, the service on this layer can provide a consistent and semantically enriched view (based on schema-level information or background knowledge) on the context facts that adheres to certain quality criteria.
- **External Layer.** The top-most layer represents the usage context of a particular application at a certain instant of time. The context information is in the schema the application understands, which could be different from the logical schema.

These layers must not be confused with an aggregation hierarchy. How to achieve aggregation and abstraction from lower level information to higher level information is presented in section 3.7.

3.2 Internal Layer

The internal layer represents the lowest level of abstraction. It provides the basic functionality required for storing and accessing collected context information together with meta information about time, validity and confidence of the collected data. More formally, a context fact is defined as follows:

Definition 1 (context fact). *A context fact is a tupel* $(U, f, o, v, t, valid, \alpha)$ *where*

- U *is a user*
- f *is context feature*
- $o \in \{=, \neq\}$ *signals a positive or negative fact*
- v *is a value*
- $valid$ *is the validity interval for the value*
- t *is the point in time at which the factum was added to the fact base.*
- α *is the probability that at point of time* t *the feature* f *has the value* v *for user* U.

The set of all context facts is called context fact base and denoted with C

The support for negative context facts directly results from the requirement for an open-world assumption. In practice, this can be used, e.g., for explicit user feedback on certain infered facts about her context.

Definition 2 (Context feature). *A context feature is a tuple* $f = (uri, T)$ *where* (uri) *is a unique identifier,* $V = (T, O)$ *a data type consisting of a value space* V *and operators* O

The data type can represent traditional atomic data types like integers, strings etc., but also ontological data (the supported operators will be discussed in section 4), as the following examples show: *(Andreas, performs-task, =, literature-search, [2005-04-15 10:00, ∞), 2005-04-15 10:00, 0.8)*, and an entry *(Andreas, performs-task, =, examine-students, [2005-04-14 14:00, ∞), 2005-04-15 13:00, 0.9)*.

As additional schema-level information, the internal layer has **aging functions** attached to each context feature, which allow for describing how the confidence in a certain value decreases over time. An **aging function** basically is a monotonically decreasing function $a : TIME \rightarrow [0, 1]$, which is multiplied with the initial confidence value in order to obtain the current confidence value. These aging functions can be assigned heuristically or – preferably – based on empirical results. Queries from the logical layer can use the current confidence, which is calculated for each fact: If $c = (U, f, v, o, t, valid, \alpha)$ is a context fact, t^* the instant in time of interest. Then the confidence for c at t^* can be calculated as:

$$confidence(c, t^*) := [A(f)](t^* - t) \cdot \alpha$$

with $A(f)$ denoting the aging function associated with the context feature f.

3.3 Logical Model

The internal layer is rather ugly to use for context-aware applications. This has mainly to do with the fact that you can have almost arbitrarily inconsistent data. In analogy to traditional databases, we need the notion of a schema guarantee. A context schema is comprised of (1) a definition of context features (as above), (2) cardinality constraints and (3) a feature hierarchy.

Cardinality constraints are a very effective instrument for checking for an elementary check of consistency. For many features, it is clear from the application semantics that

there can be only one value at a time so that any application also expects only a single value.

In order to allow for better reusing context information in different applications, the model also offers the possibility to define a *feature hierarchy* via feature inheritance, which directly corresponds to property hierarchies in RDF(S). This adds a basic inferencing capability to the model: if an applications requests the value(s) for a specific feature, the values of sub-features can be also returned. This can be formalized as follows:

Definition 3 (Feature hierarchy). *A **feature hierarchy** is an acyclical relation $H \subseteq (F \times F)$ on the set of context features F. Additionally, the following properties must hold for a feature hierarchy to be compatible with the feature set F:*

- $\forall(f_1, f_2) \in H$*: the value space of* f_2 *is a subset of the value space of* f_1
- $\forall(f_1, f_2) \in H$*: if* f_2 *is multi-valued then* f_1 *must also be multi-valued*

H^* is the transitive closure of H.

Based on this, the **context feature schema** C can be defined as $C = (F, card, H)$, where F is a set of context features, $card : F \rightarrow \{1, \infty\}$ is the cardinality assignment and H is a compatible feature hierarchy.

Definition 4 (Schema conformity). *A set $K = \{c_1, ..., c_n\}$ of context facts with $c_i = (U_i, f_i, v_i, \alpha_i)$ is conforming to a schema $C = (F, card, H)$ iff*

a) $\forall i \in \{1, ..., n\} : f_i \in F$ *and* v_i *is in the value space of* f_i*.*
b) With $values(f, U) := \{v | k = (U, f^*, ' =', v, \alpha) \in K, (f, f^*) \in H^*\}$ *the following must hold:* $\forall f \in F : |values(f)| \leq card(f)$
c) With $values^{\neg}(f, U) := \{v | k = (U, f^*, ' \neq', v, \alpha) \in K, (f, f^*) \in H^*\}$ *the following must hold:* $values(f, U) \cap values^{\neg}(f, U) = \emptyset$

This notion of schema conformity is essential for imposing a well-defined semantics ontop of imperfect data. It basically states that (a) we have only well-defined context features with associated data type definitions and that the facts conform to these data type constraints, (b) only multivalued features have multiple values, and (c) we have no contradictions resulting from positive and negative facts.

3.4 Mapping the Internal Layer to the Logical Layer

The main mapping task is the resolution of inconsistencies. Inconsistency occurs in our model if there are multiple values for a feature for which the cardinality constraints enforce a single value, or if we have positive and negative facts on the same feature. *Conflict resolution strategies* are responsible for transforming a set of context facts on the internal layer into a set of context facts conforming to the context schema. There can be different strategies to resolve these inconsistencies. The most obvious is to take the value with the highest confidence, but usually the strategy also needs to take into account that facts can be reinforced by other facts (e.g. two independent methods determine the same feature value within a limited time window).

If we apply this procedure to the example, it is clear that the restriction to a specific instant in time (e.g. *2005-04-14 11:00*) still provides two possible tasks. After applying the aging function, let's suppose that the *literature-search* has confidence 0.7 and *examine-students* has confidence 0.1. This would lead to a simple resolution strategy taking the *literature-search* as the current feature value, because we have specified that the *performs-task* feature is only single-valued.

As conflict resolution strategies, the maximum confidence strategy with some heuristic refinements has turned out to be quite effective for single-valued features. For multivalued context features, we are experimenting with strategies based on the Dempster-Shafer theory, which allows for aggregating probabilities from different sources [10].

Apart from conflict resolution, the mapping involves exploiting the feature hierarchy. This is done by rewriting a query for fact f^* to queries for the set of features $S_{f^*} = \{f \in F | (f, f^*) \in H^*\}$, i.e., all subfeatures.

3.5 The Problem of Asynchronous Notification

The most typical interaction pattern for context-awareness are publisher-subscriber patterns where the user context management service provides notifications about changes to the user context. Although we have apparently an append-only semantics on the internal layer (as typical for continuous query scenarios, [11]), there are two critical points resulting from the fact that the actual context is a time-dependent view:

- Aging makes the queries non-monotonous ([12]), i.e., it is not enough to provide additional results, but rather previous results have to be retracted. As a consequence, the notification protocol needs to incoporate both additions and removal of parts of the user context.
- Confidence-based filtering and conflict resolution are aggregation operators on the temporal data stream. These operators have to implemented in a way that they are – in most cases – self-maintainable so that with new arriving data, the changes to the view can be calculated without querying.

3.6 External Layer and Mapping from the Logical Layer

The external layer is intended to be interface for application, providing an application-specific view. In this step, the global context schema used on the logical layer is translated into an application-specific schema. This problem can be dealt with similar to schema mapping techniques in classical information integration approaches. In case of simple projections and renamings, this can be done within the user context management system, but for more powerful mapping features, external mapping services are the method of choice (in spirit of [13]).

A far more challenging problem is when we consider mappings not only on the level of user context schemas, but rather also on the value level, especially in case of ontology-based data types (see below).

3.7 Resulting System Architecture

From the model presented in this section, a high-level system architecture can be directly derived (see fig. 3.7). It shows the different layers at which the context

management service can be accessed and sketches also typical added-value services and other components.

Below the internal level, two different types of context sources are supported: push and pull context sources. Push context sources notify the context management service of changes; their context information is materialized inside the system. Pull context sources can be queries on demand as soon as queries for covered features arrive. The reasoning service provides access to ontological data.

Added-value services are useful to improve the context quality or to facilitate the management task. Currently, we have two types of services: agents that use the user context management system as a blackboard and services that are used by the management system. The most important agents are Augmentation Agents (inferring additional facts from already collected ones) and Subcontext Agents (trying to exploit dependencies in the context to improve context switching and thus the time needed to adapt to changes). Augmentation agents can either work on the raw context facts, or on the consolidated view, depending on the inferencing methods: the logical layer will be preferred by logic-based approaches, whereas imperfection-aware methods will prefer the internal layer.

Many user modeling systems concentrate on the problem of abstracting and aggregating lower-level context information to information on a higher level of abstraction. Although this is not the focus of this architecture (which is on imperfection handling), this can be easily realized using a hierarchy of user context management services. Lower level services are plugged in as pull context sources into higher level services.

4 Integrating Ontologies

4.1 Motivation

Approaches to context modelling like [14] or [15] and to applying context awareness to the e-learning and similar domains like [16], [17], [18] emphasize the potential of applying Semantic Web technologies to user context management. It enables the creation of more semantically aware processing methods, especially by introducing a shared vocabulary, which can be used across different tools and systems, and by applying reasoning techniques based on domain knowledge.

On the other side, Semantic Web technologies have still quite a way to go for solutions that are comparable in terms of scalability with traditional data management solutions. This is especially true for traditional types of queries like datatype specific range queries (e.g. for temporal data), although the description logics community tries to approach this problem with concrete domains (see e.g. [19]) and datalog-based reasoning (e.g. [20]). Also these techniques are not well-suited for highly dynamic scenarios with a high volume of changes as they so far do not consider update operations at all.

4.2 Approach

If we analyze our problem domain, it turns out that the benefits of ontologies are (beyond the ontology-like constructs like the feature hierarchy introduced in the last section) basically on the value level. We want to reference instances from the ontology in as

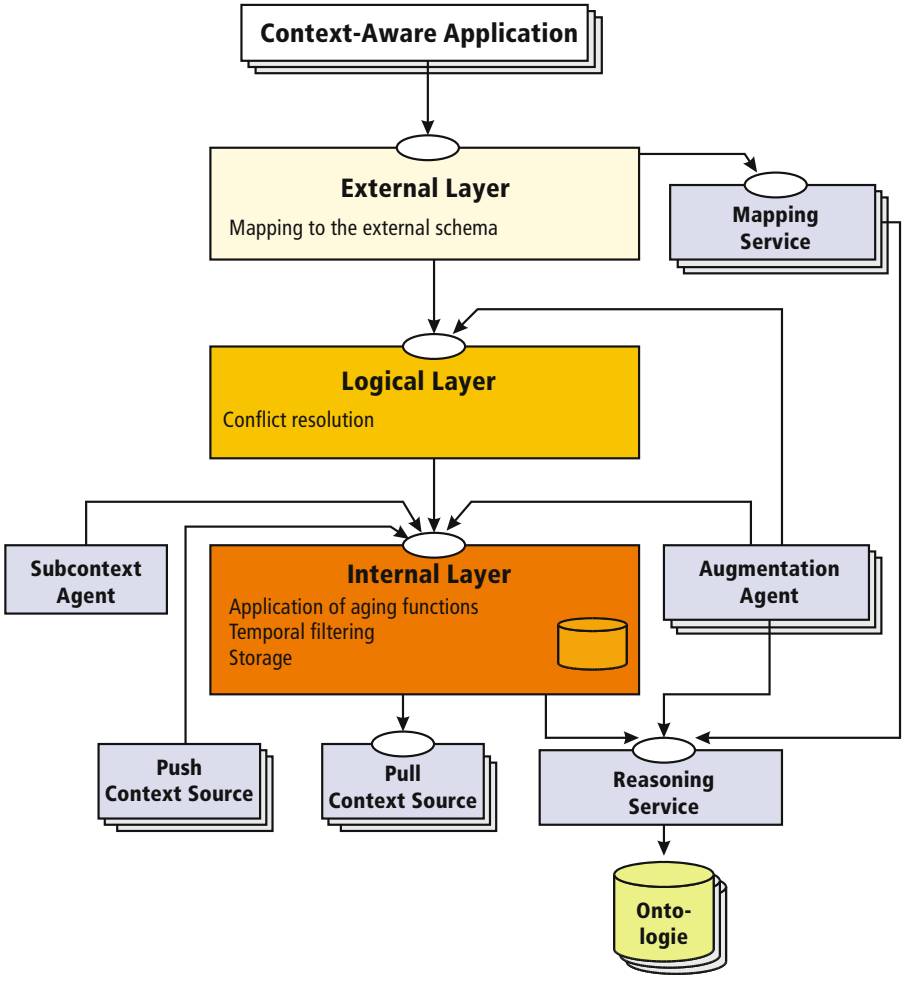


Fig. 3. High-level architecture of the user context management service

feature values (e.g. the task in the examples above or a competency from a competency catalog). Also the inferencing capabilities (i.e. mainly classification) of descriptions logics are mainly needed for queries like (a) is the user in a task of the type X or (b) retrieve all users with expert-level competency in a certain subject area.

So the idea is to treat ontologies as a datatype that has certain predicates that can be used in a query to the user context management system, just like operators on dates ($\leq$, *between ... and* etc.) or numbers or strings. As there is currently no notion of operators for description logics in spirit of the operators of a data model and especially no standardized query language for (although there are some proposals like OWL-QL [21] or SPARQL [22]), we started with the most obvious operator *instance-of*, which already covers a large portions of practical cases encountered in our application domain.

A sample query for users in *myDepartment* who were involved in a accounting process activity in August 2006 would be:

```
SELECT USERS
WHERE  process-activity instance-of AccountingActivity
       AND department = myDepartment
VALID  [2006-08-01,2006-08-31]
```

In the user context management service, this is handled by splitting the query into a query to the ontology service and a query to context fact base. Similar to join optimization, the processing order is determined based on result set estimations. In this case, a query to the ontology reasoner would give all instances of AccountingActitivity. These query would be rewritten as follows:

```
SELECT USERS
WHERE  (process-activity = a1
          OR process-activity = a2 ...)
       AND department = myDepartment
VALID  [2006-08-01,2006-08-31]
```

Alternatively, the query results of the user context fact base could be filtered with the help of the ontology, which is the preferred way if we have large sets of instances.

We are currently extending the approach with SPARQL subqueries to support more powerful navigation.

5 Implementation Case Study

The user context management infrastructure has been successfully used in the project *Learning in Process*, which was committed to implementing a learning support for context-steered learning, and further developments based on that system.

5.1 How the User Context Management Infrastructure Was Implemented

The user context management service was implemented ontop of a relational database management system. The first prototype was based on PostgreSQL, but in subsequent versions we have also used Oracle 10g XE. For the internal layer, the most challenging issue was the efficient implementation of the calculation of the current confidence. The main problem is that we cannot rewrite queries involving the calculated attribute *current-confidence* to range queries on the fact attributes. In order to avoid dynamic calculation on each query, we used a combination of techniques to speed it up, among them (a) using a subsuming query to prefilter the facts to calculate the confidence on by using approximations of the initial confidence, (b) historizing old facts and (c) using Oracle function-based indexes.

The implementation of the integration of ontologies is somewhat similar to approaches like [23] where ontologies are used for query rewriting, although the access of ontologies cannot be considered a preprocessing step. The query condition is

split into parts that can be shipped to the underlying database system and parts that are shipped to the reasoner. Furthermore, this approach is basically a join between context facts and ontology results. As a consequence, we tried to optimize the execution order by making use of result size estimates, e.g. by using statistics on the number of instances (for the *instance-of* operator). As an ontology management system, KAON [24] was used in the first prototype of the system, using Java-API access. Currently we are moving towards KAON2 [25] for OWL-DL and SPARQL support.

5.2 How User Context Was Used

As context features, a fairly broad range was used, which was divided into four categories: personal, social, organizational and technical. As personal context features, mainly learner preferences (semantic density, interactivity level) and competencies and interests were used. For characterizing the social context, we relied on a basic social relationship ontology. On the organizational level, we relied on organizational unit, role, business process activity and task. Technical features were user agent (browser, operating system, plugins), bandwidth, and availability of audio (input and output). Whereas the relationship between personal and technical context features and the available learning material was fairly straightforward, we needed additional background knowledge to relation organizational entities. This was mainly done by attaching competency requirements to organizational entities, which was encoded in an organizational ontology.

This context information was exploited by the so-called *Matching Service* which computes based on the background knowledge and the current user's context a competency gap and can compile personalized learning programs based on that gap that take into account the various aspects of of the user's context [5]. The Matching Service only operates on-demand. In order to be able to realize proactive behaviour (i.e., recommending learning material to users based on context changes), the architecture additionally consisted of a *Learning Assistant*, which subscribes to the user context management service for context changes. Whereas the Matching Service decides on *what to deliver*, the Learning Assistant decides on *when to deliver* and displays the recommendations in an unobtrusive manner.

5.3 How User Context Was Acquired

In contrast to prior work in the area of business-process-oriented knowledge management [26], we could not rely on a workflow management infrastructure to capture the organizational part of context. Rather, we experimented with a variety of heuristic sensors for application events, e.g., a plugin for Microsoft Office (relying on template information), a Browser Helper Object for Internet Explorer (for URL-based heuristics) and a plugin for Mozilla-based browsers. Additionally, we built interfaces to HR applications to extract the more static part of the organizational context.

For personal part of the context, we relied on static information from the user, but we are currently investigating the possibility of inferring the learner characteristics from application sensor data, e.g., by considering time of day, previous and upcoming meetings or other appointments etc.

For social relationships, we used mainly annotated address books as a (pull) context source, but similar to the personal level, we are currently investigating egocentric network analysis methods [27] to discover these relationships, e.g., from email conversations.

5.4 Results

After integrating the user context management service into the system, we conducted an evaluation with around 20 users at two companies. It has turned out that the system behaviour was perceived as useful by the evaluation participants.

On a more technical level, our tests have shown that the user context management service improves the robustness of the whole system in comparison to a naive approach in which we do not handle imperfection. Furthermore, the architecture has proven useful for plugging in and out different very loosely coupled context sources.

We are currently setting up a simulation environment for measuring the increase in quality and completeness with different assumptions about the context sources.

6 Related Work

6.1 Context Modeling

There are plenty of models for dealing with user context information, both from the traditional community of user modeling and the recently emerged communities for context-awareness. A good overview of recent context modeling approaches gives [28]. In general, it can be stated that the data management problem is a neglected area of research. Most approaches either ignore the problems of imperfection and dynamics (e.g. [29], [14], [15]), or assume that context can be accessed via the pull paradigm, which is certainly valid in sensor-based areas, but not appropriate for context information on a high level of abstraction. This can also be traced back to the fact that especially approaches to high-level context information rely on ontology-based techniques where imperfection is hard to integrate (although approaches exist, e.g. [30]).

6.2 Imperfection Handling in Context Modeling

The consideration of the imperfection and dynamics of user context information is also a relatively neglected area of research, especially for the case of high-level context information. [31] investigate quality criteria for context information complementing quality of service concepts. They define the following criteria: precision, confidence, trust level (for context sources), granularity and up-to-dateness. [32] introduce meta attributes like precision, certainty, last update and update rate, the approach of [33] is similar. Only [34] has investigated the role of imperfection in a more systematic way and identified the following types of imperfection: unknown values, contradictory values, imprecise values, and incorrect values. Feature values are further classified according to their source and persistence into sensed, static, profiled and derived. The causes of imperfection are analyzed along this classification. But all of these approaches do not consider

the implication on a management infrastructure, especially in terms of scalability when combined with ontology-based reasoning techniques.

6.3 Imperfection Handling in Data Management in General

The major part of research on handling imperfection in data management seems to be almost a decade ago (see e.g. [35], [36] and [37] for an overview). Apart from fuzzy logic, the major part of research in data management concentrated on probalistic extensions of the relational and other data models. Two main approaches can be identified: probabilistic attributes ([38], [39], [40]) and probabilistic relations (i.e. probabilities on tupel level) [41], [42]. Current approaches are mainly concerned with semistructured XML data (e.g. [43], [44]). However, these approaches did not consider the time dimension (i.e. the problem of aging).

There are some combinations of temporal and imperfection problems, but these approaches concentrate on the imperfection of the temporal perspective itself (e.g. [45], [46]), not on the impact of time distance on the quality.

7 Conclusions and Outlook

The approach of this paper views robust and scalable user context management as a key enabler for rolling out context-aware application in the large. In order to retain robustness in the presence of imperfection, the system needs to consciously manage the imperfect properties of the data ([47]). This approach covers the uncertainty (via attached probabilities), the incompleteness (via open-world semantics), and contradictions (via storing contradictory facts and conflict resolution strategies). Additionally, the approachs also accounts for the dynamics of change of context information by introducing aging functions that decrease the certainty over time. These aging functions are specific for context feature in order to deal with the variability in the rate of change between different aspects of the context. A layered approach helps to keep the complexity manageable and the architecture of the system extensible. Scalability is achieved through relying on traditional data management techniques and providing appropriate indexing structures for imperfection handling. Still it is possible to reference semantically rich ontologies as data types and accessing limited reasoning functionality within queries. The system has been successfully applied to a corporate learning scenario.

Central storage of user context data always raises (justified) privacy concerns. The presented architecture can be easily extended to support a distributed approach where context data is stored for each user separately, e.g., on her machine. There will be one single user context management service without any local storage that simply distributes (after checking the permission) the query to the individual context management systems, which are registered as pull context sources.

Future work will incorporate the research in a data type that captures imprecision via a probability distribution of that value, which is important for location information. A lot of previous research exists on that topic that can be integrated into the approach. Different conflict resolution strategies will also be evaluated with their effect on the context quality. For that purpose, agent-based simulation techniques will be used.

Acknowledgements

This work was partially supported by the European Commission under the Fifth Framework Programme of IST within the project *Learning in Process* (contract IST-2001-32518) and under the Sixth Framework Programme of IST within the project *AGENT-DYSL*.

References

1. Henricksen, K., Indulska, J.: Modeling and using imperfect context information. In: Second IEEE International Conference on Pervasive Computing and Communications. Workshop on Context Modelling and Reasoning (CoMoRea 04), IEEE Computer Society (2004) 33–37
2. Winograd, T.: Architectures for context. Human-Computer Interaction **16** (2001)
3. Henricksen, K., Indulska, J., Rakotonirainy, A.: Modeling context information in pervasive computing systems. In Mattern, F., Naghshineh, M., eds.: Pervasive 2002, Berlin, Springer (2002) 167–180
4. Schmidt, A.: Bridging the gap between knowledge management and e-learning with context-aware corporate learning solutions. In Althoff, K.D., Dengel, A., Bergmann, R., Nick, M., Roth-Berghofer, T., eds.: Professional Knowledge Management. Third Biennial Conference, WM 2005, Kaiserlautern, Germany, April 2005. Revised Selected Papers. Volume 3782 of Lecture Notes in Artificial Intelligence., Springer (2005) 203–213
5. Schmidt, A.: Context-steered learning: The Learning in Process approach. In: IEEE International Conference on Advanced Learning Technologies (ICALT '04), Joensuu, Finland, IEEE Computer Society (2004) 684–686
6. Schmidt, A.: Potentials and challenges of context awareness for learning solutions. In: LWA 2005: Lernen–Wissensentdeckung–Adaptivität, 13th Annual Workshop of the SIG Adaptivity and User Modeling in Interactive Systems (ABIS 2005), Saarbrücken. (2005)
7. Schmidt, A., Winterhalter, C.: User context aware delivery of e-learning material: Approach and architecture. Journal of Universal Computer Science (JUCS) **10** (2004) 28–36
8. Horvitz, E., Breese, J., Heckermann, D., Hovel, D., Rommelse, K.: The lumière project: Bayesian user modeling for inferring the goals and needs of software users. In: 14th International Conference on Uncertainty in Artificial Intelligence, Madison, Wisconsin (1998) 256–265
9. Schmidt, A.: A layered model for user context management with controlled aging and imperfection handling. In Roth-Berghofer, T.R., Schulz, S., Leake, D.B., eds.: Modeling and Retrieval of Context. Proceedings of the 2nd International Workshop on Modeling and Retrieval of Context MRC 2005, Edinburgh, Scotland, July 31 - August 1, 2005. Number 3946 in Lecture Notes in Artificial Intelligence (2006)
10. Ruthven, I., Lalmas, M.: Using dempster-shafer's theory of evidence to combine aspects of information use. Journal of Intelligent Systems **19** (2002) 267–302
11. Babu, S., Widom, J.: Continuous queries over data streams. SIGMOD Rec. **30** (2001) 109–120
12. Terry, D., Goldberg, D., Nichols, D., Oki, B.: Continuous queries over append-only databases. In: SIGMOD '92: Proceedings of the 1992 ACM SIGMOD international conference on Management of data, New York, NY, USA, ACM Press (1992) 321–330
13. Kazakos, W., Nagypal, G., Schmidt, A., Tomczyk, P.: Xi3 - towards an integration web. In: 12th Workshop on Information Technology and Systems (WITS '02), Barcelona, Spain (2002)

14. Wang, X., Gu, T., Zhang, D., Pung, H.: Ontology based context modeling and reasoning using owl. In: IEEE International Conference on Pervasive Computing and Communication (PerCom'04), Orlando, Florida. (2004)
15. Strang, T., Linnhoff-Popien, C., Frank, K.: CoOL: A Context Ontology Language to enable Contextual Interoperability. In Stefani, J.B., Dameure, I., Hagimont, D., eds.: LNCS 2893: Proceedings of 4th IFIP WG 6.1 International Conference on Distributed Applications and Interoperable Systems (DAIS2003). Volume 2893 of Lecture Notes in Computer Science (LNCS)., Paris/France, Springer Verlag (2003) 236–247
16. Nebel, I., Smith, B., Paschke, R.: A user profiling component with the aid of user ontologies. In: Workshop Learning - Teaching - Knowledge - Adaptivity (LLWA 03), Karlsruhe. (2003)
17. Heckmann, D.: A specialized representation for ubiquitous computing and user modeling. In: First Workshop on User Modeling for Ubiquitous Computing, UM 2003. (2003)
18. Dolog, P., Nejdl, W.: Challenges and benefits of the semantic web for user modelling. In: AH2003 Workshop at WWW2003. (2003)
19. Lutz, C.: Description logics with concrete domains - a survey. In Balbiani, P., Suzuki, N.Y., Wolter, F., Zakharyaschev, M., eds.: Advances in Modal Logics Volume 4. King's College Publications (2003)
20. Hustadt, U., Motik, B., Sattler, U.: Reasoning in description logics with a concrete domain in the framework of resolution. In: Proc. of the 16th European Conference on Artificial Intelligence (ECAI 2004), August, 2004, Valencia, Spain. (2004) 353–357
21. Fikes, R., Hayes, P., Horrocks, I.: Owl-ql: A language for deductive query answering on the semantic web. Journal on Web Semantics **2** (2005)
22. Prud'hommeaux, E., Seaborne, A.: Sparql query language for rdf. W3C Working Draft 20. February 2006, W3C (2006)
23. Necib, C.B., Freytag, J.C.: Query processing using ontologies. In: Proceedings of the 17th Conference on Advanced Information Systems Engineering (CAISE'05), Porto, Portugal. (2005)
24. Maedche, A., Motik, B., Stojanovic, L.: Managing multiple and distributed ontologies in the semantic web. VLDB Journal **12** (2003) 286–302
25. Hustadt, U., Motik, B., Sattler, U.: Reducing shiq-description logic to disjunctive datalog programs. In: Principles of Knowledge Representation and Reasoning: Proceedings of the Ninth International Conference (KR2004), Whistler, Canada, June 2-5, 2004. (2004) 152–162
26. Abecker, A., Bernardi, A., Hinkelmann, K., Kühn, O., Sintek, M.: Context-aware, proactive delivery of task-specific information: The knowmore project. DFKI GmbH International Journal on Information Systems Frontiers (ISF) **2** (2000) 139–162
27. Fisher, D.: Using egocentric networks to understand communication. IEEE Internet Computing **2005** (2005) 20–28
28. Strang, T., Linnhoff-Popien, C.: A context modeling survey. In: Workshop on Advanced Context Modelling, Reasoning and Management, UbiComp 2004 - The Sixth International Conference on Ubiquitous Computing, Nottingham/England. (2004)
29. Meissen, U., Pfennigschmidt, S., Voisard, A., Wahnfried, T.: Context- and situation-awareness in information logistics. In Lindner, W., Mesiti, M., Türker, C., Tzitzikas, Y., Vakali, A., eds.: Current Trends in Database Technology - EDBT 2004 Workshops, EDBT 2004 Workshops PhD, DataX, PIM, P2P&DB, and ClustWeb,Heraklion, Crete, Greece, March 14-18, 2004, Revised Selected Papers. Volume 3268 of Lecture Notes in Computer Science., Springer (2004) 335–344
30. Nottelmann, H., Fuhr, N.: pdaml+oil: A probabilistic extension to daml+oil based on probabilistic datalog. In: Information Processing and Management of Uncertainty in Knowledge-Based Systems, Perugia, Italy (2004)

31. Buchholz, T., Küpper, A., Schiffers, M.: Quality of context: What it is and why we need it. In: 10th International Workshop of the HP OpenView University Association (HPOVUA 2003), Geneva, Switzerland. (2003)
32. Judd, G., Steenkiste, P.: Providing contextual information to ubiquitous computing applications. In: 1st IEEE Conference on Pervasive Computing and Communication (PerCom 03), Fort Worth. (2003) 133–142
33. Heckmann, D.: Ubiquitous User Modeling. PhD thesis, Universität des Saarlandes (2005)
34. Henricksen, K., Indulska, J.: A software engineering framework for context-aware pervasive computing. In: PerCom, IEEE Computer Society (2004) 77–86
35. Motro, A.: Management of uncertainty in database systems. In Kim, W., ed.: Modern Database Systems: the Object Model, Interoperability and Beyond. Addison-Wesley/ACM Press (1994) 457–476
36. Motro, A.: Sources of uncertainty, imprecision and inconsistency in information systems. In Motro, A., Smets, P., eds.: Uncertainty Management in Information Systems: From Needs to Solutions. Kluwer Academic Publishers (1996) 9–34
37. Parsons, S.: Current approaches to handling imperfect information in data and knowledge bases. IEEE Transactions on Knowledge and Data Engineering **8** (1996) 353–372
38. Barbará, D., García-Molina, H., Porter, D.: The Management of Probabilistic Data. ACM Transactions on Knowledge and Data Engineering **4** (1992) 487–502
39. Dey, D., Sarkar, S.: A probabilistic relational model and algebra. ACM Transactions on Database Systems **21** (1996) 339–369
40. Lakshmanan, L.V.S., Leone, N., Ross, R., Subrahmanian, V.: Probview: A flexible probabilistic database system. ACM Transactions on Database Systems **22** (1997) 419–469
41. Cavallo, R., Pittarelli, M.: The theory of probabilistic databases. In: VLDB '87: Proceedings of the 13th International Conference on Very Large Data Bases, San Francisco, CA, USA, Morgan Kaufmann Publishers Inc. (1987) 71–81
42. Fuhr, N., Rölleke, T.: A probabilistic relational algebra for the integration of information retrieval and database systems. ACM Transactions on Information Systems **15** (1997) 32–66
43. Nierman, A., Jagadish, H.V.: Protdb: Probabilistic data in xml. In: Proceedings of the 28th VLDB Conference, Hong Kong, China, 2002. (2002)
44. Hung, E., Getoor, L., Subrahmanian, V.S.: Pxml: A probabilistic semistructured data model and algebra. In: Proceedings of the 19th International Conference on Data Engineering, March 5-8, 2003, Bangalore, India, IEEE Computer Society (2003) 467–
45. Dyreson, C., Snodgrass, R.: Supporting valid-time indeterminacy. ACM Transactions on Database Systems **23** (1998) 1–57
46. Dekhtyar, A., Ross, R., Subrahmanian, V.: Probabilistic temporal databases i: Algebra. ACM Transactions on Database Systems **26** (2001) 41–95
47. Lockemann, P.C., Lukacs, G.: Imperfection and the human component: Adding robustness to global information systems. In Brinkkemper, S., Lindencrona, E., Slvberg, A., eds.: Information Systems Engineering: State of the Art and Research Themes. Springer (2000) 3–14

Moving Towards Automatic Generation of Information Demand Contexts: An Approach Based on Enterprise Models and Ontology Slicing

Tatiana Levashova[1], Magnus Lundqvist[2], and Michael Pashkin[1]

[1] St.Petersburg Institute for Informatics and Automation of the Russian Academy of Sciences, 39, 14th line, St.Petersburg, 199178, Russia
{oleg, michael}@mail.iias.spb.su
[2] School of Engineering, Jönköping University,
Gjuterigatan 5, SE-551 11 Jönköping, Sweden
magnus.lundqvist@ing.hj.se

Abstract. This paper outlines the first experiences of an approach for automatically deriving information demands in order to provide users with demand-driven information supply and decision support. The presented approach is based on the idea that information demands with respect to work activities can be identified by examining the contexts in which they exist and that a suitable source for such contexts are Enterprise Models. However, deriving contexts manually from large and complex models is very time consuming and it is therefore proposed that a better approach is to, based on an Enterprise Model, produce a domain ontology and from this then automatically derive the information demand contexts that exist in the model.

Keywords: information demand; enterprise model; ontology engineering; ontology management; context; context derivation.

1 Introduction

As an attempt to solve many of the problems related to the identification, distribution, and management of information necessary for the daily operation of most knowledge intensive organizations today [1], Fraunhofer ISST and the Technical University of Berlin in Germany established the area of Information Logistics (ILOG) in 1997. Since then a lot of effort has been put into the development of the area as such as well as different systems, techniques, and methods for improving information flow as well as reducing information overflow related problems within organizations by providing the right user with the right information at the right time and place [2]. In recent year the Information Engineering Group at Jönköping School of Engineering in Sweden has joined in the effort to provide organizations with such demand-driven approaches to information supply. While ILOG certainly has proven to be a valid attempt at solving these problems as illustrated by several successful applications [3], it is only recently any deeper consideration has been given to the information demand aspects of ILOG. This paper builds upon the ideas and results from the work on establishing a context-based approach to information demand analysis and modeling intended to

R. Meersman, Z. Tari et al. (Eds.): OTM 2006, LNCS 4275, pp. 1012 – 1019, 2006.

facilitate the design and implementation of ILOG-solutions. Section 2 of this paper will give a short overview of the area of context-based information demand modeling while section 3 focuses on how Enterprise Models can be used as a source for deriving such contexts. However, as it will be shown in section 3.1, some aspects of information demand context derivation are rather difficult and time consuming and are therefore from a practical perspective not very suitable to perform manually. Due to this it has been suggested that the commonalities between context-based demand modeling and context- and ontology-based decision support as discussed in section 4, can be utilized in the development of a technique for deriving such contexts automatically [4, 5].

The paper ends with a short discussion and some conclusions in section 5 based on the experiences from the first steps taken towards automatic derivation of information demand context by combing the two above described approaches in order to provide users with demand-driven information supply and decision support.

2 Context-Based Information Demand Modeling

The initial steps towards demand modeling taken within the area of ILOG, while definitely suitable for its purposes were quite simplistic. Furthermore, they were taken without any clear definition of the concept of information demand. Profile-based demand modeling, successfully used when developing the WIND application, an German early warning system for severe storms, where the users information demands are capture and represented in user profiles [3] works fairly well for systems that tend to have users with small and relatively static demands but for more complex situations like ERP-applications and cross- or inter-organizational systems it is simply not flexible enough. Furthermore, such an approach also tends to be application specific, something that makes it unsuitable for larger systems spanning complete application domains. Yet another attempt to solve the problem of representing information demands is the situation-based approach [6]. By dividing a user's daily schedule into several situations that comprises a duration of time, topics of interests, and if relevant, location, as well as technical resources available for receiving or retrieving information it is possible to decide under which circumstances a user can manage information needed to perform specific tasks. While this approach indeed is suitable for deciding on when to send information to a user it does not really deal with the issue of how to initially decide on what information that is considered relevant in relation to what task.

To remedy these shortcomings as well as facilitate the analysis and representation of information demand a context-based approach has been suggested together with a definition of the Information Demand concept [7]. It has also been suggested that a suitable source from which such information demand contexts can be derived are Enterprise Models [8].

2.1 Information Demand Context

While this paper will not cover all aspects of information demand context relevant to context-based analysis and modeling of information demand the basic idea is that it,

in order to be able to support business activities by providing integrated information, which is the intended purpose of demand-driven information supply [7], is essential to understand those activities and the setting in which they exist, i.e. their context. That is to say that providing the "right information at the right time and place" entails that it should be right given the demanding party's context. Context can be, and has been, defined in many different ways but for the purpose of information demand analysis, context is here simply defined as [8]:

An Information Demand Context is the formalized representation of information about the setting in which information demands exist and comprises the organizational role of the party having the demand, work activities related, and any resources and informal information exchange channels available, to that role.

As stated by this definition it is clearly role that is considered to be the central concept when analyzing and modeling information demands. Information demand context allows for all activities performed by a specific role to be grouped together with any relevant resources for doing so.

3 Deriving Information Demand Context from Enterprise Models

Enterprise Modeling has been described as the art of externalizing enterprise knowledge. This is usually done with the intention to either add some value to an enterprise or share some need by making models of the structure, behavior and organization of that enterprise [9]. The motivation often given to why enterprises should be analyzed and modeled is that it facilitates, and to some degree is a prerequisite for, better management, coordination and integration of such diverse aspects of an enterprise as markets, processes, different development and manufacturing sites, components, applications/systems as well as contributes to an increased flexibility, cleaner and more efficient manufacturing etc. Enterprise models usually include such diverse aspects like business processes, technical resources, information flow, organizational structures, and human resources. Additional aspects might also be included but those listed above are considered to be the essential ones. Deriving contextual information can of course be done in many different ways and from many different sources it is however here claimed that one such particular suitable source for deriving the contextual information necessary for information demand analysis and modeling is Enterprise Models. If business processes are considered to be sequences of activities, technical resources and information to be resources, organization to be the structure in which roles can be identified it corresponds well to the different aspects of context listed in section 2.1 and thus, such models cover all aspects of an organization relevant when producing information demand contexts.

3.1 Example Model Used for Automatic Context Derivation

To exemplify how information demand contexts can be derived from Enterprise Models a model describing many different aspects of a fictitious bike producing

enterprise has been used. Fig. 1 provides an overview of this model with the intention of illustrating the complexity of such models. Even though fairly extensive it still incorporates relatively few roles, processes, and resources and hardly any of the relationships between different elements in the model are shown. With this in mind it should not come as a surprise that real-life models tend to be considerably larger than the model shown here. Extraction of information demand context is basically a question of analyzing the model and single out all activities within the process descriptions connected to a specific role and then determining what resources that is required to perform those activities. Once all relevant aspects of the context are identified they can be represented as a sub-model.

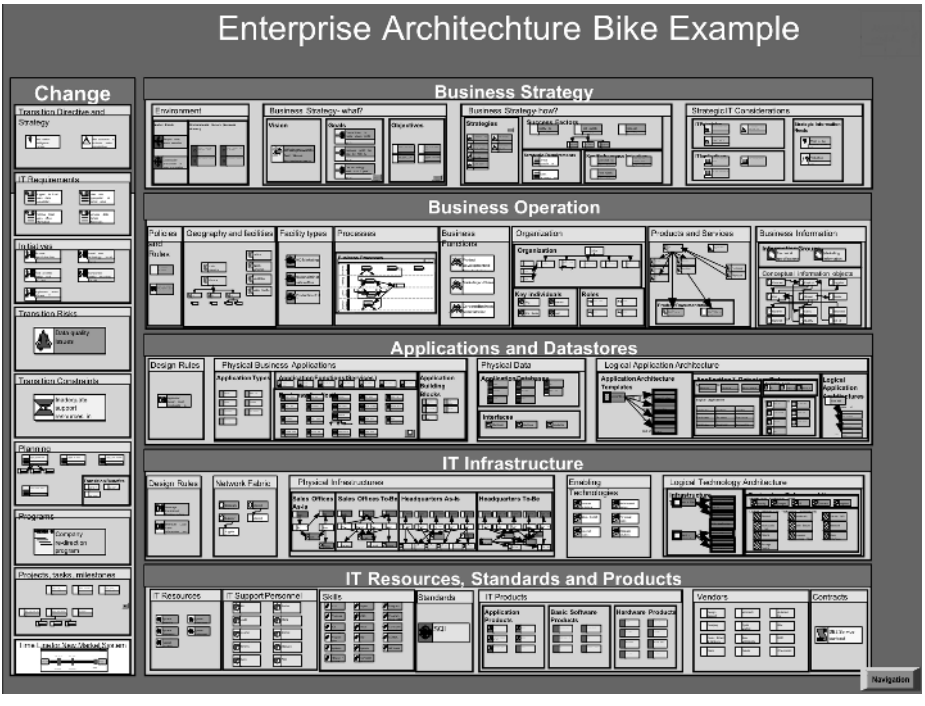


Fig. 1. Example of an Enterprise Model from a fictitious enterprise

Up to this point research done by the authors has focused mainly on understanding the knowledge relevant to demand modeling contained in Enterprise Models rather than the development of more sophisticated methods and techniques for extracting it. As a consequence the manual approach used so far is insufficient with respect to time consumption even though it works fairly well from demand identification perspective. However, producing a context on the needed level of detail for each task performed by each role within a larger organization would clearly be unpractical. As an attempt to automate the process an evaluation of the similarities between the context-based approach to information demand modeling, just presented, and theories from

context-based decision support has been performed. As it became clear that the two approached shared most of the concepts central to the problem at hand [4] it also became clear that an automated approach, as will be shown in the following sections, would not only be possible but also make sense.

4 An Ontology-Based Approach to Context Derivation

The research made on ontology-driven context derivation was originally intended create a way to represent problems formulated by a user as requests to a decision support system (DSS). Since context often is defined as any information that can be used to characterize the situation of an entity [10] and is employed as a way to organize such information [11, 12] in combination with the fact that DSS has to take into account current and relevant information provided by the environment in which it exists the choice of representing problems in terms of context seemed reasonable.

The problem or the current situation in DSS is modeled in terms of context. Two types of context are used: abstract and operational. *Abstract context* is an ontology-driven model integrating information and knowledge relevant to the problem. *Operational context* is an instantiation of the abstract context with relevant data provided by information sources [13].

The context is created through extraction of knowledge relevant to the problem from a domain ontology. It is a rather large ontology describing a macro-situation (e.g., e-business, disaster relief, tourism, etc.) and integrates knowledge from various knowledge sources. The problem or the current situation in which decision-making takes place is considered to be a subtype of the macro-situation.

A generalized scenario of DSS functioning follows the steps of 1) identification of the knowledge relevant to the problem formulated in the request to DSS; 2) creation of ontology-based problem model (abstract context); 3) instantiation the problem model with data values (operational context); 4) generation of object-oriented constraint network (OOCN) formalizing the problem by a set of constraints; and 5) solving the problem as a Constraint Satisfaction Problem (CSP).

For the purposes of the research presented here only items 1 and 2 above are of interest. The identification of relevant knowledge is carried out using a set of algorithms that implement ontology slicing operation. The intention of this operation is to extract pieces of knowledge from the domain ontology, which is believed to be relevant to the request. For this purpose, the request vocabulary is first matched against the domain ontology vocabulary. The words from the domain ontology vocabulary matching to words from the user request serves as “seeds” for the slicing operation. The algorithms capture knowledge surrounding “seeds” so as the captured knowledge would be relevant to the user request and to the inference supported by OOCN.

Generally, the result of the slicing operation is more than one ontology slice. This is explained by the fact that ontology concepts to be captured can belong to a different taxonomy branch. If such concepts are not related with any other relationships but

taxonomical ones they are considered to belong to different slices. Several slices integrated into a single piece of knowledge constitute abstract context.

In order to formalize the problem via a set of constraints and to interpret it as a CSP formalism OOCN is used as a mean for representing the ontology [14]. The set of constraints supported by the formalism is (1) *taxonomical* ("is-a") relationships, (2) *hierarchical* ("part-of") relationships, (3) *class cardinality* restriction, (4) *class compatibilities*, (5) *associative* relationships, and (6) *functional* relations.

To create an ontology-driven context and apply the DSS approach an ontology was derived from the Enterprise Model presented in Fig. 1. This ontology serves as the domain ontology described above. Roles will be used as "seeds" (the input) for the slicing operation. The purpose of this operation is to extract and combine knowledge considered relevant for specific roles. In terms of the above-presented approach the context is an ontology-driven model that characterizes the situation of a role.

4.1 Ontology Based on Enterprise Model

The tool used for modeling the EM (Fig. 1) uses XML as an internal representation for storing the information it contains. A file containing such an XML-representation was used as the source from which the ontology has been derived. Table 1 shows correspondences between the EM, its representations in XML, and OOCN formalism. The table also shows that not all types of constraints that OOCN supports are necessary. Constraints on class compatibility, class cardinality, and functional constrains are omitted.

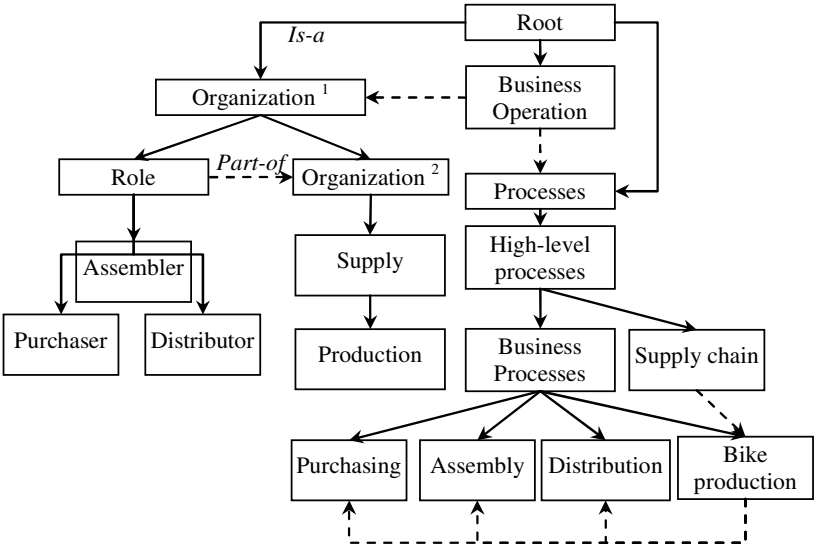
Table 1. Correspondence between Enterprise Model, XML, and OOCN representations

EM	XML representation	OOCN
Object	Object	Class
Instance	Valueset	A set of attributes
Range[1]	Data type	Domain
Parent-children relationship	Type-of; Child-link; Part-link	"Is-a" constraint
Named relationships but "Type-of"/"Of-type"	Relationship	Associative constraint

4.2 Resulting Information Demand Context

In the model (Fig. 1) four types of roles are defined: "Production Planner", "Assembler", "Purchaser", and "Distributor". Fig. 2 illustrates parts of the abstract contexts for "Assembler", "Purchaser", and "Distributor" roles. The parts are distinguished by the presence of a class representing a particular role and corresponding business processes. E.g., for class "Assembler" class "Assembly" is given, whereas classes "Purchaser", "Distributor", "Purchasing", and "Distribution" are omitted. It should be noted that even though the classes represented in Fig. 2 are interrelated by many associative relationships in the EM, only the hierarchical relationships are shown here.

[1] Ranges of *string*, *integer*, *float*, *time*, and *date* data types are used in EM.



[1] A hierarchically organized Organization with a number of formal positions and a number of roles that can be performed by individuals or Organization Units, as well as some key people for the kind development that the model is covering.
[2] Organization units that constitute the business and some of its high-level positions. Positions are formal elements of an Organization.

Fig. 2. Part of abstract contexts for roles of Assembler", "Purchaser", and "Distributor"

5 Conclusions and Outlook

While the ideas presented in this paper have been both tested and used separately and proven to be suitable for its intents and purposes respectively the work on combining the two approaches described here should only be viewed as the first stumbling steps towards automatic creation of ontology-based information demand contexts. It is undoubtedly so that several problems remain to be solved but nevertheless the result indicates that further work might prove to be both interesting and valuable for both research areas. If a working method for automatic context derivation could be developed it would greatly influence the applicability of information demand analysis on large and complex organizations. Furthermore, automatic ontology-based context creation not only provides DSS with a more convenient way to understand the setting of the problems such systems aim at solving but also increase the likelihood of such systems focusing on issues relevant to their users. From an Information Logistical perspective having users information demands represented in terms of ontology also make sense since this facilitates the possibilities to match demands against information managed by such systems. A way to express knowledge about organizations as an ontology without having to produce it manually could potentially prove to be very valuable from many perspectives.

References

1. Delphi Group: Perspectives on Information Retrieval, Boston, Mass., Delphi Group (2002)
2. Deiters, W., Löffeler, T., and Pfenningschmidt, S.: The Information Logistical Approach Toward a User Demand-driven Information Supply. In Spinellis, D. (ed.): Cross-Media Service Delivery Boston, Mass./Dordrecht/London: Kluwer Academic Publisher (2003)
3. Jaksch, S., Pfenningschmidt, S,, Sandkuhl, K., Thiel, C.: Information Logistic Applications for Information-on-Demand Scenarios: Concepts and Experiences from WIND project. In *Proceedings of the 29th Euromicro Conference* (2003)
4. Lundqvist, M., Sandkuhl, K., Levashova, T., Smirnov, A.: Context-Driven Information Demand Analysis in Information Logistics. In *Proceedings of the first International Workshop on Context and Ontologies: Theory, Practice and Applications*. Pittsburgh, Pennsylvania, USA: AAAI Press (2005) 124-127
5. Levashova, T., Lundqvist, M., Sandkuhl, K., Smirnov, A.: Context-based Modelling of Information Demand: Approaches from Information Logistics and Decision Support. To be published in *Proceedings of the 14th European Conference on Information Systems*. Gothenburg, Sweden (2006)
6. Meissen, U., Pfennigschmidt, S., Sandkuhl K., Wahnfried, T.: Situation-based Message Rating in Information Logistics and its Applicability in Collaboration Scenarios. In *Euromicro 2004 Special Session on "Advances in Web Computing"*. IEEE Computer Society Press (2004)
7. Lundqvist, M. and Sandkuhl, K.: Modeling Information Demand for Collaborative Engineering. In *Proceedings of 2nd Intl. Workshop on Challenges in Collaborative Engineering*. VEDA. Stara Lesna, Slovakia (2004) 111-120
8. Lundqvist, M.: Context as a Key Concept in Information Demand Analysis. In Proceedings of the Doctoral Consortium associated with the 5th Intl. and Interdisciplinary Conference on Modelling and Using Context (CONTEXT-05). Paris, France (2005) 63-73
9. Vernadat, F., B.: Enterprise Modeling and Integration (EMI): Current Status and Research Perspectives. Annual Reviews in Control 26. (2002) 15–25
10. Dey, A. K., Salber, D., Abowd, G. D.: A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications. In: Moran, T. P., Dourish, P. (eds): Context-Aware Computing, A Special Triple Issue of Human-Computer Interaction, Vol. 16. Lawrence-Erlbaum (2001) http://www.cc.gatech.edu/fce/ctk/pubs/-HCIJ16.pdf
11. Brézillon, P.: Context in Artificial Intelligence: I. A Survey of the Literature. In: Computer & Artificial Intelligence. Vol. 18, no. 4 (1999) 321—340
12. Brézillon P. Context in Artificial Intelligence: II. Key Elements of Contexts. In: Computer & Artificial Intelligence. Vol. 18, no. 5 (1999) 425—446
13. Smirnov, A., Pashkin, M., Chilov, N., Levashova, T.: Constraint-driven methodology for context-based decision support. In: Journal of Decision Systems. Vol. 14, no. 3, Special Issue on Design, Building and Evaluation of Intelligent DMSS. Lavoisier (2005) 279—301
14. Smirnov, A., Pashkin, M., Chilov, N., Levashova, T., Haritatos F.: Knowledge Source Network Configuration Approach to Knowledge Logistics. In *International Journal of General Systems.* Taylor & Francis Group, Vol. 32, no. 3. (2003) 251-269

Semantic Similarity of Ontology Instances Tailored on the Application Context

Riccardo Albertoni and Monica De Martino

CNR-IMATI,
Via De Marini, 6 – Torre di Francia - 16149 Genova, Italy
{albertoni, demartino}@ge.imati.cnr.it

Abstract. The paper proposes a framework to assess the semantic similarity among instances within an ontology. It aims to define a sensitive measurement of semantic similarity, which takes into account different hints hidden in the ontology definition and explicitly considers the application context. The similarity measurement is computed by combining and extending existing similarity measures and tailoring them according to the criteria induced by the context. Experiments and evaluation of the similarity assessment are provided.

1 Introduction

In this decade, the ontologies have been imposing in the computer science as artefact to represent explicitly shared conceptualisations. A remarkable research effort has been spent to develop new ontology languages, proper reasoning mechanisms and correlated management tools. Less attention has been posed instead on the similarity among the ontology instances. Methods to assess similarity among instances are needed to exploit the knowledge modelled in the ontology in different research fields pertaining the Knowledge Management such as Data Mining and Information Visualization. They should consider as much as possible the implicit information encoded in the ontology as they provide useful hints to define the similarity. Moreover, they should be sensible to specific contexts inasmuch as different contexts induce different criteria of similarity.

So far, the most of research activity pertaining to similarity and ontologies has been carried out within the field of ontology alignment or to assess the similarity among concepts. Unfortunately, all these methods result inappropriate for the similarity among instances. On the one hand the similarities for the ontology alignment strongly focus on the comparison of the structural parts of distinct ontologies and their application to assess the similarity among instances might result misleading. On the other hand, the concepts' similarities mainly deal with lexicographic database ignoring the comparison of the instances values. Apart from them, few methods to assess similarities among instances have been proposed. Unfortunately these methods rarely take into account the different hints hidden in the ontology and they do not consider that the ontology entities differently concur in the similarity assessment according to the application contexts.

To overcome the limitations mentioned above the paper proposes a framework to assess the semantic similarity among instances. Its contribution is twofold. Firstly, the

R. Meersman, Z. Tari et al. (Eds.): OTM 2006, LNCS 4275, pp. 1020 – 1038, 2006.

framework provides a measurement of semantic similarity more sensitive to the hints hidden in the ontology. It is defined by an amalgamation function, which combines and extends different similarities already defined in literature: it takes into account both the structural comparison between two instances in terms of the classes that the instances belong to, and the instances comparison in term of their attributes and relations. Secondly, the framework provides the parametric evaluation of the similarity with respect to different applications. The application induces the criteria of similarity which are explicitly formalized in the application context. An application context models the importance of the entities, which concur in the assessment of similarity, and the operation used to compare the instances. The parametric evaluation allows to tailor the similarity assessment to specific application contexts, but also to obtain different similarity assessments employing the same ontology.

The paper is organised as follows. In the first section, we illustrate the main principle of the approach. Then a formalization of the similarity criteria induced by the context is proposed. The remaining sections are devoted to the definition of the similarity functions which characterise our method followed by two experiments and an evaluation of the results. At the end, we evaluate the related works underlining our contributions.

2 Semantic Similarity Method

The paper proposes a semantic similarity among instances within an ontology taking into account the different hints hidden in the ontology and the application context. As the hints that can be considered largely depend on the level of formality of the ontology model adopted, it is important to state clearly to which ontology model a similarity method is referring. In the paper, the ontology model with data type defined by Ehrig et al.[1] is considered.

Definition 1: Ontology with Data Type. *An Ontology with data type is a structure* $O := (C, T, \leq_c, R, A, \sigma_R, \sigma_A, \leq_R, \leq_A, I, V, l_C, l_T, l_R, l_A)$ *where* C,T,R,A,I,V *are disjointed sets respectively of classes, data types, binary relations, attributes, instances and data values, and the relations and functions are defined as follows:*

$\leq_C$	the partial order on C, which defines the classes hierarchy,
$\leq_R$	the partial order on R which defines the relation hierarchy,
$\leq_A$	the partial order on A which defines the attribute hierarchy,
$\sigma_R : R \to C \times C$	the function that provides the signature for each relation,
$\sigma_A : A \to C \times T$	the function that provides the signature for each attribute,
$I_C : C \to 2^I$	the function called class instantiation,
$I_T : T \to 2^V$	the function called data type instantiation,
$I_R : R \to 2^{I \times I}$	the function called relation instantiation,
$I_A : A \to 2^{I \times V}$	the function called attribute instantiation.

Two kinds of similarity exist with symmetric or with asymmetric properties. A symmetric normalized similarity $S : IxI \to [0,1]$ is a function that maps a pair of instances to a real number in the range [0,1] such that:

$$\forall x, y \in I \quad S(x,y) \geq 0 \qquad \textit{Positiveness}$$
$$\forall x \in I, \forall y, z \in I, S(x,x) \geq S(y,z) \qquad \textit{Maximality}$$
$$\forall x, y \in I \quad S(x,y) = S(y,x) \qquad \textit{Symmetry}$$

An asymmetric normalized similarity is a function $\overline{S} : IxI \rightarrow [0,1]$ that does not satisfy the symmetric axiom. The preference between symmetric and asymmetric similarity mainly depends on the application scenario, there is no a-priori reason to formulate this choice. A complete framework to assess the semantic similarity should provide both of them. In the paper only the asymmetric similarity is described due to lack of space.

The proposed model adopts the schematisation of the similarity framework defined by Ehrig et.al. [1]: they structure the similarity in terms of *data, ontology* and *context* layers plus the *domain knowledge* layer which spans all the other. The *data layer* measures the similarity of entities by considering the data values of simple or complex data types such as integer and string. The *ontology layer* considers the similarities induced by the ontology entities and the way they are related each other. The *context layer* assesses the similarity according to how the entities of the ontology are used in some external contexts. The framework defined by Ehrig et al. is suitable to support the ontology similarity as well as instances similarity.

Our contribution with respect to the framework defined by Ehrig et al. is mainly in the definition of a *context layer* including an accurate formalization of the criteria to tailor the similarity with respect to a context and in the definition of an *ontology layer* explicitly parameterised according to these criteria. Concerning the data and domain knowledge layers the paper adopts a replica of what is illustrated in [1].

The formalization of the criteria of similarity induced by the context is employed to parameterise the computation of the similarity in the *ontology layer*, forcing it to adhere to the application criteria.

The overall similarity is defined by the following amalgamation function ($\overline{Sim}$) which aggregates two similarity functions defined in the ontology layer named *external similarity* ($\overline{ExternSim}$) and *extensional similarity* ($\overline{ExtensSim}$). The external similarity performs a structural comparison between two instances $i_1 \in l_c(c_1)$, $i_2 \in l_c(c_2)$ in terms of the classes c_1, c_2 the instances belong to, whereas the extensional similarity performs the instances comparison in term of their attributes and relations.

$$\overline{Sim}(i_1, i_2) = \frac{w_{ExternSim} * \overline{ExternSim}(i_1, i_2) + w_{ExtensSim} * \overline{ExtensSim}(i_1, i_2)}{w_{ExternSim} + w_{ExtensSim}} \qquad (1)$$

$w_{ExternSim}$ and $w_{ExtensSim}$ are the weights to balance the functions importance. By default they are equal to 1\2. In the below sections the Context Layer is described as well as the two similarities $\overline{ExternSim}$ and $\overline{ExtensSim}$.

3 Context Layer

The context layer, according to Ehrig at al. [1], describes how the ontology entities concur in different contexts. The paper adopts this point of view. However it aims to formalize the application context in the sense of modelling the criteria of similarity

induced by the context. This design choice does not hamper to define eventually a generic description of context and then to determine automatically which criteria would have been suitable for a given context. Rather, it allows to calculate directly the similarity acting on the criteria especially when it is necessary to refine them. In the following we underlay the importance of this formalization and we provide it.

3.1 Motivation Behind the Application Context Formalization

The application context provides the knowledge to formalise the criteria of similarity induced by the application. Criteria are context-dependent as the context influences both the choice of classes, attributes and relations to be considered in the similarity assessment and the operations to compare them.

We describe the motivation behind the proposed formalization through an example. Let consider a simplified version of the ontology KA[1] that defines concepts from academic research (Fig 1) and focus on the two applications: "comparison of the members of the research staff according to their working experience" and "comparison of the members of the research staff with respect to their research interest". Two distinct application contexts can be induced according the applications:

- "Exp" induced by the comparison of the members of the research staff according to their working experience. The similarity among the members of the research staff (instances of the class *ResearchStaff*[2]) is roughly assessed considering the member's age (the attribute *age* inherited by the class *Person*), the number of projects and publications a researcher has worked on (the number of instances reachable through the relation *publication* and relation *workAtProject* inherited by *Staff*).
- "Int" induced by the comparison of the members of the research staff with respect to their research interest. The researchers can be compared with respect to their interest (instances reachable through the relation *interest*), and again the publications (instances reachable through the relation *publications*), the projects (instances reachable through the relation *workAtProject*).

Analysing these examples the follows considerations can be pointed out:

1. the similarity between two instances can depend on the comparison of their related instances: the researchers are compared with respect to the instances of the class *Publication* connected through the relation *publications*;
2. the attributes and relations of the instances can differently contribute in the evaluation according to the context: the attribute *age* of the researchers is functional in the first application but it might not be interesting in the second; the relations *publication* and *workAtProject* are included in both the application contexts but using different operator of comparison: in the first case just the number of instances is important whereas in the latter the related instances have to be compared;
3. the ontology entities can be considered recursively in the similarity evaluation: in the context "Int" the members' research topic (instances of *ResearchTopic* reachable

[1] http://protege.stanford.edu/plugins/owl/owl-library/ka.owl

[2] The italics is used to explicit the reference to the entities (attributes, relations, classes) of the ontology in Fig 1.

navigating through the relation *ResearchStaff->interest*[3]*)* are considered and their related topics (instances of *ResearchTopic* reachable via *ResearchStaff->interest->relatedTopic*) are recursively compared to assess the similarity of distinct topics;

4. the classes' attributes and relations can differently contribute in the evaluation according to the recursion level of the assessment: in the second application the attribute *topicName* and the relation *relatedTopic* can be considered at the first level of recursion to assess the similarity between *researchTopic*. By navigating the relation *relatedTopic* it is possible to apply another step of recursion, and here the similarity criteria can be different from the previous ones, for example in order to limit the computational cost and stop the recursion, only the *topicName* or the instances identifier could be adopted to compare the *relatedTopic*.

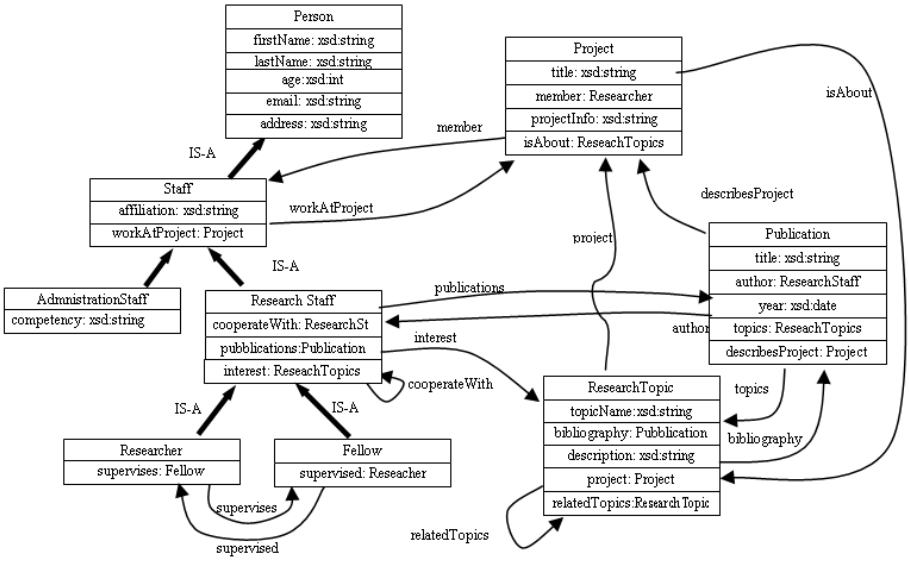


Fig. 1. Ontology defining concepts related to the academic research

As pointed out in the second remark, there are different operations that can be used to compare the ontology entities:

- operation based on the "cardinality" of the attributes or relations: the similarity is assessed according to the number of instances the relations have, or the number of values that an attribute assumes. For example in the first context **"**Exp" two researchers are similar if they have a similar "number" of publications;
- operation based on the "intersection" between sets of attributes or relations: the similarity is assessed according to the number of elements they have in common. For example in the context "Int" the more papers two researchers share, the more their interests are similar;

[3] The arrow is used to indicate the navigation through a relation, for example *A->B->C* means that starting from the class *A* we navigate through the relations B and C.

- operation based on the "similarity" of attributes and relations: the similarity is assessed in terms of similarity of the attributes values and related instances. For example, in the context **"Int"** two researchers are similar if they have "similar" research topics.

The example evidences that an accurate formalism is needed to properly express the criteria which might arise from different application contexts. The formalization has to model the attributes and relations as well as the operation to compare their values. Moreover, as noticed in the fourth remark also the level of recursion of the similarity assessment has to be considered.

3.2 Application Context Formalization

The formalization provided in the sequel represents the restrictions that the application context must adhere to. An ontology engineer is expected to provide the application context according to specific application needs. The formalization relies on the concepts of "sequence of elements belonging to a set X" which formalizes generic sequences of elements and "path of recursion of length i" to track the recursion during the similarity assessment. In particular, a "path of recursion" represents the recursion in terms of sequence of relations used to navigate the ontology.

The application context function (*AC*) is defined inductively on the length of the path of recursion. It returns the set of attributes and relations as well as the operations to be used in the similarity assessment. The considered operations are those illustrated in the previous paragraph and named respectively *Count* to evaluate the cardinality, *Inter* to evaluate the intersection, *Simil* to evaluate the similarity.

Definition 2: Sequences of a Set X. *Given a set X, a sequence s of elements of X with length n is defined by the function* $s:[1,..,n]\to X, n\in N^+$ *and represented in simple way by the list [s(1),..,s(n)].*

Let mark $S_X^n=\{s \mid s:[1,n]\to X\}$ the set of sequences on X having length n and $\cdot: S_X^n x S_Y^m \to S_{X\cup Y}^{n+m}$ the operator "concat" between two sequences.

Let define in Table 1 the polymorphic functions which identify specific sets of entities in the ontology model.

Table 1. List of functions defining specific sets of elements in the ontology model

$\delta_a: C\to 2^A; \delta_a(c)=\{a:A \mid \exists\, t\in T, \sigma_A(a)=(c,t)\}$	set of attributes of $c\in$ C ,
$\delta_a: R\to 2^A; \delta_a(r)=\{a:A \mid \exists c,c'\in C\ \exists\, t\in T\ \sigma_R(r)=(c,c') \wedge\sigma_A(a)=(c',t)\}$	set of attributes of the classes which are reachable through the relation r∈ R,
$\delta_r: C\to 2^R; \delta_r(c)=\{r:R \mid \exists\, c'\in C, \sigma_R(r)=(c,c')\}$	set of relations of $c\in$ C ,
$\delta_c: R\to 2^C; \delta_c(r)=\{c':C \mid \exists\, c\in C\ \ \sigma_R(r)=(c,c')\}$	set of concepts reachable through r∈ R,
$\delta_r: R\to 2^R; \delta_r(r)=\{r':R \mid \exists\, c\in C, \exists\, c'\in \delta_C(r); \sigma_R(r')=(c',c)\}$	set of relations of the concepts reachable through r ,
$\delta_c: C\to 2^C; \delta_c(c)=\{c':C \mid \exists\, r\in \delta_r(c); \sigma_R(r)=(c,c')\}$	set of concepts related to $c\in$ C through a relation.

Definition 3: Path of Recursion. *A path of recursion p with length i is a sequence whose first element is a class and the other are relations recursively reachable from the class:* $p \in S^{i}_{C \cup R} \mid p(1) \in C \wedge \forall j \in [2,i]\ \ p(j) \in R \wedge p(j) \in \delta_r(p(j-1))$.

For example of path of recursion with length longer than three is a path which starts from a class (p(1)) and continues in one of its relations as second element p(2), in one of the relations of the class reachable from p(2) as third element p(3) and so on. In general, a path of recursion p represents a path to be followed to assess the similarity recursively. The recursion expressed in the previous paragraph in the context "Int" as *ResearchStaff->interest->relatedTopic* is formalised with the path of recursion [ResearchStaff, interest, relatedTopic].

Let name P^i the set of all paths of recursion with length i and P the set of all paths of recursion $P = \cup_{i \in N} P^i$.

Definition 4: Application Context AC. *Given the set P of paths of recursion,* $L = \{Count, Inter, Simil\}$ *the set of operations adopted, an application context is defined by a partial function AC having signature* $AC : P \rightarrow (2^{A \times L}) \times (2^{R \times L})$ *returning the attributes and relations as well as the operations to perform their comparison.*

In particular, each application context AC is characterised by two operators $AC_A : P \rightarrow 2^{A \times L}$ and $AC_R : P \rightarrow 2^{R \times L}$ which return respectively the part of context AC related to the attributes and the relations. Formally $\forall p \in P\, AC(p) = (AC_A(p),\ AC_R(p))$ and $AC_A(p)$ and $AC_R(p)$ are set of pairs $\{(e_1,o_1), (e_2,o_2),\ldots, (e_i,o_i),\ldots,(e_n,o_n)\}$ $n \in N$ where e_i is respectively the attribute or the relation relevant to define the similarity criteria and $o_i \in L$ is the operation to be used in the comparison.

We provide two examples of AC formalization referring to the two application contexts "Exp", "Int" mentioned in the previous paragraph.

Example 1. Let formalise the application context "Exp" with AC_{Exp} to assess the similarity among the members of a research staff according to their experience. We consider the set of paths of recursion {[ReasearchStaff], [Reasearch], [Fellow]} and we compare them according to the age similarity, the number of publications and projects. Thus AC_{Exp} is defined by:

$$\begin{aligned}
&[\text{ResearchStaff}] \xrightarrow{AC_{Exp}} \{\{(\text{age},\text{Simil})\},\{(\text{publications},\text{Count}),(\text{workAtProject},\text{Count})\}\}\\
&[\text{Researcher}] \xrightarrow{AC_{Exp}} \{\{(\text{age},\text{Simil})\},\{(\text{publications},\text{Count}),(\text{workAtProject},\text{Count})\}\}\\
&[\text{Fellow}] \xrightarrow{AC_{Exp}} \{\{(\text{age},\text{Simil})\},\{(\text{publications},\text{Count}),(\text{workAtProject},\text{Count})\}\}
\end{aligned} \tag{2}$$

An example of AC_R is {(publication,Count),(workAtProject,Count)} while an example of AC_A is {(age,Simil)}.

Let note that [Researcher] and [Fellow] belong to the set of path of recursion considered in AC_{Exp} because their instances are also instance of *ResearchStaff*. The application context can be expressed in a more compact way assuming that whenever a context is not defined for a class but is defined for its super class, the comparison criteria defined for a super class are by default inherited by the subclasses. According to this assumption AC_{Exp} can be expressed through,

$$[\text{ResearchStaff}] \xrightarrow{AC_{Exp}} \{\{\text{age}, \text{Simil}\}, \{(\text{publications}, \text{Count}), (\text{workAtProject}, \text{Count})\}\} \quad (3)$$

Example 2. Let formalise the application context "Int" to assess the similarity among the members of a research staff according to their research interest. The similarity is computed considering the set of path of recursion {[ResearchStaff],[ResearchStaff, interest]}. The researchers are compared considering common publications, common projects or similar interests. A compact formalization for "Int" is defined by AC_{Int}:

$$\begin{aligned} &[\text{ResearchStaff}] \xrightarrow{AC_{Int}} \{\{\phi\}, \{(\text{publications}, \text{Inter}), (\text{workAtProject}, \text{Inter}), (\text{interest}, \text{Simil})\}\} \\ &[\text{ResearchStaff}, \text{interest}] \xrightarrow{AC_{Int}} \{\{\text{topicName}, \text{Inter}\}, \{(\text{relatedTopics}, \text{Inter})\}\} \end{aligned} \quad (4)$$

Let note that the researchers are compared recursively: [ReseachStaff,interest] is the path of recursion to navigating the ontology from *ResearchStaff* to *ResearchTopic* via the relation *interest.* The interests are compared with respect to both their *topicName* and their *relatedTopic*, thus two *ResearchTopic*(s) having distinct *topicName* but some *relatedTopic* in common are not considered completely dissimilar.

The image of an AC function can be further characterized:

1. For a path of recursion p, AC has to return only the attributes and relations belonging to the classes reached through p. For example, considering the ontology in fig 1 and the path of recursion [ReseachStaff,interest] it is expected that only the attributes and relations belonging to the class *ResearchTopic* reachable via [ReseachStaff,interest], can be identified by AC([ReseachStaff,interest]). Attributes or relations (as *age, publications, etc)* which do not belong to *ResearchTopic* define an incorrect application context.
2. Given a path of recursion p, an attribute or a relation can appear in the context image at most one time. In other words, given a path of recursion it is not possible to associate two distinct operations to the same relation or attribute. For example the following application context definition is not correct as *interest* is specified twice

$$[\text{ResearchStaff}] \longrightarrow \{\{\phi\}, \{(\text{publications}, \text{Inter}), (\text{interest}, \text{Simil}), (\text{interest}, \text{Inter})\} \quad (5)$$

4 Ontology Layer

The ontology layer defines the asymmetric similarity functions $\overline{ExternSim}$ and $\overline{ExtensSim}$ which compose the amalgamation function (formula 1). The "external similarity" $\overline{ExternSim}$ measures the similarity at the level of the ontology schema computing a structural comparison of the instances: given two instances, it compares the classes they belong to considering the attributes and relations shared by the classes and their position within the class hierarchy. The "extensional similarity" $\overline{ExtensSim}$ compares the extension of the ontology entities: the similarity is assessed by computing the comparison of the attributes and relations of the instances.

At the ontology layer additional hypotheses are assumed:

- All classes defined in the ontology have the fake class *Thing* as super-class.
- Given $i_1 \in l_c(c_1)$, $i_2 \in l_c(c_2)$, if c_1, c_2 do not have any common super-class different from *Thing,* their similarity is equal to 0.
- The least upper bound (*lub)* between c_1 and c_2, is unique and it is c_2 if c_1 IS-A c_2, or c_1 if c_2 IS-A c_1, otherwise the immediate super-class of c_1 and c_2 that subsumes both classes.

The aim is to force the *lub* to be a sort of "template class" which can be adopted to perform the comparison of the instances whenever the instances belong to distinct classes. Referring to the ontology in Fig 1, it can be appropriate to compare two instances belonging respectively to *AdministratorStaff* and *ResearchStaff* as they are both a kind of staff and *Staff* is their *lub*. However, it does not make sense to evaluate the similarity between two instances belonging to *Publication* and to *Staff,* because they are intimately different: actually there is not any *lub* available for them.

Whenever a *lub* x between the two classes exists, the path of recursion [x] is the starting path in the recursive evaluation of the similarity.

4.1 External Similarity

The external similarity ($\overline{ExternSim}$) performs the structural comparison between two instances i_1, i_2 in terms of the classes c_1, c_2 the instances belong to: more formally $\overline{ExternSim}(i_1, i_2) = \overline{ExternSim}(c_1, c_2)$ where $i_1 \in l_c(c_1), i_2 \in l_c(c_2)$.

In the paper the external similarity function is defined starting from the similarities proposed by Maedche and Zacharias [2] and Rodriguez and Egenhofer [3]. The structural comparison is performed by two similarity evaluations:

- **Class Matching.** which is based on the distance between the classes c_1, c_2 and their depth respect to the hierarchy induced by $\leq_C$.
- **Slot Matching.** which is based on the number of attributes and relations shared by the classes c_1, c_2 and the overall number of their attributes and relations. Then two classes having a plenty of attributes/relations, some of whose are in common, are less similar than two classes having less attributes but the same number of common attributes/relations.

Both similarities are needed to evaluate the similarity with respect to the ontology structure with success. For example, let us consider the ontology schema in Fig. 2 and let compare an instance of the class D with an instance of the class E.

They are quite similar with respect to the class matching but less similar with respect to the slot matching. At the fact, the sets of IS-A relations joining the classes D and E to *Thing* are largely shared. However, from the point of view of the slots, D and E share only the attribute A_1 and relation $\underline{C}_1$ and they differ with respect to the others. Likewise it would be easy to show an example of two classes similar with respect to the slots matching and dissimilar according to the class matching.

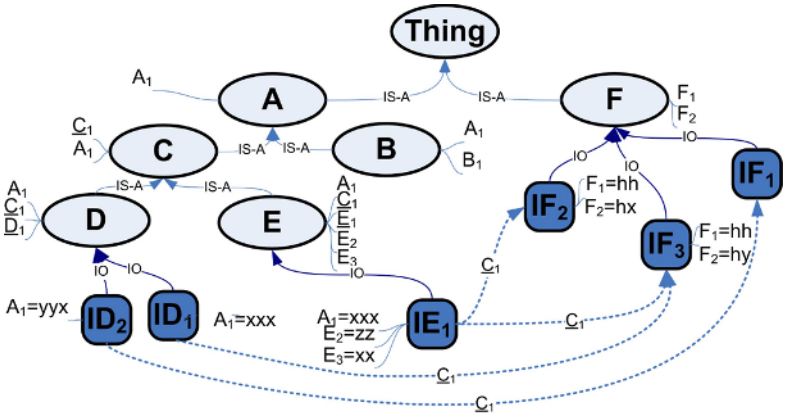


Fig. 2. Class hierarchy example: A, B, C, D, E, F are classes, A_1,B_1, E_2, E_3, F_1, F_2 are attributes, $\underline{C}_1$, $\underline{D}_1$, $\underline{E}_1$ are relations, ID_1, ID_2, IE_1, IF_1, IF_2, IF_3 are instances

Definition 5: ExternSim similarity. *The similarity between two classes according to the external comparison is defined by:*

$$\overline{ExternSim}(c_1,c_2)=\begin{cases}1 & if\ \ c_1=c_2\\ \dfrac{w_{SM}*\overline{SM}(c_1,c_2)+w_{CM}*\overline{CM}(c_1,c_2)}{w_{SM}+w_{CM}} & Otherwise\end{cases} \tag{6}$$

where ($\overline{SM}$) is the Slots Matching, ($\overline{CM}$) is the Classes Matching and w_{SM}, w_{CM} the respectively weights in the range [0,1].
w_{SM} and w_{CM} are defined for the purpose of this paper equal to 1\2.

4.1.1 Class Matching

Classes Matching is evaluated in terms of distance of the classes with respect to the IS-A hierarchy. The distance is based on the concept of Upwards Cotopy (*UC*)[2]. We define an asymmetric similarity adapting the symmetric definition of CM in [2] .

Definition 6: Upward Cotopy (UC). *The Upward Cotopy of a set of classes C with the associated partial order $\leq_C$ is:*

$$UC_{\leq_C}(c_i) := \{c_j \in C \mid (c_i \leq_C c_j) \vee c_i = c_j\} \tag{7}$$

It is the set of classes composing the path to reach from c_i the furthest super-class (*Thing*) of the IS-A hierarchy: for example considering the class D in Fig. 2 $UC_{\leq_C}(D)=\{\mathrm{D,C,A,Thing}\}$.

Definition 7: Asymmetric Class Matching. *Given two classes c_1, c_2, the Upward Cotopy $UC_{\leq_C}(c_i)$, the asymmetric Class Matching is defined by:*

$$\overline{CM}(c_1,c_2) := \frac{\left|UC_{\leq_C}(c_1)\cap UC_{\leq_C}(c_2)\right|}{\left|UC_{\leq_C}(c_1)\right|} \tag{8}$$

$\overline{CM}$ between two classes depends on the number of their common classes in the hierarchy. Let note that the class matching is asymmetric, for example referring to Fig. 2, $\overline{CM}(B,D) = 2/3$ but $\overline{CM}(D,B) = 2/4$. Moreover, it is important to note that $\overline{CM}(A,D) = 1$, the rationale behind this choice of design is that the instances of D are suitable as instances of A.

4.1.2 Slot Matching

Slot Matching is defined by the slots (attributes and relations) shared by the two classes. We refer to the similarity proposed by Rodriguez and Egenhofer [3] based on the concept of distinguishing features employed to differentiate subclasses from their super-class. In their proposal, different kinds of distinguishing features are considered (i.e. functionalities, and parts) but no one coincides immediately with the native entities in our ontology model. Of course it would be possible to manually annotate the classes adding the distinguishing features but we prefer to focus on what is already available in the adopted ontology model. Therefore only attributes and relations are mapped as two kinds of distinguishing features.

Definition 8: Slot Matching. *Given two classes c_1,c_2, two kinds of distinguishing features (attributes and relations), w_a, w_r, the weights of the features, the similarity function $\overline{SM}$ between c_1 and c_2 is defined in terms of the weighted sum of the similarities $\overline{S}_a$ and $\overline{S}_r$, where $\overline{S}_a$ is the slot matching according to the attributes and $\overline{S}_r$ in the slot matching according to the relations.*

$$\overline{SM}(c_1,c_2) = \omega_a \cdot \overline{S}_a(c_1,c_2) + \omega_r \cdot \overline{S}_r(c_1,c_2) \tag{9}$$

The sum of weights is expected to be equal to 1, and by default we assume to be $w_a=w_r=1/2$. The two slot matching $\overline{S}_a$ and $\overline{S}_r$ rely on the definitions of *slot importance* as defined in the following.

Definition 9: Function of "Slot Importance" α. *Let c_1, c_2, be two distinct classes, d the class distance $d(c_1,c_2)$ in term of the number of edges in a IS-A hierarchy, α is the function that evaluates the importance of the difference between the two classes.*

$$\alpha(c_1,c_2) = \begin{cases} \dfrac{d(c_1,\mathrm{lub}(c_1,c_2))}{d(c_1,c_2)} & d(c_1,\mathrm{lub}(c_1,c_2)) \le d(c_2,\mathrm{lub}(c_1,c_2)) \\ 1 - \dfrac{d(c_1,\mathrm{lub}(c_1,c_2))}{d(c_1,c_2)} & d(c_1,\mathrm{lub}(c_1,c_2)) > d(c_2,\mathrm{lub}(c_1,c_2)) \end{cases} \tag{10}$$

Where $d(c_1,c_2) = d(c_1,\mathrm{lub}(c_1,c_2)) + d(c_2,\mathrm{lub}(c_1,c_2))$.

$\alpha(c_1, c_2)$ is a value in the ranges [0,0.5]. Referring to the image Fig. 2, α(D,C) is equal to zero because the lub between D and C is C itself, d(C,D)=1 and d(C,C)=0. Whereas α(D,E) is equal to 0.5 because the lub is still C, and d(D,E)=2.

Definition 10: Slot Matching according to the kind of distinguishing feature t. *Given two classes c_1 (target) and c_2, (base), t a kind of distinguishing feature (t=a for*

attributes or t=r for relations), let be C_1^t *and* C_2^t *the sets of distinguishing features of type t respectively of* c_1 *and* c_2*; the Slot Matching* $\overline{S}_t(c_1,c_2)$ *is defined by:*

$$\overline{S}_t(c_1,c_2)=\frac{\left|C_1^t\cap C_2^t\right|}{\left|C_1^t\cap C_2^t\right|+\alpha(c_1,c_2)\left|C_1^t\setminus C_2^t\right|+(1-\alpha(c_1,c_2))\left|C_2^t\setminus C_1^t\right|} \tag{11}$$

According to the ontology in Fig. 2, considering the classes D and E their sets of distinguishing features of type relation are $D^r=\{\underline{C}_1,\ \underline{D}_1\}$ and $E^r=\{\underline{C}_1,\ \underline{E}_1\}$ and $\alpha(D,E)=0.5$; then $\overline{S}_r(D,E)=0.5$.

In general, whenever $\alpha=0.5$ the difference of the features of both classes are equally important for the matching: for example it happens when the classes are sisters as in the case of D and E. In the case $\alpha=0$ only the features that are in c_2 and not in c_1 are important for the matching. In particular it happens, whenever c_2 is the subclass of c_1, in this case the matching is inversely proportional to the higher number of features of c_2 compared to those of c_1 .

4.2 Extensional Similarity

The extension of entities plays a fundamental role in the assessment of the similarity among the instances, it is needed to perform a comparison of the attribute and relation values. For example, in the ontology in Fig. 2 relying only on the structural comparison it is not possible to assess that ID_1 is more similar to IE_1 than to ID_2. The main principle of the proposed extensional similarity between two instances is to consider the lub x of their classes as the common base to compare them when the instances belong to different classes: it is adopted to define the path of recursion [x] from which starts the recursive assessment induced by an application context.

For example, considering the instances ID_1 and IE_1 in Fig. 2, the class C is their lub. Then the initial path of recursion from which to start the similarity assessment is [C]. Let us suppose to have already defined an application context as the follow [C]→{{(A_1,*Iter*)},{($\underline{C}_1$, *Simil*)}}; [C,$\underline{C}_1$]→{{(F_1,*Simil*)},{}}. The computation starts from the values of attribute A_1 for the instances ID_1 and IE_1, then through the relation $\underline{C}_1$ the new path of recursion [C,$\underline{C}_1$] is considered to compare the instances related to $\underline{IE}_1$ and $\underline{ID}_1$ with respect to the values of the attribute F_1.

The extensional comparison is characterised by two similarities functions: a function based on the comparison of the attributes of the instances and a function based on the comparison of the relations of the instances.

Definition 11: Extensional Asymmetric Similarity. *Given two instances* $i_1\in l_c(c_1)$, $i_2\in l_c(c_2)$, $c=lub(c_1,c_2)$, $p=[c]$ *a path of recursion. Let* $\overline{Sim}_a^p(i_1,i_2)$ *and* $\overline{Sim}_r^p(i_1,i_2)$ *be the similarity measurements between instances considering respectively their attributes and their relations. The extensional similarity with asymmetric property is defined:*

$$\overline{ExtensSim}(i_1,i_2)=\begin{cases}1 & i_1=i_2\\ \overline{Sim}_I^p(i_1,i_2) & Otherwise\end{cases} \tag{12}$$

Where $\overline{Sim_I}^p(i_1,i_2)$ *is defined by:*

$$\overline{Sim_I}^p(i_1,i_2)=\frac{\sum_{a\in\delta_a(c)}\overline{Sim_a}^p(i_1,i_2)+\sum_{r\in\delta_r(c)}\overline{Sim_r}^p(i_1,i_2)}{|\delta_a(c)|+|\delta_r(c)|} \tag{13}$$

Let note that the index p is a kind of stack of recursion adopted to track the navigation of relations whenever the similarity among instances is recursively defined in terms of the related instances. $\overline{Sim_a}^p(i_1,i_2)$ and $\overline{Sim_r}^p(i_1,i_2)$ are defined by a unique equation as following.

Definition 12: Similarity on Attributes and Relations. *Given two instances* $i_1\in l_c(c_1)$, $i_2\in l_c(c_2)$, $c=lub(c_1,c_2)$, $p=[c]$ *a path of recursion, X a placeholder for the "A" or "R",* $x\in A\cup R$ *let be:*

- $i_A(i)=\{v\in V\mid (i,v)\in l_A(a),\exists y\in C\ s.t.\ \sigma_A(a)=(y,T)\wedge l_T(T)=2^V\}$ *the set of values assumed by the instance i for the attribute a,*
- $i_R(i)=\{i'\in l_c(c')\mid\exists c\ i\in l_c(c)\exists c'\ s.t.\ \sigma_R(r)\in(c,c')\wedge(i,i')\in l_R(r)\}$ *the set of instances related to the instance i by the relation r,*
- *AC the application context defined* according *to the restrictions defined in paragraph 0*
- $F_X=\{g: i_X(i_1)\to i_X(i_2)\mid g\ is\ partial\ and\ bijective\}$

The similarity between instances according to their attributes or relations is defined:

$$\overline{\mathrm{Sim}}_x^p(i_1,i_2)=\begin{cases}0 & if\ ((x,Simil)\in AC_X(p)\wedge (i_X(i_1)\vee i_X(i_2)\ \text{are empty sets})\\ 0 & If\ \neg(\exists l\in L\ s.t.\ (r,l)\in AC_R(p))\\ \dfrac{|i_X(i_2)|}{\max(|i_X(i_1)|,|i_X(i_2)|)} & if\ (x,Count)\in AC_X(p)\\ \dfrac{|i_X(i_1)\cap i_X(i_2)|}{|i_X(i_1)|} & if\ (x,Inter)\in AC_X(p)\\ \dfrac{\max_{f\in F}\sum_{v\in i_A(i_1)}\overline{Sim_T}^a(v,f(v))}{\min(|i_A(i_1)|,|i_A(i_2)|)}*(1-\max(0,\dfrac{|i_A(i_1)|-|i_A(i_2)|}{|i_A(i_1)|}) & if\ (x=a)\wedge(a,Simil)\in AC_A(p)\\ \dfrac{\max_{f\in F}\sum_{i\in i_R(i_1);}\overline{Sim_I}^{pNew}(i,f(i))}{\min(|i_R(i_1)|,|i_R(i_2)|)}*(1-\max(0,\dfrac{|i_R(i_1)|-|i_R(i_2)|}{|i_R(i_1)|}) & if\ (x=r)\wedge(r,Simil)\in AC_R(p)\\ & pNew=p\cdot s,\ s\in S_R^1, s(1)=r\end{cases}$$

The above formulas are designed to be asymmetric. Asymmetry is used to ensure that considering the relations and attributes selected by the application context, if an instance i_1 has at least the same attribute and relation values of i_2 then the extensional similarity between i_2 and i_1 is equal to one.

The method compute $\overline{\mathrm{Sim}}_x^p$ selecting one of the above formulas according to the definition of *AC*: if AC returns a relation or attribute having as operation *Count* the third formula is adopted, as operation *Inter* the fourth is considered, and so on. The fifth formula is adopted whenever the AC returns an attribute whose operation is

Simil. In this case, the comparison of attribute values rely on $\overline{Sim}_T^a$ which defines the similarity for values of the attribute a having data type T. $\overline{Sim}_T^a$ is provided by the data layer as suggested by [1]. The set of partial functions in F are employed to represent the possible matching among set of values when instances have relations or attributes with multiple values. For example, the instance IE_1 has IF_3 and IF_2 related via $\underline{C_1}$, ID_1 has IF_3, when IE_1 and ID_1 are compared two possible partial and bijective functions f_1 and f_2 can be considered between the instances related to IE_1 and ID_1: f_1:$IF_2 \rightarrow IF_3$ and f_2:$IF_3 \rightarrow IF_3$.

It is important to note that each time the similarity is assessed in terms of related instances (whenever $(r,Simil) \in AC_R(p)$) the relation r that is followed to reach the related instances is added to the path of recursion. Thus during the recursive assessment the AC is always worked out on the most updated path of recursion.

5 Experiment and Evaluation

The similarity assessment among the research staff working at the Institute (CNR-IMATI-GE) is considered as application case to evaluate the proposed method. Two experiments are performed considering the contexts "Exp", "Int" mentioned in paragraph 4.1. Eighteen members of the research staff are considered; the information related to their projects, journal publications and research interests are inserted as instances in the ontology depicted in Fig. 1 according to what is published at the IMATI web site[4]. The ontology is expressed in OWL paying attention to adopt only the language constructs that allow to remain within the ontology model considered in definition 1. The resulting ontology is available at the web site [4]. Our method is implemented in JAVA and is tested on this ontology.

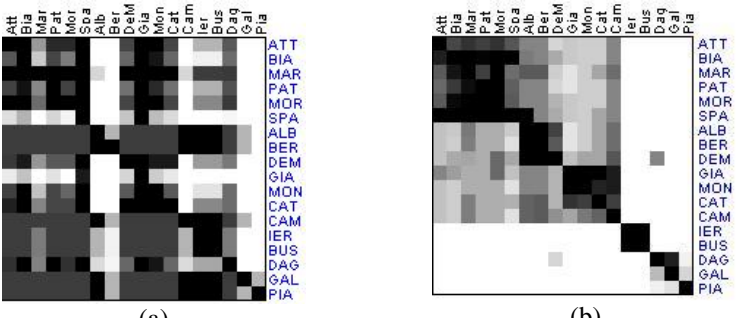


Fig. 3. (a) Similarity matrix for context "Exp"; (b) Similarity matrix for context "Int"

Using the formalization of the two application contexts AC_{Int} e AC_{Exp} previously defined (formulas (3), (4)) we have computed the similarity through the proposed framework. The results are represented by the similarity matrices in Fig. 3: (a) is the

[4] http://www.ge.imati.cnr.it

result related the context "Exp" and (b) is the result related to the context "Int". Each column j and each row i of the matrix represent a member of the research staff (identified by the first three letter of his name). The grey level of the pixel (i,j) represents the similarity value (Sim(i,j)) between the two members located at the row i and columns j: the darker is the colour the more similar are the two researchers.

Analysing the similarity matrices it is easy to realize that they are asymmetric; this confirms that the proposed model assesses an asymmetric similarity. Comparing the two matrices, it stands out how they are different: it is evident that the two contexts induce completely different similarity values. For example, "Dag" results very similar to "Bia" with respect to their experience (black pixel in Fig. 3.a), but they are no similar with respect to their research interest (white pixel in Fig. 3.b). Moreover sim(Dag,Bia)>sim(Bia, Dag) in Fig. 3.a means that Bia has at least the experience of Dag and she/he can be considered similar to Dag (if somebody with the Dag experience is searched), but the inverse is not true.

Two kind of evaluations of the result concerning the similarity obtained with respect to the research interest (Fig. 3.b) are performed.

The first evaluation is based on the concept of recall and precision calculated considering the same adaptation of recall and precision made by [5]. More precisely, considering an entity x the recall and precision are defined respectively as $(A \cap B)/A$, $(A \cap B)/B$ where A is the set of entities expected to be similar to x, and B is the set of similar entity calculated by a model. A critical issue in the similarity evaluation is to have a ground truth with respect to comparing the results obtained. We face this problem referring to the research staff of our institute and considering "similar" two members of the same research group. At the fact at IMATI researchers and fellows are grouped in three main research groups and one of those is composed by further three sub-groups. Then we consider the research staff as split in five groups. For each member i, A is the set of members of his research group while B is composed by the first n members retrieved by the model. For each group we have calculated recall and precision considering as "n" the smaller number of member needed to obtain a recall of 100%, and then we have evaluated the precision. The average recall is estimated equal to 100% with a precision of 95%. These results are quite encouraging: the recall equal to 100% demonstrates that for each research group the similarity is able to rank all the expected members while the precision equal to 95% means that the average number of outsiders to be considered to rank all group members is equal to 5%.

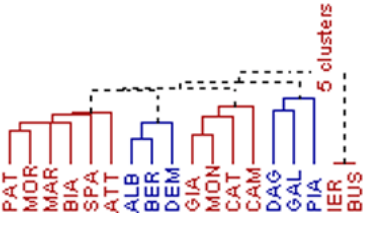


Fig. 4. The dendrogram obtained through the hierarchical gene clustering

We have performed a second evaluation according to the context "Int" using a data mining application. For each researcher and fellow we have computed his similarity with respect to the other members applying our method. In this way, we associate to each research staff member a string of values which correspond to his relative distance from the other members. The strings correspond to the rows of the similarity matrix (Fig. 3.b). Then we have applied a tool to perform the hierarchical clustering among genetic microarray [6] to the set of strings, considering each string as a kind of researcher genetic code. The dendrogram obtained is shown in Fig. 4, it recognizes the five clusters which resemble the research group structure of our institute.

6 Related Work

Semantic similarity is intended differently according to the application domain where it is adopted. Currently it is relevant in the ontology alignment [7,8], conceptual retrieval [9] as well as semantic web service discovery and matching [10,11] and it is expected to increase its relevance in framework as for the metadata analysis [12].

We discuss related works grouping them in different tracks according to their purpose and the ontology model they adopt.

Similarities in the Ontology alignment. There are plenty of methods to align ontology, as pointed out by Euzenat et al. [8]. The semantic similarity is adopted in this context to figure out relations among the entities in the ontology schemas. It is employed to compare the name of classes, attributes and relations, determining reasonable mapping between two distinct ontologies. On the contrary, the method proposed in this paper is specifically designed to assess similarity among instances belonging to the same ontology. Some similarities adopted for the ontology alignment consider quite expressive ontology language, (e.g., [7] focus on a subset of OWL Lite) but they mainly focus on the comparison of the structural aspects of ontology. Due to the different purpose of these methods, they result to be unsuitable to properly solve the similarity among instances.

Concepts similarity in lexicographic databases. Different approaches to assess semantics similarity among concepts represented by words within lexicographic databases are available. They mainly rely on edge counting-base [13] or information theory-based methods [14]. The edge counting-base method assumes terms which are subjects of the similarity assessment as edges of a tree-like taxonomy and defines the similarity in terms of the distance between edges [13]. The information theory-based method defines the similarity of two concepts in terms of the maximum information content of the concept which subsumes them [15,16]. Recently new hybrid approaches have been proposed: Rodriguez and Egenhofer [3] takes advantage from the above methods and adds the idea of features matching introduced by Tversky [17]. Schwering [9] proposes a hybrid approach to assess similarity among concepts belonging to a semantic net. The similarity in this case is assessed comparing properties of concept as feature [17] or as geometric space [18]. With respect to the method presented in this paper Rada et al. [13], Resnik [15], Lin [16] work on lexicographic databases where the instances are not considered. If they are adopted as they were originally defined to evaluate the instances similarity they are doomed to fail since they ignore important information provided by instances, attributes and

relations. Moreover, Rodriguez and Egenhofer [3] and Schwering [9] use the features or even conceptual spaces, information that are not native in the ontology design and should be manually added. Instead the method proposed here aims at addressing as much as possible the similarity taking advantage from the information that have been already spread in the ontology. Additional information are considered only to perform a tuning of the similarity with respect to different application context.

Similarities which rely on ontology models having instances. Other works define similarity relying on ontology models closer to those adopted in the semantic web standards. On the one hand, Hau et al. [11] identifies similar services measuring the similarity between their descriptions. To define a similarity measure on semantic services it explicitly refers to the ontology model of OWL Lite and defines the similarity among OWL objects (classes as well as instances) in terms of the number of common RDF statements that characterize the objects. On the other hand, Maedche and Zacharias [2] adopts a semantic similarity measure to cluster ontology based metadata. The ontology model adopted in this similarity refers also to IS-A hierarchy, attributes, relations and instances. Even if these methods consider ontology models which are more evolved than the taxonomy or terminological ontology, their design ignores the need to tailor the semantic similarity according to specific application contexts. Thus to assess the similarity experimented in this paper, two distinct ontologies need to be defined instead of simply defining two contexts as we do.

Contextual dependent similarity. Some papers combine the context and the similarity. Kashyap and Sheth [19] use the concept of semantic proximity and context to achieve the interoperability among different databases. The context represents the information useful to determine the semantic relationships between entities belonging to different databases. However they do not define a semantic similarity in the sense we are addressing and the similarity is classified in some discrete value (Semantic Equivalence, Semantic Relevance, Semantic Resemblance, etc). Rodriguez and Egenhofer [3] integrate the contextual information into the similarity model. They define as application domain the set of classes that are subject to the user's interest. As in our proposal, they aim to make the similarity assessment parametric with respect to the considered context. Moreover, differently from our methods they formalise the context rather then the similarity criteria induced by the context.

The discussion of the related works shows that beside semantic similarity is defined from different parties, these definition are far from provide a complete framework as intended in our work: they often have different purposes, they consider simpler ontology model, or they completely ignore the need of tailoring the similarity assessment with respect to specific application context. Of course, some of the mentioned works have been particularly precious in the definition of our proposal. As already mentioned during the presentation of the paper both Maedche and Zacharias [2] and Rodriguez and Egenhofer [3] have strongly inspired the part related to the structural similarity. However, to successfully support our purposes the class slots have been considered as distinguishing features. Furthermore, the methods proposed by Maedche and Zacharias [2] for the class matching defines a similarity which is symmetric, thus we have adapted the original in order to make it asymmetric.

7 Conclusions and Future Work

The paper proposes a framework to assess the semantic similarity among instances within an ontology. It combines and extends different existing similarity methods taking into account as much as possible the hints encoded in the ontology and considering the application context. A formalization of the criteria induced by the application is provided as a mean to parameterise the similarity assessment and to formulate a measurement more sensible to the specific application needs.

The framework is expected to bring a great benefit in the analysis of the ontology driven metadata repository. It provides a flexible solution to tailor the similarity assessments according with the different applications: the same ontology can be employed in different similarity assessments simply defining distinct criteria, and there is no more need of building a different ontology for each similarity assessment.

Nevertheless some research and development issues are still open. For example in the proposed approach the formalization of application context affects only the similarity defined by the extensional comparison. It could be interesting to deepen if the context results also in the external comparison similarity. Moreover, it would be worth to extend the similarity to ontology model towards OWL and to test it in more complex use cases.

Acknowledgements

This research started within the EU founded INVISIP project and then has been partially performed within the Network of Excellence AIM@SHAPE.

References

1. Ehrig, M., Haase, P., Stojanovic, N., and Hefke, M.: Similarity for Ontologies - A Comprehensive Framework. ECIS 2005. Regensburg, Germany (2005)
2. Maedche, A. and Zacharias, V.: Clustering Ontology Based Metadata in the Semantic Web. PKDD 2002 LNAI Springer-Verlag (2002) 348-360
3. Rodriguez, M. A. and Egenhofer, M. J.: Comparing geospatial entity classes: an asymmetric and context-dependent similarity measure. IJGIS. Vol. 18[3]. (2004) 229-256
4. Test Ontology, http://www.ge.imati.cnr.it/ima/personal/albertoni/odbase06p.owl.
5. Rodriguez, M. A. and Egenhofer, M. J.: Determining semantic similarity among entity classes from different ontologies. IEEE Trans.Knowl.Data Eng. Vol. 15[2]. (2003) 442-456
6. Hierarchical Clustering Explorer, 3.0, http://www.cs.umd.edu/hcil/multi-cluster/.
7. Euzenat, J. and Valtchev, P.: Similarity-Based Ontology Alignment in OWL-Lite. ECAI. Valencia, Spain IOS Press (2004) 333-337
8. Euzenat, J., Le Bach, T., and et al.: State of the Art on Ontology Alignment. (2004) http://www.starlab.vub.ac.be/research/projects/knowledgeweb/kweb-223.pdf
9. Schwering, A.: Hybrid Model for Semantic Similarity Measurement. OTM Conferences. LNCS Vol. 3761 Springer-Verlag (2005) 1449-1465
10. Usanavasin, S., Takada, S., and Doi, N.: Semantic Web Services Discovery in Multi-ontology Environment. OTM Workshop 2005 LNCS Vol. 3762 Springer-Verlag (2005) 59-68

11. Hau, J., Lee, W., and Darlington, J.: A Semantic Similarity Measure for Semantic Web Services. Web Service Semantics:Towards Dynamic Business Integration, workshop at WWW 05. (2005)
12. Albertoni, R., Bertone, A., and De Martino, M.: Semantic Analysis of Categorical Metadata to Search for Geographic Information. Proceedings 16th International Workshop on Database and Expert Systems Applications, 2005. IEEE (2005) 453-457
13. Rada, R., Mili, H., Bicknell, E., and Blettner, M.: Development and application of a metric on semantic nets. IEEE Trans.Syst.Man Cybern. Vol. 19[1]. (1989) 17-30
14. Li, Yuhua, Bandar, Zuhair, and McLean, David: An Approach for Measuring Semantic Similarity between Words Using Multiple Information Sources. IEEE Trans.Knowl.Data Eng. Vol. 15(2003) 871-882
15. Resnik, P.: Using Information Content to Evaluate Semantic Similarity in a Taxonomy. Proc. of the Fourteenth Int. Joint Conference on Artificial Intelligence (1995) 448-453
16. Lin, D.: An Information-Theoretic Definition of Similarity, Proc. of the Fifteenth Int. Conference on Machine Learning. Morgan Kaufmann (1998) 296-304
17. Tversky, Amos: Features of similarity. Psychological Review. Vol. 84[4]. (1977) 327-352
18. Gädenfors, P.: How to make the semantic web more semantic.FOIS.IOS Press (2004) 17-34
19. Kashyap, Vipul and Sheth, Amit: Semantic and schematic similarities between database objects: a context-based approach. VLDB J. Vol. 5[4]. (1996) 276-304

Finding Similar Objects Using a Taxonomy: A Pragmatic Approach

Peter Schwarz[1], Yu Deng[2], and Julia E. Rice[1]

[1] IBM Almaden Research Center
San Jose, CA 95120
{schwarz, julia}@almaden.ibm.com
[2] IBM Thomas J. Watson Research Center
Yorktown Heights, NY 10598
dengy@us.ibm.com

Abstract. Several authors have suggested similarity measures for objects labeled with terms from a hierarchical taxonomy. We generalize this idea with a definition of information-theoretic similarity for taxonomies that are structured as directed acyclic graphs from which multiple terms may be used to describe an object. We discuss how our definition should be adapted in the presence of ambiguity, and introduce new similarity measures based on our definitions.

We present an implementation of our measures that is integrated with a relational database and scales to large taxonomies and datasets. We evaluate our measures by applying them to an object-matching problem from bioinformatics, and show that, for this task, our new measures outperform those reported in the literature. We also verified the scalability of our approach by applying it to patent similarity search, using patents classified with terms from the taxonomy defined by the United States Patent and Trademark Office.

Keywords: Semantic similarity measures, Object matching, Taxonomy, Information theoretic similarity.

1 Introduction

Taxonomies have long been recognized as a useful tool for classification. In addition to providing a precise way to name classes of individuals that share certain properties or behavior, they also provide a means of determining how similar one such individual is to another. In its simplest form, a taxonomy defines a hierarchical grouping of individuals into ever more specific classes. Two individuals share the properties of the most specific grouping that includes both of them, and the degree to which the two individuals are similar depends on the location of this class in the hierarchy. The lower in the hierarchy, the more similar the individuals are. Various authors[8], [10], [5] have defined ways of turning this intuitive idea of similarity into a numeric value that can be used to rank the similarity of objects.

The ability to find similar objects given a description of a target is useful in many domains. For example, one may wish to find patents similar to a given patent, or subjects similar to a hypothetical "ideal" subject for a clinical trial. In bioinformatics, one may

R. Meersman, Z. Tari et al. (Eds.): OTM 2006, LNCS 4275, pp. 1039–1057, 2006.

wish to find gene products (e.g. proteins) similar to a given gene product. In each of these domains, and others, comprehensive taxonomies have been defined and used by various organizations to classify sets of objects. Examples include the Gene Ontology (GO)[1], Medical Subject Headings (MESH) [1] and the patent classification taxonomies of the United States Patent and Trademark Office (USPTO)[2] and the World Intellectual Property Organization (WIPO) [3].

Classification using such taxonomies is more complex than the simple example described above. Firstly, it is frequently the case that a class of individuals may specialize the properties of more than one parent class. Furthermore, taxonomies often evolve, as new specialized groupings are formed and older ones are reorganized. Even with an unchanging taxonomy, the classification of a particular object may evolve as more is learned about it, or users of the taxonomy may disagree as to how it should be classified. Lastly, real taxonomies tend to be quite large, and the sets of objects they are used to classify are often very large. Thus, any approach to finding similar objects must scale well in both these dimensions.

In this paper, we present a pragmatic approach to the use of taxonomies to find similar objects. Given a set of objects, we assume that each object has been labeled with one or more terms from a taxonomy structured as a directed acyclic graph. A similarity measure allows one to select any specific object from the set (the *target* object) and order the remaining objects (the *candidate* objects) by how similar they are to the one selected. We define two such similarity measures in this paper. The first, *holistic similarity*, is a new information-theoretic similarity measure that is more general and well-founded than prior art. The second, *generic holistic similarity*, adapts our holistic similarity measure to cope with ambiguity, as introduced by an evolving taxonomy or classifiers with imperfect knowledge. To our knowledge, this is the first attempt to consider this latter problem. We also describe a scalable implementation of our measures that is tightly integrated with an object-relational database, and we evaluate our approach by applying it to an object-matching problem from bioinformatics for which the correct answers are known *a priori*. The results show that our new measures are more successful than those previously reported, and that our implementation scales well for large taxonomies and object sets. We further evaluated the implementation by testing its ability to search a large set of patents classified with the USPTO's classification taxonomy for patents similar to a designated target. The results demonstrate the scalability of our implementation, and the importance of finding good candidate-selection heuristics when the search space is large.

The remainder of this paper is organized as follows. To open Section 2, we review the information-theoretic definition of similarity. Next, in Section 2.1, we define our holistic similarity measure. In Section 2.2, we consider how classifiers that have incomplete knowledge of the objects being classified introduce ambiguity, and define the generic holistic similarity measure for such situations. Section 3 summarizes related work. In Section 4 we describe our implementation, and Section 5 compares the efficacy of our

[1] http://www.nlm.nih.gov/pubs/factsheets/mesh.html
[2] http://www.uspto.gov/web/patents/classification/
[3] http://www.wipo.int/classifications

new similarity measures to a variety of others. Section 6 summarizes our contributions, and suggests some areas for future work.

2 Information-Theoretic Similarity

The idea of using an information-theoretic definition of similarity to compare objects labeled using a taxonomy was introduced by Resnik[8]. Given a taxonomy defined over a set of terms T, Resnik defined the similarity of two terms $t_1 \in T$ and $t_2 \in T$ with respect to a corpus of objects $\mathcal{O}$ as $sim_{Resnik} = \max_{\hat{t} \in S(t_1,t_2)}[-\log p(\hat{t})]$, where $S(t_1, t_2)$ is the set of terms in T that subsume t_1 and t_2 and $p(t)$ is the probability that an object randomly chosen from $\mathcal{O}$ represents an occurrence of term t. An object represents an occurrence of a term t if it is labeled with t or a descendant of t.

The quantity $I(t) = -\log p(t)$ is known as the *information content* of term t. If the taxonomy is a hierarchy (i.e a tree), the term $\hat{t}$ satisfying Resnik's definition will be the least common ancestor of t_1 and t_2.

Lin[5] derives an axiomatic definition for similarity based on a limited set of assumptions that can be summarized as follows:

1. The maximum similarity between two (identical) objects is 1 and the minimum similarity is 0.
2. The similarity between two objects is a function of their commonality and their differences.
3. If each object can be described from several perspectives, the overall similarity of two objects is a weighted average of their similarities as seen from the individual perspectives.[4]

Under these assumptions, and additionally assuming that the target and candidate objects are selected independently, the similarity between a target object T and a candidate object C is[5]:

$$sim(T, C) = \frac{I(common(T, C))}{I(descr(T)) + I(descr(C))} \tag{1}$$

where $common(T, C)$ represents a description of what the two objects have in common, and $descr(T)$ and $descr(C)$ represent descriptions of the two objects individually.

If T and C are each labeled by a single term from a hierarchical taxonomy, their commonality is represented by the term that is the least common ancestor a of the terms t and c that were used to describe T and C, respectively. With respect to a particular corpus of objects $\mathcal{O}$, the similarity becomes[5]:

$$sim_{Lin}(T, C) = \frac{2I(a)}{I(t) + I(c)} = \frac{-2\log p(a)}{-\log p(t) - \log p(c)} \tag{2}$$

[4] While not strictly necessary, this assumption is intuitive and makes the similarity function that satisfies the assumptions unique. We follow Lin's lead and adopt it.

2.1 Holistic Similarity

Equation 2 intuitively captures the idea of similarity in the most straightforward case, in which the taxonomy is structured as a tree and each object is labeled by a single term. As noted in Section 1, however, in practice many taxonomies are not trees, but allow a new term to be derived from multiple parents. Furthermore, classification systems often allow an object to be labeled using multiple terms. In this section, we examine how to define similarity under these more general conditions. We begin by defining a taxonomy as a directed acyclic graph.

Definition 1 (Taxonomy). *A taxonomy $\mathcal{T}$ is a directed acyclic graph (N, E, r), where N is a set of nodes, $E \subseteq N \times N$ is a set of directed edges and $r \in N$ is a unique root node of $\mathcal{T}$ from which every other node is reachable.*

Each node $n \in N$ represents a term of $\mathcal{T}$, and each directed edge $e \in E$ connects a more general parent term to a more specific child term. Typically, the relationship between parent and child terms can be described as an *is-a* or *kind-of* relationship, but our measures can also be used with other transitive relationships, such as *part-of*, *works-for*, *belongs-to*, etc., as long as one wishes to treat objects closely related in these ways as similar.

In accordance with standard graph terminology, the in-neighbor region of a given node refers to all nodes from which the given node is reachable. We will use the term *in-neighbor graph* to refer to both the nodes of the in-neighbor region and the edges that connect them. For convenience, we will refer to nodes in the in-neighbor region of a given node (i.e all nodes from which the given node is reachable) as "ancestors" of the given node, and similarly use the term "descendants" to refer to nodes in the given node's out-neighbor region. We will also find it useful to describe portions of the taxonomy as subgraphs of $\mathcal{T}$. We will use the notation $Terms(g)$ to refer to the set of nodes in a subgraph g.

A *label* is a subset of the terms in the taxonomy. A label can be used to represent the classification of a specific object, but it can also be used to represent more general concepts, like what two objects have in common.

Definition 2 (Label). *Given a taxonomy $\mathcal{T} = (N, E, r)$, a label is a nonempty set of terms $L \subseteq N$.*

If a term in a taxonomy applies to an object, so do the terms that correspond to each of its ancestors. Therefore, some of the terms in a label may be redundant. Although, as will be seen, such redundant terms do not affect similarity as calculated by our measures, their presence complicates the descriptions of objects, and obscures the equivalence of labels. We therefore define the concept of a *minimal label* to eliminate such terms.

Definition 3 (Minimal Label). *A label L is a minimal label if for every term $l \in L$, no ancestor of l is also in L.*

Given a label L, one can derive a unique minimal label L' by removing from L every term that is an ancestor of another term in L. Let $Lmin(L)$ denote the minimal label derived from L.

A *labeling* assigns a label to each object in some corpus.

Definition 4 (Labeling). *Given a taxonomy $\mathcal{T}$ and a corpus $\mathcal{O}$, a labeling is a total function $L : \mathcal{O} \mapsto 2^N$. If o is an object in $\mathcal{O}$, let $L(o)$ denote the label for object o.*

A label can also be associated with any subgraph of the taxonomy. The label for a subgraph g is the set of nodes contained in the subgraph, i.e $Terms(g)$. For convenience, however, we will generally refer to labels with the notation L_X, where X may be either an object or a graph.

A labeling is a *minimal labeling* if for all $o \in \mathcal{O}$, $L(o)$ is a minimal label. For the remainder of this paper we will assume that all labelings are minimal labelings unless otherwise noted.

We interpret the labeling of an object with one or more terms to imply only that *at least* those terms (and their ancestors) apply to the object. We adopt this open-world model because we believe it more accurately reflects how taxonomies are used in many domains. Frequently, both the label of an object and the structure of the taxonomy itself change as knowledge accumulates concerning the domain and the objects of interest. At any point in time, the label of an object only reflects what has been discovered about it so far. Furthermore, when the taxonomy is structured as a DAG, the closed-world assumption is incompatible with the use of interior terms in labels. We defer a discussion of this latter point to Section 2.2.

It is also useful to be able to enumerate all the terms associated with a label, either directly or indirectly. We therfore formalize the concept of an *ancestor graph*.

Definition 5 (Ancestor Graph). *Let L be a label. For each term $l \in L$, let $in(l)$ be the in-neighbor graph of l in $\mathcal{T}$. The ancestor graph of L is the union of the in-neighbor graphs of the terms contained in L. That is: $Anc(L) = \cup_{l \in L} in(l)$.*

Given an object o with label L_o, we will use the notation $Anc(o)$ to refer to the ancestor graph of L_o. The set of nodes $Terms(Anc(o))$ represent an exhaustive list of the terms associated with o.

We now associate a probability with an arbitrary label. Its value is the probability of finding an object to which *at least* those terms in the label L apply. We refer to this as the *inclusion probability*, $p_i(L)$.

Definition 6 (Inclusion Probability). *Let L be a label. Then $p_i(L)$ is the probability that the ancestor graph of the label of an object chosen at random from $\mathcal{O}$ contains L. That is, given a randomly chosen object o: $p_i(L) = p(L \subseteq Terms(Anc(o)))$.*

If the ancestor graph of an object's label contains L, it also contains $Lmin(L)$ and vice versa. Hence, if L is not minimal, the extra terms do not affect its inclusion probability.

Inclusion probability gives us the tool we need to apply Lin's general definition of similarity from Equation 1 to a taxonomy. To quantify the individual information content of the objects being compared (the denominator of Equation 1), we use the inclusion probability of their labels. To quantify the information content of the commonality between the two objects (the numerator of Equation 1), we find a label L_A to represent that commonality, and use the corresponding inclusion probability. The label L_A is constructed by intersecting the ancestor graphs of the labels of the objects being compared. We refer to the resulting measure as *holistic similarity* because it treats all the terms in

a label as a group. As we will see in Section 3, other measures that have been suggested for use when objects are labeled with multiple terms treat each term individually.

Definition 7 (Holistic Similarity). *Let L_T and L_C be the labels of objects T and C, respectively, and let $L_A = Lmin(\ Terms(Anc(L_T) \cap Anc(L_C)))$. Then:*

$$sim_H(T,C) = \frac{-2\log p_i(L_A)}{-\log p_i(L_T) - \log p_i(L_C)} = \frac{2I(L_A)}{I(L_T) + I(L_C)}$$

When the taxonomy is a tree and each object is labeled with a single term, Definition 7 reduces to Equation 2.

As defined by Equation 2, Lin's formulation cannot be used when the taxonomy is a DAG, because the terms describing the objects do not necessarily have a unique least common ancestor. Nor is it applicable when objects are labeled with multiple terms. Resnik's similarity measure, which is defined in terms of the ancestor with the maximal information content, could be used with a DAG taxonomy, but not when multiple terms are used as labels. However, because it is specified over labels, not terms, holistic similarity is well-defined in both of these cases.

2.2 Generic Similarity

In this section, we consider more carefully the use of the interior terms of a taxonomy in labels. We begin by examining the meaning of using such a term in a label.

When an interior term is used in a label, there are two possible interpretations. The term may have been selected because no more specific term in the taxonomy applies to the object in question. On the other hand, the individual doing the labeling may choose an interior term because he or she does not know which (if any) more specific term applies. We refer to the first type of labeling as *careful* and to the second as *generic*. Both often occur in practice, in particular as a taxonomy evolves. Initially, a single term may be applied to what later turns out to be a whole group of distinct subclasses of objects. Over time, as these subclasses are recognized, new more-specific terms are created. However, objects classified under one version of the taxonomy are not necessarily reclassified whenever the taxonomy evolves.

To accommodate the careful use of interior terms as labels, we augment the taxonomy by adding a new descendant term, X^*, for every interior term X used as a label. We refer to X^* as an *anonymous term*, because it describes an unnamed subset of the objects to which the term X applies. If an object can be labeled with multiple terms, it may also be necessary to introduce an anonymous term for combinations of terms that are used carefully. As far as our definition of similarity is concerned, anonymous terms behave exactly like real terms in T and we will not consider them further in this paper.

Unlike careful labeling with interior terms, generic labeling forces us to rethink our basic understanding of similarity by introducing uncertainty into the labeling of objects. If a target object is labeled "fruit", and we are uncertain as to which specific kind of fruit it is, candidate objects labeled "apple", "pear", or 'fruit" all fulfill the only specific requirement posed by the labeling of the target object, that of being a fruit. However, if we apply Definition 7, only the candidate labeled "fruit" will receive a similarity score

of 1 with respect to the target. In effect, sim_H penalizes objects labeled "apple" or "pear" for being "too specific" when the target object is generic. Note that the situation changes when the roles of target and candidate are reversed.

To reflect the asymmetry introduced by generic labeling, we define a revised similarity measure, sim_G, such that $sim_G(T, C) = 1$ if and only if object C is *substitutable* for object T.

Definition 8 (Generic Holistic Similarity). *Let L_T and L_C be the labels of objects T and C, respectively, and let $L_A = Lmin(Terms(Anc(L_T) \cap Anc(L_C)))$. Then:*

$$sim_G(T, C) = \frac{-2 \log p_i(L_A)}{-\log p_i(L_T) - \log p_i(L_A)} = \frac{2I(L_A)}{I(L_T) + I(L_A)}$$

The generic similarity measure views the candidate object, C, as an instance of the most specific class of objects that includes both the target and the candidate.

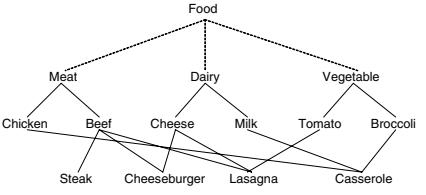


Fig. 1. Labeling With Generic Terms

Generic labeling reinforces our choice of an open-world model, because when the taxonomy is structured as a DAG, the closed-world model is incompatible with the use of generic terms as labels. To see why, consider the taxonomy of Figure 1. If an object is labeled with the generic term $\{Beef\}$, under generic labeleing we interpret that to mean the labeler is uncertain as to which of the terms describing a more specific type of beef applies. The object could be a steak, which is "just" beef, or it could be a cheeseburger. Because a cheeseburger contains additional ingredients, the ancestor graph of the label $\{Cheeseburger\}$ contains terms like $\{Cheese\}$ that are neither descendants nor ancestors of the term $\{Beef\}$, and these terms may also apply to the object in question. Thus, labeling a cheeseburger $\{Beef\}$ would violate the closed-world assumption, which states that terms not in the ancestor graph of $\{Beef\}$ do not apply to the object.

We conclude this section with a set of examples that demonstrates how sim_G orders candidate objects with respect to a fixed target. The examples are based on the taxonomy of Figure 1, and are intended to be illustrative rather than to provide an exhaustive case analysis.

Table 1 shows the similarity of five candidate objects to a common target object T labeled $\{Beef, Cheese\}$. The first candidate, labeled $\{Cheeseburger\}$, is a specialization of the target object and therefore substitutable for it. It receives a similarity score of 1 because its ancestor graph includes the complete ancestor graph of the target. The same is true of c_2, the candidate labeled $\{Beef, Cheese, Tomato\}$. The ancestor graph of c_3, labeled $\{Steak\}$, includes the term "Beef" and all its ancestors, but only

Table 1. Similarity to Target Labeled $\{Beef, Cheese\}$

C	L_C	L_A	$sim_G(T, C)$
c_1	$\{Cheeseburger\}$	$\{Beef, Cheese\}$	$\frac{2I(Beef,Cheese)}{I(Beef,Cheese)+I(Beef,Cheese)} = 1$
c_2	$\{Beef, Cheese, Tomato\}$	$\{Beef, Cheese\}$	$\frac{2I(Beef,Cheese)}{I(Beef,Cheese)+I(Beef,Cheese)} = 1$
c_3	$\{Steak\}$	$\{Beef\}$	$\frac{2I(Beef)}{I(Beef,Cheese)+I(Beef)}$
c_4	$\{Steak, Milk\}$	$\{Beef, Dairy\}$	$\frac{2I(Beef,Dairy)}{I(Beef,Cheese)+I(Beef,Dairy)}$
c_5	$\{Casserole\}$	$\{Meat, Dairy\}$	$\frac{2I(Meat,Dairy)}{I(Beef,Cheese)+I(Meat,Dairy)}$

intersects with the ancestors of "Cheese" at the root of the taxonomy ("Food"). L_A is therefore $\{Beef\}$, the minimal label of the intersection of the target and candidate ancestor graphs. Since $I(Beef)$ is calculated using inclusion probability, it reflects not only the number of objects labeled $\{Beef\}$, but also the populations of objects whose labels are descendants of $\{Beef\}$ (e.g. $\{Steak\}$, $\{Lasagna\}$) and those that include the term "Beef" among others in their label (e.g. $\{Beef, Broccoli\}$). c_4, the candidate labeled $\{Steak, Milk\}$, is similar to c_3, but in this case $L_A = \{Beef, Dairy\}$. The size of the population associated with this label is smaller than the one associated with $\{Beef\}$, since it includes only those objects whose label includes the term "Dairy" (or one of its descendants) as well as the term "Beef" (or one of its descendants). Hence the information content of this label is higher, and c_4 will receive a higher similarity score than c_3. For c_5, labeled $\{Casserole\}$, $L_A = \{Meat, Dairy\}$. Because inclusion probability associates more objects with this label than with the label $\{Beef, Dairy\}$, c_5 will receive a lower similarity score than c_4.

3 Related Work

The literature describes three distinct groups of similarity measures that can be applied to taxonomies. The first group of measures, which we will refer to as *term-similarity* measures, can be used to compute the similarity of two individual terms. The other two groups of measures can be used when an object is labeled with multiple terms.

Wu and Palmer[10] defined a term-similarity measure based on the depth in the taxonomy of the least common ancestor of two terms relative to the depths of the terms individually. The "closer" the common ancestor is to the terms themselves, the greater the similarity. Wu and Palmer did not describe how to use their approach when the taxonomy is a DAG. Another problem with this approach is that some portions of the taxonomy may have been extensively developed and contain many terms, whereas other areas are sparse. Such variations in the "density" of terms make this and other measures that rely on edge counts a poor estimate of similarity.

As noted in Section 2, the idea of using information content to measure similarity is due to Resnik[8]. Using the WordNet taxonomy and frequency estimates derived from

a large body of English text, Resnik calculated the semantic similarity of word pairs by selecting the common ancestor with the greatest information content. For words with multiple senses, Resnik used the sense that produced the maximum similarity. Using judgments made by human subjects as the standard, Resnik found that his measure worked better than earlier ones based on edge-counting. Although Resnik's measure can be used when the taxonomy is structured as a DAG, it cannot be used directly when objects are labeled with multiple terms, and it has a number of other disadvantages. Its range is not normalized to $[0, 1]$, but more importantly, by selecting the ancestor with the greatest information content it understates the similarity of objects by focusing on the single most significant aspect of their commonality, at the expense of all others.

Lin[5] provided an axiomatic definition of similarity, and showed how Resnik's approach could be adapted to fit this framework. Whereas Resnik's measure was based solely on the commonality between word meanings, Lin's also takes into account the differences in meaning to determine a normalized similarity score (see Equation 2). Lin compared his measure to Resnik's and to Wu and Palmer's, and found that it produced scores that were better correlated with human judgments than those produced by the other two measures. However, Lin does not describe how to use his measure when the taxonomy is a DAG, or when multiple terms are used to describe an object.

Maguitman et al.[7] generalize Lin's measure for taxonomies that are structured as a DAG, and even for certain more general graphs that contain cycles. They also allow different classes of edges to carry different weights for the purposes of determining similarity. However, unlike our holistic measures, their measure chooses the single common ancestor that maximizes similarity in order to compute the relative information content of the ancestor with respect to the candidate and target objects (i.e. their similarity). This is not the same as applying our formulation in Definition 7 to a pair of single terms, and does not follow directly from Lin's axiomatic definition of similarity in Equation 1 because, like Resnik's measure, it considers only a portion of the commonality between the terms. Maguitman et al. also do not generalize their approach to classification systems that use multi-term labels.

The similarity measures in the second group described in the literature measure the similarity of objects based on the number or frequency of terms that are common to the descriptions of both objects. Measures in this group include Jaccard, Dice and Set Cosine, which are used frequently in information retrieval systems and differ in how the count of common terms is normalized, as well as the FMS measure of Keller at al[4]. These measures do not take the structure of the taxonomy into account. Any candidate object that does not share terms with the target will receive a score of zero, even though it may be quite similar.

The third set of proposed similarity measures relies on an underlying term-similarity measure to determine the similarity between individual pairs of terms, and then combines these to yield an overall similarity score. Halkidi et al.[3] defined a similarity measure of this type for use in clustering web documents. Using the Wu and Palmer term similarity measure, they consider each term in the target and candidate sets individually, and find the most similar term from the other set. Then, over each term set (the target set and the candidate set), they average the similarity of these best matches. Finally, they combine the average similarity from the two sets with equal weight. Since

they use Wu and Palmer as the underlying term similarity measure, it is not clear how to apply this measure when the taxonomy is a DAG.

Wang et al.[9] developed a similarity measure that avoids this problem by using a generalized form of Lin's information-theoretic similarity measure to determine the similarity of each term-pair. If t and c are the target and candidate terms:

$$sim_{Lin*}(t,c) = \frac{2 * \max_{t_a \in Terms(Anc(c) \cap Anc(t))}(I(t_a))}{I(t) + I(c)}$$

Note that Wang et al. generalize Lin's formula from Equation 2 for use in a DAG taxonomy by selecting the least common ancestor with the maximum information content, and therefore their measure, like Resnik's and Maguitman et al's, is not holistic. Wang et al. also use a different function than Halkidi et al. for combining term-pair scores. Instead of averaging the scores of the best match from the other set for each term, they average the term similarity scores across all term pairs.

All measures in this third group draw an arbitrary distinction between combinations of terms for which a new term has been coined and those for which one has not. Furthermore, although the term similarities can be combined in various ways, none of these follows directly from Lin's axiomatic assumptions.

Keller et al[4] present several ways of quantifying similarity using fuzzy measures based on the depth or information content of terms. However, their measures either require subjectively-specified weights, or the solution of a high-order polynomial equation for each pair of target and candidate objects. We believe this to be prohibitively expensive for large problems.

Our holistic measures sim_H and sim_G do not belong to any of these groups. Unlike the term similarity measures, they can be used when labels contain multiple terms. Unlike the common term measures, they take distinct but similar terms into account. Unlike the pairwise measures, they do not consider individual terms, but rather take all the terms in each label into account simultaneously.

4 Implementation

In implementing the similarity measures described in Section 2, one of our key goals was to accommodate both large taxonomies and large corpora of objects. We therefore built our implementation in the context of an object-relational database management system, specifically IBM DB2 Universal Database V8.2.[5] We believe this approach offers a number of advantages. In the first place, storing the corpus in a database allows the full power of SQL to be used to select those objects of interest in a particular situation. For example, a user searching for similar objects in a large database of gene products may wish to restrict the search to human gene products. Secondly, the ability to extend the database management system with user-defined types and functions allowed us to implement certain critical operations very efficiently within the database, without requiring large amounts of data to be retrieved for manipulation by an application.

In our implementation, each similarity measure is implemented as an SQL query against a set of relations with a fixed schema. The taxonomy is represented as a table

[5] http://www.ibm.com/software/data/db2/udb/

of (parent term id, child term id) pairs and the associations between terms and objects are represented as (term id, object id) pairs. Since the associations for the target and candidate objects may in general come from different sources, we use separate relations to represent them. These relations may be base tables, but are more likely views created over the native representation of the associations.

Additional tables are defined to store information that can be precomputed once and used repeatedly in subsequent evaluations of the similarity measure. We will provide more information on this auxiliary data in Section 4.2.

4.1 User-Defined Types

Within each query, certain critical operations are implemented as User Defined Functions (UDF's) that operate on User Defined Types (UDTs) represented in the database as Binary Large Objects (BLOBs). Two types of operations warranted such special treatment.

Firstly, the taxonomy is naturally represented as a directed graph, and key steps in the evaluation of sim_H, sim_G, and various other measures involve graph-theoretic operations, e.g. the determination of common ancestors between the labels of target and candidate objects. In principle, such operations could be coded in ordinary SQL, but such implementations are cumbersome and not as efficient as state-of-the-art procedural algorithms. Therefore, we made extensive use of a general-purpose graph library for DB2. The library allows graphs to be constructed efficiently from database data using a user-defined aggregate function. Once constructed, they can be stored in the database as BLOBs and manipulated by UDFs that implement a wide range of graph-theoretic operations. A full discussion of the graph library, which scales to very large graphs, is beyond the scope of this paper.

A second critical step is the determination of the inclusion probabilities of particular labels. To determine the inclusion probability of a label L, one must know its frequency, that is, the number of objects in the corpus to which all of the terms in L apply. Recall that a term t applies to an object o if and only if the ancestor graph of the object's label includes the term in question, i.e. if $t \in Terms(Anc(o))$.

In principle, one could precompute frequencies for each of the $2^{|N|}$ combinations of terms that can be used as a label. For taxonomies of realistic size, however, this approach is impractical. Instead, we build an inverted list for each term, identifying the objects to which the term applies, i.e. those objects whose labels contain the term or any of its descendants. Let $O(t)$ denote the list of objects for term t. The frequency of a label can then be determined by finding the size of the intersection of the inverted lists of its individual terms. The inclusion probability is therefore:

$$p_i(L) = \frac{|\cap_{t \in L} O(t)|}{|\mathcal{O}|}$$

Like the taxonomy graph, the inverted lists are implemented as a User Defined Type optimized to support the operations needed to compute label frequency: intersection and length. The inverted list UDT stores a list of object identifiers as a simple vector. Identifiers can be inserted in any order as the list is built (using a user-defined aggregate function), and the list is sorted once when insertion is complete. The intersection of two

lists can be computed with a single pass through both lists, and a user-defined aggregate function is provided to find the intersection of a set of lists.

4.2 Precomputation

Certain information used to find and rank candidate objects can be used repeatedly for different target objects, as long as neither the taxonomy nor the corpus changes. In this section we consider several situations in which we opted to precompute such values.

Whenever a target and candidate object are compared, we need to find the intersection of their respective ancestor graphs. Furthermore, at least in our experiments, the same candidates are evaluated for many different targets. We therefore precompute the ancestor graph for the label of each object in the corpus of candidate objects. The graph library can generate ancestor graphs quite quickly, so precomputation is practical even for corpora of large size. The space requirement is modest, because the ancestor graph for an object is typically a small fraction of the entire taxonomy. Details for our experimental scenario can be found in Section 5. Updates to the candidate corpus can be handled incrementally, but updates to the taxonomy may require a complete recomputation of these graphs.

Note that we only precompute the ancestor graphs for candidate object labels, not target object labels, since each target object is generally only referenced once. However, we also precompute the ancestor graph for each individual term in the taxonomy. These graphs make the dynamic computation of ancestor graphs for target object labels more efficient, and are reused many times since many targets refer to the same terms. They consume much less space than the candidate label ancestor graphs, because there are far fewer terms than labels, and each graph is smaller. The term ancestor graphs are not affected by updates to the corpus, but may need recomputation when the taxonomy changes.

The final set of precomputed objects is the inverted lists. For each term in the taxonomy, we build a list containing the identifiers of all objects in the corpus that contain the term, or a descendant of the term, in their label. As noted above, this allows us to find the inclusion probability of an arbitrary label by intersecting the lists corresponding to its terms. The size of these lists is proportional to the size of the corpus. See Section 5 for details for our experimental scenario. The lists can be updated incrementally as objects are added to the corpus, but may need to be recomputed if objects are deleted or the taxonomy changes.

5 Experimental Evaluation

Similarity measures are difficult to evaluate, because similarity of objects is ultimately subjective in nature and individuals may differ in how they rank candidate objects with respect to a particular target. Other authors have tried to work around this problem by comparing their rankings to those of a panel of human subjects [8], [5] or to an independent measure of similarity appropriate to their domain[6,9,4].

We took a different approach. In search of a result that could be more objectively quantified, we applied our similarity measures to the more difficult problem of matching. Suppose that two sources of associations are using the same taxonomy to assign

labels to objects. Given the label of a target object as supplied by one source of labels, the task is to find the same object in a corpus of labels supplied by another source. In our experimental framework, the two sources of associations use a common identifier for objects, so it is easy to verify that a match has been found.

Specifically, we use the taxonomy associated with the Gene Ontology (GO)[1]. This taxonomy contains about 17000 terms, and can be represented by a graph with as many nodes and about 22000 edges. The terms are divided into three independent facets, representing a gene product's cellular location(s), molecular function(s) and the biological process(es) in which it participates. Each facet is a directed acyclic graph, and in our experiments, we ignored the facets and considered all terms as a single label. The relationships between terms are characterized as either *is-a* or *part-of*. We also ignored this distinction, since both relationships possess the transitive behavior that underlies our understanding of classification. Finally, a small number of terms in the taxonomy are considered obsolete, and have been gathered together as children of a special node. We removed these nodes from the taxonomy, since their location in the taxonomy does not reflect their semantics.

Numerous organizations have used the GO taxonomy to annotate a large number of gene products. We restrict our attention to human gene products registered in the SwissProt databank, which number about 28000. Almost all of these gene products have annotations supplied by the UniProt Knowledgebase[2], and these gene products constitute our candidate objects. We use the approximately 110000 annotations for these objects as our corpus of term associations. We removed only those annotations that associate a gene product with an obsolete term.

Of the 28000 gene products in the corpus, 5789 also have annotations supplied by Proteome, Inc.[6] These gene products constitute our set of targets, and the corresponding set of about 20000 annotations constitute our target associations. The matching task is therefore: given the annotations for a gene product supplied by Proteome Inc., find the same gene product using the annotations supplied by UniProt.

We note that this is inherently a very difficult problem. The two organizations that generated the annotations we use for matching are completely independent. They did not necessarily have the same goals or use the same guidelines in assigning terms to gene products, nor did they necessarily work from the same source information. Furthermore, the space to search is large (28000 objects), and often contains many objects similar to the desired target.

To apply a similarity measure to this problem, we rank the candidate objects using the similarity measure, and then determine whether the matching object is found in the top K objects as ranked by the measure. We also keep track of the depth in the candidate list at which the matching object was found.

In practice, it is too expensive to compute the similarity score of a target with respect to all 28000 objects in the corpus. We therefore considered as candidates only those objects whose labels have at least one term in common with the label of the target object. We chose this heuristic because it could be applied to all the measures we wished to evaluate, some of which are based entirely on common terms and their properties. However, it should be noted that for many target objects, the labels supplied by the

[6] http://www.proteome.com

two organizations have no terms in common. Thus, regardless of the measure used, this heuristic limits the number of targets that can be successfully matched to a maximum of 3660 of 5789, or 63.2%. We considered other heuristics for determining the candidate set; see Section 5.4 for discussion.

5.1 Measures Evaluated

In addition to the holistic measures sim_H and sim_G, we applied various other similarity measures to the matching problem described above. In this section, we briefly describe each of these measures.

The simplest measure we studied was **Common Term Count**, which ranks candidates based solely on the number of terms they have in common with the target. Next, we evaluated the **Jaccard** measure, a normalized form of **Common Term Count** that is commonly used to measure similarity in IR tasks.[7]

We also considered two analogous measures that are based on common terms, but weight each common term by its information content. In the first measure, **Common Term IC**, we sum the information content of the common terms to determine similarity. In a normalized form of this measure, the common information content is compared to the total information content of all the terms in the two labels.

Next, we studied two measures that are not based solely on common terms, but also recognize that pairs of terms that are close in the taxonomy represent similar concepts. These measures calculate the pairwise similarity of individual terms from the target and candidate labels, and combine these to produce an overall score.

The first such measure we studied was proposed by Wang et al.[9]. As noted in Section 3, their measure uses a generalized form of Lin's similarity measure to determine the similarity of each term-pair. Given this definition of term similarity, their measure combines the pairwise similarities by averaging them across all pairs of terms in the two labels.

The second pair-set measure is a variant of one proposed by Halkidi et al.[3]. Halkidi et al. used the similarity measure defined by Wu and Palmer[10] to compute the similarity of term pairs, but it is unclear how to generalize this to a taxonomy structured as a DAG. Hence we instead used the same generalized form of Lin's measure as Wang et al.

As noted in Section 3, Halkidi et al.'s measure considers each term individually, and finds the most similar term from the other set. Then, the similarity of these best matches is averaged over the terms in each set. Finally, the average similarity from the two sets is combined with equal weight.

5.2 Results

We tested sim_H, sim_G and the six measures described above on targets from the set of 5789 gene products annotated by both UniProt and Proteome, Inc. Using the candidate-generation heuristic described above, we calculated the similarity score for each candidate with respect to the target. If the measure ranked the object corresponding to the target among the top 100 candidates, we considered it to have found a match. While this

[7] In addition, we tested the **Dice** and **Set Cosine** measures. Results were very close to those for the **Jaccard** measure.

may seem like a generous definition of success, we note again the inherent difficulty of our matching problem.

We calculated the success rate for each measure, along with the depth at which the matching object was found, averaged over the successful matches. The results are presented in Table 2.[8] Recall that the maximum possible success rate using the common-term heuristic for candidate selection is 63.2%.

Table 2. Comparison of Similarity Measures

Method	Success Rate (%)	Average Depth
sim_H	39.3	23.9
Common Term IC	39.1	22.2
sim_G	37.5	25.2
Normalized CT IC	36.9	25.4
Halkidi IC	36.1	26.2
Common Term Cnt.	35.2	28.4
Jaccard	30.9	28.7
Wang	24.5 ($\pm$ 2.42@95%)	35.9

We do not compare the measures on the basis of speed, since we made a concerted effort to reduce costs only for sim_H and sim_G. However, we note that our strategy of precomputation enabled us to test a single candidate in about 4.4ms using sim_G and about 6.4ms using sim_H. These values are for a 2.4GHz Pentium 4 with 1Gb of memory, running Windows XP and DB2 UDB v.8.2. The space requirements for the precomputed object ancestor graphs, term ancestor graphs and term inverted lists were modest: 122Mb, 40Mb and .5Mb, respectively.

5.3 Discussion

Although the differences in the success rates are not striking, some observations can be made. Firstly, the measures based solely on common terms did not adequately take into account pairs of terms that were closely related, but not identical. Because such pairs do not contribute at all to the similarity score, common-term measures that penalize candidates for terms that are not shared with the target understate similarity to a greater extent than those which do not. Thus, both **Common Term Count** and **Common Term IC**, which ignore unmatched terms, outperformed their normalized counterparts, **Jaccard** and **Normalized Common Term IC**, which incur a penalty for unmatched terms. For the holistic measures, all terms contribute to the similarity score, and indeed sim_H outperforms the other measures. However, our candidate-selection heuristic requires each candidate to have at least one term in common with the target. While this heuristic provides a level playing field for comparing the various measures, it limits the beneficial effect of taking related terms into account. We explore the effects of relaxing this restriction in Section 5.4.

[8] For one measure where we did not test the entire set of targets, a confidence interval is provided for the success rate.

Table 3. Patent Similarity Search

KNN	Patent ID	Score	Title
0	14019	1.0	'Apparatus and method for collecting flue gas particulate with high permeability filter bags'
0	47255	0.9673537	'Advanced hybrid particulate collector and method of operation'
0	195208	0.9673537	'Volatile materials treatment system'
0	265087	0.9673537	'Char for contaminant removal in resource recovery unit'
2	304641	0.9673537	'System and method for removing gas from a stream of a mixture of gas and particulate solids'
0	344644	0.9673537	'Electric dust collector and incinerator'
1	21467	0.9441908	'Thief process for the removal of mercury from flue gas'
0	25179	0.9441908	'Multi-stage particulate matter collector'
1	473644	0.9441908	'Method of regulating the flue gas temperature and voltage supply in an electrostatic precipitator...'

We also observe that measures that weight the importance of pairs of common terms based on their information content performed better than the analogous measures based on counting common terms. In particular, **Common Term IC** outperformed **Common Term Count**, and **Normalized Common Term IC** outperformed **Jaccard**.

The differences among the measures based on information content stem from differences in how similarity is derived from information content in each case. Both **Wang** and **Halkidi IC** normalize the common information content of each term-pair independently, and then combine these normalized values to reach an overall similarity value. As a result, term-pairs with relatively low common information content are given the same weight as pairs with much greater common information content. This is particularly so in the case of **Wang**, which pairs each term with every other, as opposed to **Halkidi IC**, which just pairs each term with its best match. However both measures suffer in comparison to those based on common term information content, which weight each common term in accordance with its information content. Another source of error in these measures is correlation. If a label contains two terms whose occurrence is correlated, these measures overestimate their information content when they occur together.

The holistic measures sim_H and sim_G avoid both these problems by calculating the combined information content of all the terms in each relevant set. This allows the similarity to be calculated exactly in accordance with the axiomatic definition from Equation 1, modified only as necessary to handle generic labeling (in the case of sim_G).

Lastly, although we believe that generic labeling with interior terms occurs pervasively in our corpus, we note that sim_H outperformed sim_G in our experiments. We believe this reflects our choice of a matching problem rather than one based strictly on similarity. The difference between the measures is that sim_G does not penalize a candidate for being labeled with terms that are more specific than those used to label the target. This may elevate the score of the matching candidate enough to make it competitive with others labeled with more general terms. However, it also elevates the

scores of candidates that use more specific terms than the matching candidate and its competitors, which has the opposite effect. If one views the terms associated with the target as requirements, these additional candidates satisfy the requirements as well as the matching candidate does, and are therefore equivalently similar. But in a matching problem, we are looking for a particular object, and the presence of the others makes finding it more difficult.

5.4 Candidate Selection

As we noted previously, all the measures we evaluated were limited by our candidate-selection heuristic. For the measures based on common terms, this limitation is absolute, since objects that have no terms in common with the target have a similarity score of zero. However, the other measures we evaluated have the potential to find additional matches if we consider additional candidates.

We tested this hypothesis with an alternative candidate-selection heuristic. For each association between a target object and a term, we augmented the set of terms associated with the target by adding associations between the target object and the original term's immediate neighbors. We then used as candidates all objects that have at least one term in common with this expanded set. We refer to this heuristic as *KNN-1*, because the candidate set comprises those objects that are described by at least one of the original terms or a neighboring term one hop away. The original common term heuristic could be described as KNN-0; KNN-2 and other values could be considered as well.

When we tested sim_H with KNN-1 on the matching problem, we found that it raised the success rate from 39.3% to 40.5%. While not a dramatic increase, this suggests that our measures can find similar objects even when they have no terms in common with the target. The downside is that the time required to match a target is proportional to the number of candidates considered, and increasing the value of N increased the number of candidates to test significantly. For KNN-0 on our dataset, the average number of candidates per target was 1429. For KNN-1 this rose to 3705, causing a greater than twofold increase in the average time required to match a target. For KNN-2, the average number of candidates rose to 6824.

We also tested the KNN candidate-selection heuristic on a second dataset. The United States Patent and Trademark Office (USPTO) maintains a classification scheme for patents based on a tree-structured taxonomy of about 160000 terms, referred to as classes and subclasses.[9] Individual patents are labeled with one or more terms from the taxonomy. We tested the holistic similarity measures on a corpus of approximately 500000 patents filed between 2001 and 2003, as classified by about 1.9 million associations between patents and terms.

While we could not perform an objective matching experiment with this dataset, it nevertheless provided a number of interesting opportunities for evaluating our measures and their implementation. Firstly, we were able to verify that our implementation could scale to a much larger corpus and a much larger taxonomy with satisfactory performance. The USPTO taxonomy also has a different structure than the GO taxonomy; it is broader and flatter, and does not have facets or multiple inheritance.

[9] http://www.uspto.gov/web/patents/classification/

Lastly, our candidate-selection heuristics generated fewer candidates per target than in the matching experiment. We were thus able to more easily observe the value of the KNN candidate-selection heuristic, coupled with our measures' ability to detect similarity without the presence of common terms between the candidate and target objects.

Table 3 shows the combined result of three similarity searches for a typical target, with KNN values of 0, 1 and 2. The first patent listed, ID 14019, is the target, and thus had an ideal similarity score of 1.0. Below, similar patents are listed with their similarity scores, as well as the KNN value of the search in which they were initially found. For example, patent 304641 was found by the search that used KNN-2 to select candidates, and scored as highly as any of the patents found with smaller KNN values. Similarly, two other highly-scored patents were found with KNN-1 that would not have been found with KNN-0, nor by any of the measures that rely solely on common terms.

6 Conclusions and Future Work

Similarity ranking of objects labeled using a taxonomy is an interesting problem with a variety of useful applications. This work has made several contributions to the state of the art. We developed new similarity measures that are applicable to real classification systems, in which the taxonomy can be structured as a DAG, objects can be labeled with multiple terms, internal terms can be used in labels, and different users may label the same object in different ways.

We implemented our measures using SQL and a pair of libraries for specialized data structures, realized as user-defined types. The result is a flexible, scalable implementation that is tightly integrated with a database management system and achieves good performance through strategic precomputation of key data structures.

We evaluated our measures on an object-matching task using the Gene Ontology, a taxonomy with all the properties noted above. Our measures were more successful at matching objects than those reported in the literature. We also tested our measures on a search task, using the patent classification taxonomy of the USPTO to find patents similar to a specified target. We evaluated two heuristics for candidate selection, a critical issue for large data sets.

A number of areas suggest themselves for future study. Most important is to apply these measures to more datasets and domains, to determine whether the results are comparable to those reported here. In addition, one could adapt the definition of similarity to more general scenarios. For example, there may be a confidence level associated with each term in a label, or a value representing a level of conformance with the term.

Some more specific issues should also be addressed. Since the expense of ranking is proportional to the number of candidates tested, an adaptive candidate-selection heuristic might be beneficial. The number of candidates generated by the KNN heuristic for a given value of N varies widely, so it could be applied for increasing values of N until a threshold number of candidates is exceeded.

Lastly, while the speed of our implementation was adequate for the task we studied, further improvements are possible. When calculating the inclusion probabilities for labels, terms from upper levels of the taxonomy occur often, since they represent the common ancestors of more specific terms that are only distantly related. These terms

tend to have lengthy inverted lists, making it expensive to compute their mutual intersection. Significant performance improvements might result from recognizing such term clusters and precomputing the intersections of their inverted lists.

References

1. M. Ashburner et al. Gene ontology: Tool for the unification of biology. *Nat. Genet.*, 25(1):25–29, May 2000.
2. R. Apweiler et al. Uniprot: the universal protein knowledgebase. *Nucleic Acids Res.*, 32(1):D115–119, Jan. 2004.
3. M. Halkidi, B. Nguyen, I. Varlamis, and M. Vazirgiannis. Thesus: Organizing web document collections based on semantics and clustering. Technical Report 230, INRIA Project Gemo, 2003.
4. J.M. Keller, M. Popescu, and J. Mitchell. Taxonomy-based soft similarity measures in bioinformatics. In *Proc. of the 2004 IEEE Int'l. Conf. on Fuzzy Systems*, 2004.
5. D. Lin. An information-theoretic definition of similarity. In *Proc. 15th Int'l. Conf. on Machine Learning*, pages 296–304. Morgan Kaufmann, San Francisco, CA, 1998.
6. P. W. Lord, R. D. Stevens, A. Brass, and C. A. Goble. Investigating semantic similarity measures across the gene ontology: The relationship between sequence and annotation. *Bioinformatics*, 19(10):1275–1283, 2003.
7. A. G. Maguitman, F. Menczer, H. Roinestad, and A. Vespignani. Algorithmic detection of semantic similarity. In *Proc. of the 14th Int'l World Wide Web Conf.*, pages 107–116, 2005.
8. P. Resnik. Using information content to evaluate semantic similarity in a taxonomy. In *IJCAI*, pages 448–453, 1995.
9. H. Wang, F. Azuaje, O. Bodenreider, and J. Dopazo. Gene expression correlation and gene ontology-based similarity: An assessment of quantitative relationships. In *The 2004 IEEE Symp. on Comp. Intelligence in Bioinformatics and Comp. Biology (CIBCB-2004)*, 2004.
10. Z. Wu and M. Palmer. Verb semantics and lexical selection. In *32nd. Annual Mtg. of the Assoc. for Comp. Linguistics*, pages 133 –138, New Mexico State Univ., Las Cruces, NM, 1994.

Towards an Inductive Methodology for Ontology Alignment Through Instance Negotiation

Ignazio Palmisano[1], Luigi Iannone[2], Domenico Redavid[1], and Giovanni Semeraro[1]

[1] Dipartimento di Informatica, Università degli Studi di Bari
Campus Universitario, Via Orabona 4, 70125 Bari, Italy
{palmisano, redavid, semeraro}@di.uniba.it
[2] Computer Science Department, Liverpool University
Ashton Building
Ashton Street
L69 BX Liverpool, UK
luigi@csc.liv.ac.uk

Abstract. The Semantic Web needs methodologies to accomplish actual commitment on shared ontologies among different actors in play. In this paper, we propose a machine learning approach to solve this issue relying on classified instance exchange and inductive reasoning. This approach is based on the idea that, whenever two (or more) software entities need to align their ontologies (which amounts, from the point of view of each entity, to add one or more new concept definitions to its own ontology), it is possible to learn the new concept definitions starting from shared individuals (i.e. individuals already described in terms of both ontologies, for which the entities have statements about classes and related properties); these individuals, arranged in two sets of positive and negative examples for the target definition, are used to solve a learning problem which as solution gives the definition of the target concept in terms of the ontology used for the learning process. The method has been applied in a preliminary prototype for a small multi-agent scenario (where the two entities cited before are instantiated as two software agents). Following the prototype presentation, we report on the experimental results we obtained and then draw some conclusions.

1 Motivation

The Semantic Web (SW), being an evolution of the World Wide Web, will inherit Web decentralized architecture. Decentralization, in its turn, was one of the success factors of the Internet, granting its structure with scalability, no single point of failures, and so on. However, in order to implement the SW scenario envisioned in [1], semantic convergence is crucial. This point has been always identified in the employment of ontologies that, even before the rise of SW, were considered as *"shared formalizations of conceptualizations"* [2].

In the SW vision, different actors that want to take advantage from interoperating should be able to converge onto shared ontologies in order to communicate. Such a problem turned out to be crucial and very difficult to work out. Indeed, each party involved has its peculiar view of the domain it shares with other parties. Very often, different applications are interested in different aspects of the same *world state*, e.g. in a

R. Meersman, Z. Tari et al. (Eds.): OTM 2006, LNCS 4275, pp. 1058–1074, 2006.

typical B2B scenario in which suppliers and final customers usually are interested into different aspects of the goods that are dealt with. Both quantitative and, more likely, qualitative concepts often turn out to be fuzzy depending on those actors interested in assessing them to interoperate. Suppliers may likely consider features like materials, provenance, standard processes, whereas customer satisfaction may depend also on other factors, e.g. warranties, comfort, support, which may turn out to be only marginal for the suppliers, unexpressed or simply difficult to acquire.

In such cases, partially overlapping visions of the same world should be integrated in value chains in order to provide goods. However, The wide range of possible B2B scenarios hinders a centralized approach to ontology sharing for semantic convergence, which contrasts with the architectural paradigm of the SW. Indeed, such a scenario requires flexibility, since it basically builds up open situations where the interacting actors cannot be precisely individuated in advance. Hence, a central ontology should foresee a high number of particular situations, and therefore it would be detailed enough, and therefore useful, only for restricted cases and toy domains. Moreover, single application ontologies are supposed to vary in time, an issue that is not easily manageable in centralized approaches.

We agree on the fact that semantic convergence has to be addressed at the ontology level, but we argue that the approaches for solving such issues should fulfil the following properties to be effectively applied in the mentioned scenarios:

- they have to be automatizable, so that applications could integrate among each other without human intervention (or with as little human intervention as possible);
- they have to be flexible. It should be possible to apply the same convergence schema to any domain on which the actors have to communicate their own (partially overlapping) conceptualizations;
- they have to be on-demand and limited in scope. The aim is not to fully map the actors conceptualization, which is often impossible or even useless, but to reconciliate only those parts that are necessary for the dialogue.

Developing platforms enabling SW systems to communicate with each other, without committing to a *superimposed* ontology, ensures a very loose coupling among them which allows for independent evolution and maintaining of the applications. In our opinion, other SW technologies spanning from Web Service discovery, orchestration, and choreography, up to software agents immersed in the SW, could profit from the solutions that research will find to these issues.

In this paper, we suggest a Machine Learning approach to accomplish semantic integration, taking into account the requirements listed above. The remainder of the paper is organized as follows: the next section will briefly summarize the state of the art (to our knowledge) on these problems. Sect. 3 illustrates our approach to Ontology Alignment together with a proposed algorithm. Sect. 4 explains in detail the Machine Learning techniques used to implement the algorithm, while Sect. 5 presents the implemented prototype employed to carry out a preliminary empirical evaluation reported in this section. Finally, in Sect. 6, some conclusions are drawn outlining future work directions.

2 Related Work

In this section, we present a short discussion of some recent relevant research describing different approaches to ontology alignment. Ontology Alignment is a broad umbrella for a variety of subtask that try to solve some of the issues pointed out in the previous section. In order to keep the subject simple and general, we might define alignment as: *any process that enables communication among two or more systems, adopting two or more different ontologies*. Following Noy [3], there are (at least) two types of alignment processes, classified according to their output:

- Ontology Merging: in which the outcome of the alignment of n ontologies is a single new one, embedding the knowledge coming from the n sources
- Ontology Mapping: in which the results of the alignment among n different ontologies consists of the starting ones plus the mapping between similar entities of the input sources.

In the following, we briefly discuss a non-exhaustive list of some remarkable research approaches pursuing such alignment strategies.

2.1 GLUE

GLUE [4, 5] exploits Machine Learning techniques to find semantic mappings between concepts stored in distinct and autonomous ontologies. Basically, its strategy falls into the Ontology Merging category as its output is a *mediator* ontology (or database schema in its earlier version). The process relies on a initial manual bootstrapping phase, in which user provides some sample mappings between some entities within the ontologies to be merged. GLUE then, tries to infer a set of rules (or classifiers) that can *synthesize* the mapping strategy followed by the users for the initial mappings. GLUE then, applies what it learnt to find other semantic mappings. One of its strong points is the possibility of using any kind of probabilistic similarity measure (measure neutrality) and, furthermore, also the weighting of the single similarity assessments is learnt by the system. In [4] the authors underline that there are many kinds of mappings that can be found, comparing entities from different ontologies, however GLUE focuses on finding 1-1 mappings between concepts belonging to two taxonomies.

2.2 PROMPT Suite

The PROMPT Suite by Stanford Medical Informatics [6, 3] provides two tools for alignment: iPROMPT and ANCHORPROMPT. iPROMPT is an interactive tool performing ontology merging. It starts from initial mappings among two entities (provided either by users or by a automatic process based on linguistic similarity) and generates suggestions for further mappings between other entities. After the user triggers one among the suggestions proposed (or performs some changes on the resulting ontology), iPROMPT applies the required changes, computes the possible cases of inconsistency trying to suggest fixes for those, and produces new suggestions for further mappings.

ANCHORPROMPT, on the contrary, is an ontology mapping tool. It can be used for supporting iPROMPT in order to individuate new related concepts during the iterations.

Unlike iPROMPT, it exploits also non-local contexts for the concept pairs whose similarity has to be assessed. Indeed, it represents ontologies as graphs and compares the paths in which concepts are involved, both within the taxonomy and in slot dependencies and domain/range constraints.

2.3 APFEL

APFEL is really a whole framework rather than a single alignment technique. In [7], Ehrig et al. propose a very interesting classification of alignment strategies dividing them into manually predefined automatic methods and learning from instance representations methods. The former are very complex to design, since all the possible peculiarities of the domains they will be applied must be considered; the latter, on the contrary, can be used only in presence of instances (and sometimes this is not the case) though they show more flexibility as the underlying ontology domain changes. The proposed framework (PAM - Parameterizable Alignment Methodology) is fully parameterizable to the extent of being able to reproduce the strategy used in other methodologies by just adjusting some parameters. APFEL tries to learn from pre-validated example mappings the right parameters that would instantiate the fittest PAM for the particular learning problem. Such parameters are:

- *Features*, i.e. the smallest set of features on which to compute similarity among entities
- *Selection criteria* for the next pair of entities to compare
- *Similarity measures*, aggregation, and interpretation strategies
- *Iteration*, that is the extent to which neighbor entity pairs are influenced by the current pair (whose similarity has been evaluated).

3 Ontology Alignment Algorithm

According to Noy's classification cited in the previous section, the approach described in this work falls into the ontology mapping methodology. Indeed, without loss of generality, we assume to work on two ontologies dealing with the same domain. Besides, we also assume that some of the concepts within the input ontologies may partially overlap in their semantics. We intentionally will not define formally what this semantic overlap means, due to the large variety of practical situations in which this may happen in real-world applications. We might have concepts defined with different names, structures, etc. that share exactly the same meaning. On the other hand, we can have concepts that, though sharing their name, might be different in their actual meaning, and it could be necessary that an application understands what its peer exactly means for such a concept, possibly in terms of its own ontological primitives.

A couple of examples can better explain these situations. The former case can be exemplified as follows. Suppose that there are two applications dealing with two ontologies about cars, differing just for the language: one uses Italian names and the other English names. The concept of, say, Wheel is equivalent to the concept Ruota in the Italian version of the car ontology. The latter case, instead, may happen when you have

two ontologies describing a domain from two different perspectives. Suppose that the ontology FoodRestaurant deals with food from the typical point of view of a restaurant and that the ontology FoodNutritionist deals with the same subject from the side of a dietician. The concept of HighQualityFood depends on different aspects of food according to the adopted viewpoint. Indeed, when defining this concept, a dietician, would care of nutritional values of the ingredients, whereas a chef would take into account the provenance of the ingredients, their rarity and so on. In such case, we have two legal HighQualityFood concepts; an agent for dinner arrangements dealing with restaurants and people could be more effective if it knows both senses.

The intuition behind our approach is that a useful mapping is the one in which at least one of the two parties can *understand* what the other means in terms of its own ontology. Indeed, if an application is to use information coming from another system, it should be able to frame this information in its own knowledge model, hence, in our opinion, it has to grasp some meaning using primitive concepts and properties from its own ontology. Though this may seem straightforward, many approaches limit themselves to find a correspondence between equivalent (or very similar) entities, provided that such an equivalence exists. Yet such approaches disregard the importance of maintaining different definitions for the same entity (the viewpoints discussed in the previous example) and of having one's own definitions, in order to communicate with different parties.

Therefore, a way is needed for allowing a system to build a representation of a concept belonging to another ontology using its own terminology. We argue that the inductive reasoning techniques of Machine Learning are a suitable way to accomplish this objective. In particular, we employ the *Concept Learning* paradigm [8] for treating this problem. Indeed, with Concept Learning aims at inducing an intensional descriptions of concepts starting from its examples and (possibly) counterexamples.

In particular, we intend to implement the following scenario. Suppose there are two applications A and B with their respective ontologies O_A and O_B and suppose that application A wants to communicate about some concept, say $a : C$, with B (we suppose to indicate concepts with the usual URIs, i.e. namespace:conceptLocalName) regarding a concept that does not appear in O_B. Then, in our scenario B could ask A to provide some instances (examples) of $a : C$ and, possibly, also some counterexamples (i.e.: instances of the complement class of $a : C$). If such examples occur as instances within O_B, then B, by means of a Concept Learning algorithm, can infer a definition of $a : C$ in terms of the descriptions of those instances contained in O_B. In this way B obtains a definition of $a : C$ using its own terminology, that is, in other words, B *understands* $a : C$ according to what it knows (its ontology). In the following section (Sect. 4) the particular Concept Learning algorithm employed is described; however, for the purposes of this section the particular algorithm is irrelevant since the only requirement is that it solves a generic Concept Learning problem, described formally in the following definition:

Definition 3.1 (Concept Learning problem)
Given *a knowledge base $\mathcal{K}$ and a set of instances divided into examples and counterexamples $E = E^+ \cup E^-$ and described in $\mathcal{K}$*
Learn *the definition of a concept C such that all elements of E^+ none of E^- are instances of C.*

The result of the learning problem can be further revised in the following way. B identifies a suitable subset of instances (different from those provided by A as a training set), classifies them as belonging or not to the learnt concept and passes them to A asking the peer to do the same. If A and B disagree on the classification of some individuals, it will be the case that B refines properly the learnt definition in order to fix the discrepancy with the classification provided by A.

After learning a definition of $a : C$, let us call it $b : C$ supposing there are no clashes with pre-existing names, B has to compare it with the concepts in its O_B, evaluating the similarity of $b : C$ to pre-existing concepts. This can accomplished using suitable similarity measures such as the one proposed in [9] that score the semantic affinity/diversity of a pair of concepts within an ontology. If the similarity between $b : C$ and another concept in O_B exceeds a given threshold or, better, if the equivalence can be proved by a reasoner, then an equivalence axiom can be added, otherwise the definition of $b : C$ is left unchanged within O_B.

Careful readers should have already found out that there is an assumption on which the whole process relies on. The assumption is that there exists a subset of individual URIs in common between O_A and O_B. This corresponds to the initial knowledge that is provided in other systems during the bootstrapping phase. To cite only the related work, GLUE needs a complete example of mapping to start the learning phase, whereas iPROMPT relies on initial suggestions on linguistic similarity, assuming that concept names are expressed in their natural language form (or very close to it). ANCHORPROMPT, on the contrary, exploits the graph structure assuming that, for ontologies partially overlapping on the same domain, their graph structure should be similar. Our assumption seems at least as reasonable as these, also because it can be relaxed thinking that, instead of having some identical individual URIs across ontologies, we have mechanisms to map some URIs into some others (either manually or automatically).

Consider, for instance, the ontologies about paper indexing systems, e.g. DBLP and the ACM Digital Library, and suppose there exist their OWL (or RDF) [1] versions. Each system has automatic strategies for generating identifiers and there is a large subset of research works that are indexed by both of them. It could be easy then to provide URIs translator for the two systems. Actually, ACM and DBLP bibliographic information are used in one of the test cases we devised for our system; more details in Sect. 5.3.

4 Machine Learning in Description Logic

In this section, we describe in more details the learning algorithm we developed for solving a Concept Learning Problem in Description Logic. In particular, we focus on concept descriptions in the $\mathcal{ALC}$ [10] Description Logic as it is the least expressive subset of OWL-DL allowing for disjunction and ensuring a reasonable expressiveness without features (like role hierarchies) that would make learning intractable.

First we cast the generic Concept Learning Problem reported in Def. 3.1 to the Description Logic setting.

[1] http://www.w3.org/2004/OWL/, http://www.w3.org/RDF/

Definition 4.1 (learning/tuning problem). *In a search space of concept definitions* $(\mathcal{S}, \sqsubseteq)$ *ordered by subsumption*
Given *a knowledge base* $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ *and a set of positive and negative assertions* $\mathcal{A}_C = \mathcal{A}_C^+ \cup \mathcal{A}_C^-$ *regarding the membership (or non-membership) of some individuals to a concept* C *such that:* $\mathcal{A} \not\models_{\mathcal{T}} \mathcal{A}_C$
Find *a new T-box*[2] $\mathcal{T}' = (\mathcal{T} \setminus \{C \equiv D\}) \cup \{C \equiv D'\}$ *such that:* $\mathcal{A} \models_{\mathcal{T}'} \mathcal{A}_C$

A Concept *Tuning* (or Revision) Problem is similar but for the fact that learner already knows an incorrect definition for C which has to be revised. This means that there are some assertions within $\mathcal{A}_C$ that are not logical consequences of the knowledge base and the current definition of C. Such incorrect definition can be *incomplete*, i.e.: there are some individuals that are said to belong to C in $\mathcal{A}_C$ but this cannot be entailed, and/or it can be *incorrect*, meaning that there are individuals that are deduced to belong to C from the definition of C, while they actually don't belong to it.

In general, Concept Learning relies on the intuition that the solution of any problem can be found traversing a proper search space.

A learning algorithm, then, is cast as a search that moves across such a space with two kinds of possible movements, called *refinements*:

- Specific towards general (*generalization*)
- General towards specific (*specialization*)

We provide here the formal (though generic) definitions for both kinds of refinement.

Definition 4.2 (downward refinement operator - ρ). *Let* $(\mathcal{S}, \preceq)$ *be a search space of concept descriptions* C *ordered by subsumption* $\sqsubseteq$. *We define the function* $\rho : \mathcal{S} \to 2^{\mathcal{S}}$ *such that* $\rho(C) \subseteq \{E \in 2^{\mathcal{S}} \mid E \sqsubseteq C\}$.

Definition 4.3 (upward refinement operator - δ). *Let* $(\mathcal{S}, \preceq)$ *be a search space of concept descriptions* C *ordered by subsumption relation* $\sqsubseteq$ *we define the function* $\delta : \mathcal{S} \to 2^{\mathcal{S}}$ *such that* $\delta(C) \subseteq \{E \mid E \in 2^{\mathcal{S}} \wedge C \sqsubseteq E\}$.

Notice that it does not hold in general that:

$$E \sqsubseteq C \to E \in \rho(C) \text{ or } C \sqsubseteq E \to E \in \delta(C)$$

Refinement has been thoroughly studied in literature (see [11, 12]) also specifically for Description Logic knowledge representations [13]. For the sake of brevity we will not expose here all the theoretical framework for refinement. It suffices to saying that, unlike many other approaches, we try to use examples and counterexamples of the concept to be learnt to bias the search and, hence, the refinement strategies, rather than using them just for testing intermediate candidate refinements. Indeed, during learning, as the algorithm searches for a solution of the problem, it may find various intermediate hypotheses that satisfy the required constraints[3] but not all of them. Hence refinement

[2] T-box (terminological box), is the portion of the knowledge base containing the theory ($\mathcal{T}$), while A-box (assertional box) is the portion containing the ground statements.

[3] It should covers a subset of examples and does not cover a part of counterexamples, but there would remain some uncovered examples and/or some covered counterexamples.

operators must be applied to fulfil such constraints. The problem is that at each refinement step there are many alternative directions that can be chosen. Examples come into play in this choice, guiding the refinement process towards selecting the most promising candidates, thus pruning the search space and avoiding the learner to backtrack for the sake of efficiency.

Our solution to the learning problem is the procedure reported as Algorithm 1 and 4 that are implemented in the learning system named YINYANG that is an evolution of the system described in [14]. In YINYANG examples are not processed at the individual description level but, for each of them, a concept level representative is computed. This is very frequent in Machine Learning when the example and hypothesis languages do not coincide, and is called *Single Representation Trick*. Concept level representatives should be as much adherent as possible to the examples they stand for. In Description Logic there exist a non standard inference called Most Specific Concept (denoted msc) that fulfills such a requirement but, unfortunately, it needs not exist for many Description Logics (such as $\mathcal{ALC}$). Hence it will be approximated provided that it has to be possible to distinguish among concept level representative of positive and negative examples. In other words it cannot exist a unique concept level representative for a positive and a negative example.

The algorithm starts choosing a seed that is a concept level representative of an example. Then, it keeps generalizing it until there are residual positive to be covered or until the generalization covers some negative example (overgeneralization). In the former case, the algorithm exits returning a complete and correct generalization. In the latter, specialization is necessary in order to correct the overly general hypothesis (concept definition).

Specialization can take place in two different ways. The first one is based on the notion of *counterfactuals* [15]. The idea behind is that if we have a concept that covers some negative examples one can single out the parts of such definitions that are *responsible* for the coverage and rule them out by means of negation. There is a well known *non-standard* inference in DLs called *concept difference* or *residual* [16] used for the aforementioned identifications of part of definitions which are responsible for the incorrect coverage (also known as *blame assignment*). For eliminating in one single negation all the different parts that are responsible for the coverage of each negative, a generalization of the blamed parts is needed in order to obtain a single description to be negated (i.e. a counterfactual). Thus, again generalization is needed, for the negative residuals with respect to the overly general hypothesis. Such negative residuals, will now become positive examples in a new nested learning problem and the solution of such problem will be our counterfactual. Since its negation has to be conjoined to the starting incorrect hypothesis in order to specialize, one do not want to incur in the dual error, i.e. overspecialization. Indeed, the refined resulting concept definition must keep covering the positive examples it covered before specialization. Hence the counterfactual should not cover the residual of covered positive example representatives w.r.t. the covering hypothesis to specialize. This means that they represent negative examples for the new nested learning problem solved by the recursive call.

Specialization by means of counterfactuals can be proved not to be complete in the sense that it sometimes fails in finding a correct counterfactual for properly specializing

Algorithm 1. YINYANG algorithm

Generalize(Positives, Negatives)
input: Positives, Negatives: positive and negative msc approximations
output: Generalization: generalized concept definition
ResPositives ← Positives
Generalization ← $\bot$
while ResPositives $\neq \emptyset$ **do**
  $ParGen \leftarrow$ select_seed$_M$(ResPositives)
  CoveredPos ← $\{Pos \in$ ResPositives $\mid ParGen \sqsupseteq Pos\}$
  CoveredNeg ← $\{Neg \in$ Negatives $\mid ParGen \sqsupseteq Neg\}$
  while CoveredPos $\neq$ ResPositives $\wedge$ δ returns a proper refinement $\wedge$
  CoveredNeg $= \emptyset$ **do**
    $ParGen \leftarrow$ select$(\delta(ParGen)$, ResPositives, Negatives$)$
    CoveredPos ← $\{Pos \in$ ResPositives $\mid ParGen \sqsupseteq Pos\}$
    CoveredNeg ← $\{Neg \in$ Negatives $\mid ParGen \sqsupseteq Neg\}$
  end while
  if CoveredNeg $\neq \emptyset$ **then**
    $K \leftarrow$ **Counterfactual**$(ParGen$, CoveredPos, CoveredNeg$)$
    ParGen ← ParGen $\sqcap \neg K$
    CurrentCoveredPos ← $\{Pos \in$ ResPositives $\mid ParGen \sqsupseteq Pos\}$
    LeftOutPositives ← CoveredPos \ CurrentCoveredPos
    if |LeftOutPositives| > 0 **then**
      LeftInPositives ← CurrentCoveredPos \ CoveredPos
      if LeftInPositives $= \emptyset$ **then**
        ParGen ← Add conjunct$\rho^E_\sqcap$(ParGen, LeftOutPositives, coveredNegatives)
        if Add conjunct$\rho^E_\sqcap$ failed **then**
          ParGen ← lcs(coveredPositives)
        end if
      else
        CoveredPos ← LeftInPositives
      end if
    end if
  end if
  Generalization ← Generalization $\sqcup$ ParGen
  ResPositives ← ResPositives \ CoveredPos
end while
return Generalization

Counterfactual(ParGen, CoveredPos, CoveredNeg, K)
Input ParGen: inconsistent concept definition
Input CoveredPos, CoveredNeg: covered positive and negative msc approx.
Output K: counterfactual
NewPositives ← $\emptyset$
NewNegatives ← $\emptyset$
for $N_i \in$ CoveredNeg **do**
  NewP$_i$ ← residual$(N_i$, ParGen$)$
  NewPositives ← NewPositives $\cup$ {NewP$_i$}
end for
for $P_j \in$ CoveredPos **do**
  NewN$_j$ ← residual$(P_j$, ParGen$)$
  NewNegatives ← NewNegatives $\cup$ {NewN$_j$}
end for
$K \leftarrow$ **generalize**(NewPositives, NewNegatives)
return K

Algorithm 2. Add Conjunct ρ Implementation

$rho^{E}_{\sqcap}$(c, coveredPositives, negatives)
Input: c is a Concept
Input: coveredPositives is a set of positive examples concept level representatives
Input: negatives is a set of negative examples concept level representatives
Output: A specialization covering all elements within input when possible or c itself
toReturnConcept ← c
noPossibleRefinementKeepingAllPositives ← false
moreChanches ← true
cs ← ∅
alreadyChosenConjuncts ← new empty map that associates positive examples with already used conjuncts
coveredNegatives ← negatives
while |coveredNegatives| $>$ 0 $\wedge$ (¬noPossibleRefinementKeepingAllPositives $\vee$ moreChanches) **do**
  moreChanches ← false
  for each p ∈ coveredPositives **do**
    possiblyLeftOut ← ∅
    if $p \not\sqsubseteq \bigsqcup_{C \in \text{cs}} C$ **then**
      availableConjuncts ← chooseConjunct
        (p, toReturnConcept, coveredNegatives, alreadyChosenConjuncts)
      aSelectedConjunct ← selectConjunct(availableConjuncts)
      if $lVert$availableConjuncts| > 1 **then**
        moreChanches ← true
      end if
      if aSelectedConjunct $\not\sqsubseteq \bot$ **then**
        cs ← cs ∪ {aSelectedConjunct}
        Add aSelectedConjunct to alreadyChosenConjuncts for p
      end if
    end if
  end for
  if |cs| > 0 **then**
    toReturnConcept ← $c \sqcap$ lcs(cs)
    noPossibleRefinementKeepingAllPositives
      ← ¬($\forall p \in$ coveredPositives : $p \sqsubseteq$ lcs(cs))
    if noPossibleRefinementKeepingAllPositives **then**
      cs ← ∅
    end if
  else
    noPossibleRefinementKeepingAllPositives ← true
  end if
  coveredNegatives ← {$n \in$ negatives, $n \sqsubseteq$ toReturnConcept}
end while
return toReturnConcept

overly general hypotheses. That is why another specialization strategy was introduced, named Add Conjunct, which is applied whenever the Counterfactual routine fails. Such a strategy aims at finding a conjunct per positive example that does not appear yet in the hypothesis to be specialized nor in none of the negatives that are currently

covered. In order to minimize the number of conjuncts, each positive provides such a conjunct only if there are no conjuncts (already provided by some other positives) that cover it. After having computed such conjuncts, they are generalized into their least common subsumer (lcs) [10] and such lcs is conjoined to the overly general hypothesis. Since the requirements for a conjunct to be chosen for each positive are very tight, it may sometimes happen that such specialization strategy fails as well as counterfactual before it. In such cases, the overly general hypothesis is simply replaced by the lcs of its covered positive example representatives, such lcs is added as a disjunct to the main generalization and the algorithm chooses another seed starting again the inner loop.

5 Prototype Description and Preliminary Evaluation

We developed a preliminary prototype that implements the methodology in Sect. 3. We thought that one of the most suitable settings could be a Multi-Agent System. Indeed, it seems natural to think of communicating actors as autonomous agents that have their own knowledge bases and wish to dialogue with each other. In such environments, it is very likely that there is no common ontology to commit to; this can offer plenty of possible scenarios in which ontology alignment can be evaluated.

Here we present the prototype in terms of its architecture and we propose an evaluation on some simple alignment problems that are characteristic (in our opinion) of typical situations that can happen in such settings. We conducted such simple tests in order to single out the direction towards which a mature system should move as well as the practical issues that arise. Therefore, these tests are not to be considered a thorough empirical evaluation, which will follow once robust solutions for the issues described in the following have been devised.

Our prototype is called SELA (Self Explaining and Learning Agents) and consists of a software agent (namely SELAAgent) that has two main tasks: it can either explain concepts, providing examples and validating classifications, or it can learn concepts from examples acquired by some other SELAAgent and ask for validations of what it learned. The agent has been developed within the framework provided by the Java Agent Development Environment (JADE http://jade.tilab.com) and the alignment methodology has been designed as a protocol as sketched in Fig. 1.

We can describe it briefly as follows. A SELAAgent A (the teacher) initiates a conversation with another SELAAgent B (the learner) asking whether B knows the concept $a : C$ which is what A wants to speak about. B can confirm it knows $a : C$ or disconfirm, stating it does not. In case of disconfirmation A provides some individual names that are instances of $a : C$ and some names of counterexamples of $a : C$ (individuals in the extension of the complement of $a : C$). After receiving the names of the instances (both examples and counterexamples) B takes into account only the subset of individuals it owns in its ontology. Then B starts learning as described in the previous Section. Notice that instance descriptions are not to be provided by A, but they are built on the assertions that B has about such individuals. When learning terminates, B has a definition of $a : C$ in terms of its ontology vocabulary. As we shall see in the following,

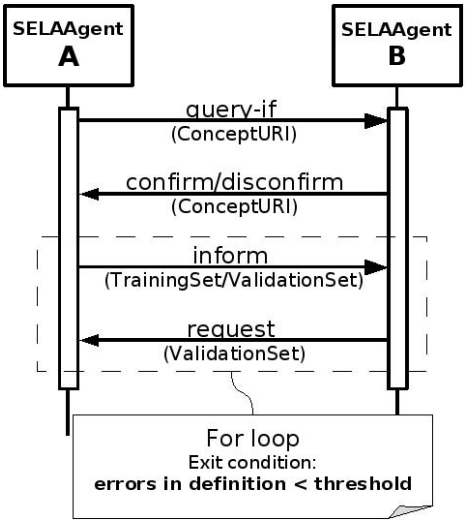


Fig. 1. SELA Alignment Protocol

this needs not to be exactly corresponding to the definition of $a : C$ that is actually stored into the ontology of A. B then, has somehow to be sure that its *idea* of $a : C$ corresponds to the one of A. This can be accomplished to some extent by means of the cyclic remainder of the protocol. B chooses some individuals and classifies them employing the definition of $a : C$ it has learned. Then, it sends the classification results to A that validates them. A answers with validation results and then B can fine-tune (see Def. 4.1) its definition $a : C$. The loop stops when the number of discordances between B classifications and A validations is below a given threshold.

5.1 Artificial Ontologies

We prepared two sample pairs of ontologies. The first pair is made of two ontologies that are structurally identical but for concept and property names. In fact, we prepared an ontology in the academic domain with an English nomenclature and then its translation in Italian (see Fig. 2 for its layout in Protégé).

The second pair presents a different situation. We started from a common ontology in a fancy restaurants domain that deals with food, meals, and courses. Actually, it is a simplified version of the famous food ontology sample on the W3C website.[4] Then we derived two ontologies containing both the concept called HighQuality defined, however, in two different namespaces (in order to represent two different points of views for it). One deals with HighQuality from the perspective of a particular restaurants defining it as:

$$\mathsf{HighQuality} \equiv \mathsf{Meal} \sqcap ((\exists \mathsf{hasCourse.Starter} \sqcap \exists \mathsf{hasCourse.MainCourse} \sqcap \sqcap \exists \mathsf{hasCourse.Dessert}) \sqcup \exists \mathsf{hasCourse.}(\exists \mathsf{hasDrink.Alcoholic}))$$

[4] http://www.w3.org/TR/2002/WD-owl-guide-20021104/food.owl

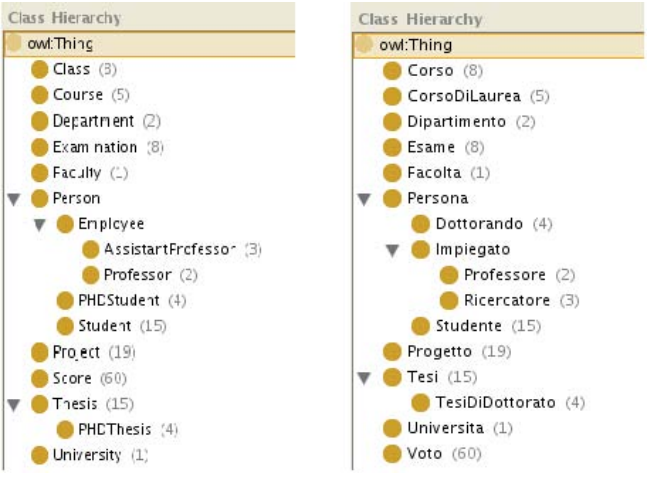


Fig. 2. English and Italian Academy ontologies

The other one defines its notion of HighQuality from the hypothetical perspective of a dietician as:

$$\mathsf{HighQuality} \equiv \mathsf{Meal} \sqcap (\forall \mathsf{hasCourse}.(\forall \mathsf{hasDrink}.(\neg \mathsf{Alcoholic}) \sqcap \exists \mathsf{hasFood}(\mathsf{LowCaloric} \sqcup \mathsf{NegligibleCaloric} \sqcup \mathsf{ModerateCaloric})))$$

It is obvious that there are two different alignment purposes and situation. Indeed, in the former, the process should detect that there is a complete correspondence among the entities of the two ontologies. In the latter, there is not a 1-1 mapping between the two conceptualizations of HighQuality but it could be interesting if each party (restaurant owner and dietician) could build up, in their respective knowledge base, the point of view of their counterpart during the dialogue.

As concerns the mapping between the two academic ontologies, we report briefly that for each concept the learning SELAAgent was able to build up a definition that was equivalent to the corresponding translated concept in its own ontology, even when a few instances where exchanged. Such a good result depends obviously on the structural similarity of the ontologies: the individuals in common between the two ontologies have the same structure but for the names of the relations; moreover, the A-box contains all the relevant information for the individuals, which in general may not be the case (see Sect. 5.3 for further details).

In the restaurant domain the results were not as satisfactory as the previous case. Consider the more likely scenario in which the agent with the restaurant owner version of HighQuality tries to learn the dietician's concept of HighQuality. It learns a definition that is more specific than the desired one though it never required tuning in all the iterations we ran through. This is likely due to the limitedness of the number and especially of the variety of examples. It is indeed well known in inductive learning that too few examples not so different among each other may yield a phenomenon called *overfitting*. Overfitting occurs when the learned definition is too adherent to the training set used for building it up and hence suffers for poor predictive accuracy on future unseen

examples classification. Future experiments will aim to devise also suitable strategies for choosing examples and, above all, counterexamples. As a matter of fact, a careful selection of counterexamples could improve the effectiveness of learning. Winston [17] claimed the usefulness of *near-misses* counterexamples that are instances that very close to the concept to be learnt but not belonging to its extension. In our case, we could exploit the taxonomy for individuating the most likely near-misses: e.g. the instances of the superconcepts of the concept to be learned (or those of its sibling concepts) that do not belong to it.

5.2 Ontology Alignment Evaluation Initiative Ontologies

Our second test is based on two ontologies taken from the Ontology Alignment Evaluation Initiative test set for 2005, namely onto101 and onto206[5]; onto101 is the reference ontology for the contest, which consists on a set of ontologies that are modifications of onto101 (e.g., hierarchy flattening, deletion of concepts, translation of names, URIs, comments, or deletion of comments). In particular, onto206 is the French translation of onto101, where all local names of the defined classes and properties have been translated from English to French. These ontologies consist of 39 named classes, 24 properties and over 70 individuals, of which about 50 are identified by an URI; these individuals have the same URI in both onto101 and onto206, and this enables us to apply our algorithm to them; in particular we assigned onto101 to one of our agents (the teacher) and onto206 to the other agent, then we made the teacher agent try to teach the definition of the http://oaei.inrialpes.fr/2005/benchmarks/101/onto.rdf#Institution[6] concept; however, the resulting definition:

$$\exists \text{adresse.Adresse} \sqcap \text{Éditeur}$$

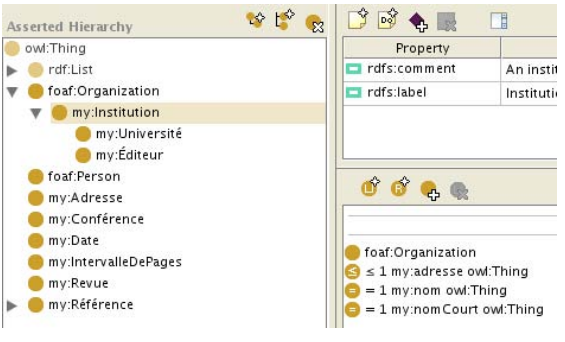


Fig. 3. inria206:Institution Definition

[5] Full test set at http://oaei.ontologymatching.org/2005/; as a side note, it is necessary to operate some corrections on the ontologies, since there are some small errors: the literals for year, month and day should be typed with the corresponding XSD types, but they are plain literals in both ontologies, and the DIG reasoner we use (Pellet, http://www.mindswap.org/2003/pellet/) finds the ontology inconsistent.

[6] Namespaces will be shortened in the following.

is poor w.r.t. the original definition (Fig. 3) and moreover it is overfitting (being Éditeur is not necessary to be Institution); while the second issue only depends on the available individuals (only two individuals of class Éditeur were available in the ontology), the first issue depends on the targeted DL, since the expressivity of the ontologies we use here is greater than that of the language our learner uses (specifically, cardinality restrictions are not handled). This example clearly shows that at least cardinality restrictions should be added to the representation language of the algorithm in order for it to be useful in real world scenarios.

5.3 ACM and DBLP Test Case

Our last test case has been built from ACM SIGMOD dataset[7] and from DBLP RDF dump[8]; the ACM dataset is an XML file with associated DTD; in order to translate it into OWL, we designed a very small ontology reflecting the DTD, and then produced the OWL ontology we needed.

In order to find common individuals, we analyzed the available data and found that a good way to identify matching individuals for this setting is to look at the paper titles; this enabled us to choose a subset of the DBLP individuals (amounting to roughly 700 individuals) that were described both in terms of the DBLP ontology and of our homemade ACM SIGMOD ontology (both sketched in Fig. 4[9]; the learning problem here consisted in finding the definition for the concept Article in the ACM SIGMOD ontology; the definition we obtain for Article in the original ACM SIGMOD ontology is:

$$\mathsf{sigmod{:}Article} \sqcap \exists \mathsf{sigmod{:}author}.\top$$

while the definition w.r.t. the DBLP ontology is:

$$\mathsf{dblp{:}Article} \sqcap \exists \mathsf{dblp{:}url}.\top \sqcap \exists \mathsf{dblp{:}ee}.\top$$

The definitions appear to be very short; in fact, this depends on the two chosen ontologies being very small, and having mainly datatype properties, which are not useful in the learning algorithm.

At the time of writing, no quantitative tests have been run on this dataset; the reason is that, being all the individuals in DBLP and ACM SIGMOD almost equal from the structural point of view, a very small amount of examples is enough to converge on the presented definitions. These definitions score 100% precision on the presented datasets, but the reason for this high performance lies in the regularity of the dataset (the actual number of examples used in the tests is 10 positive examples and 5 negative examples out of 700 available individuals). Also, the fact that so many properties in real ontologies are defined as datatype, even when they in fact represent entities (e.g. dblp:author

[7] Available at http://www.cs.washington.edu/research/xmldatasets/www/repository.html

[8] Available at http://www.semanticweb.org/library/

[9] It is worth noting that the DBLP RDF dump needed a little data cleaning, since it was not completely conformant RDF; in particular, many URIs in the data contained spaces, that needed handling before being submitted to the reasoner.

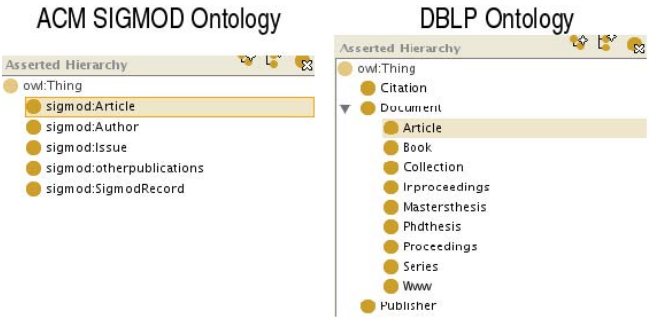


Fig. 4. ACM SIGMOD and DBLP Ontologies

is a datatype property, while usually an author of a paper is a person, and so is typically modeled as an individual) raises the idea that building more structured datasets is necessary before attempting a complete empirical evaluation of the system.

¿From the computational complexity point of view, it is well known that OWL DL reasoning algorithms have exponential worst-case complexity[10]; however, most real world ontologies do not exploit all DL constructors, and therefore reasoning with these ontologies can be done in reasonable time. We conducted a very preliminary test on the DBLP dataset presented, where we sliced the available examples in 10 disjoint sets and ran the learning algorithm separately; each learning problem was made up of 50 positive examples (randomly chosen) and 5 negative examples, and the elapsed time for building the definition is around one minute for each one of the ten trials. The last trial was made using all individuals at once, so that there were 500 positive examples and 50 negative examples; in this case, the required time is 80 seconds. So far, then, the number of examples seems not to hamper the practical use of the algorithm.[11]

6 Conclusions and Future Work

Giving solutions to the ontology alignment problem is one of the most compelling issues in the roadmap to realize the Semantic Web as a real-world infrastructure. We recalled the motivations for the investigation on this subject and underlined our own viewpoint on alignment. In such a vision, we prefer to slightly stray from the most widespread idea of a rigid centralized (top level) ontology and to adopt on-demand partial mappings among ontologies owned by the parties in play. We have proposed a concept learning algorithm to accomplish this that works under assumptions justified throughout the paper. We have illustrated the prototype implemented these ideas together with some preliminary results using limited datasets.

[10] http://www.cs.man.ac.uk/ ezolin/logic/complexity.html

[11] Note that this last test was designed as a ten-fold cross validation experiment, however the results of the test seem too influenced by the specific dataset to be taken into account to measure the performance of the algorithm in terms of precision and recall, and that's the reason they're not presented here in detail.

We plan to improve our work along the following directions. First, we should evaluate suitable concept similarity measures for assessing the closeness of learned concept definitions to the pre-existing conceptualizations in the ontologies to be aligned. We will also investigate on methodologies for aligning properties and not only concepts, using techniques that are very similar to those employed in tools like GLUE (see Sect. 2). Last, but not least, we will evaluate the impact of different strategies for example selection in terms of the quality (effectiveness and/or efficiency) of the induced definitions and, therefore, of the computed alignment themselves.

References

[1] Berners-Lee, T., Hendler, J., Lassila, O.: The Semantic Web. Scientific American (2001)
[2] Gruber, T.R.: A translation approach to portable ontology specifications (1993)
[3] Noy, N.F.: Tools for mapping and merging ontologies. In Staab, S., Studer, R., eds.: Handbook on Ontologies. International Handbooks on Information Systems. Springer (2004) 365–384
[4] Doan, A., Madhavan, J., Domingos, P., Halevy, A.Y.: Learning to map between ontologies on the semantic web. In: WWW. (2002) 662–673
[5] Doan, A., Domingos, P., Halevy, A.Y.: Learning to match the schemas of data sources: A multistrategy approach. Machine Learning **50** (2003) 279–301
[6] Natalya F. Noy, M.A.M.: The prompt suite: interactive tools for ontology merging and mapping. International Journal of Human-Computer Studies **59** (2003) 983–1024
[7] Ehrig, M., Staab, S., Sure, Y.: Bootstrapping ontology alignment methods with apfel. In Gil, Y., Motta, E., Benjamins, V.R., Musen, M.A., eds.: International Semantic Web Conference. Volume 3729 of Lecture Notes in Computer Science., Springer (2005) 186–200
[8] Mitchell, T.M.: Concept Learning and General to Specific Ordering. In: Machine Learning. McGraw-Hill, New York (1997) 20–51
[9] dAmato, C., Fanizzi, N., Esposito, F.: A semantic similarity measure for expressive description logics. In: Proceedings of CILC 2005. (2005)
[10] Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P., eds.: The Description Logic Handbook. Cambridge University Press (2003)
[11] van der Laag, P.R.J., Nienhuys-Cheng, S.H.: Existence and nonexistence of complete refinement operators. In Bergadano, F., Raedt, L.D., eds.: ECML. Volume 784 of Lecture Notes in Computer Science., Springer (1994) 307–322
[12] Nienhuys-Cheng, S., de Wolf, R.: Foundations of Inductive Logic Programming. Volume 1228 of LNAI. Springer (1997)
[13] Badea, L., Nienhuys-Cheng, S.H.: A refinement operator for description logics. In Cussens, J., Frisch, A., eds.: Proceedings of the 10th International Conference on Inductive Logic Programming. Volume 1866 of LNAI., Springer (2000) 40–59
[14] Esposito, F., Fanizzi, N., Iannone, L., Palmisano, I., Semeraro, G.: Knowledge-intensive induction of terminologies from metadata. In McIlraith, S.A., Plexousakis, D., van Harmelen, F., eds.: Proceedings of the 3rd International Semantic Web Conference, ISWC2004. Volume 3298 of LNCS., Springer (2004) 411–426
[15] Vere, S.: Multilevel counterfactuals for generalizations of relational concepts and productions. Artificial Intelligence **14** (1980) 139–164
[16] Teege, G.: A subtraction operation for description logics. In Torasso, P., Doyle, J., Sandewall, E., eds.: Proceedings of the 4th International Conference on Principles of Knowledge Representation and Reasoning, Morgan Kaufmann (1994) 540–550
[17] Winston, P.: Learning Structural Descriptions from Examples. MIT (1970) Ph.D. dissertation.

Combining Web-Based Searching with Latent Semantic Analysis to Discover Similarity Between Phrases

Sean M. Falconer[1], Dmitri Maslov[2], and Margaret-Anne Storey[1]

[1] University of Victoria, Victoria BC V8W 2Y2, Canada
{seanf, mstorey}@uvic.ca
[2] University of Waterloo, Waterloo ON N2L 3G1, Canada
dmitri.maslov@gmail.com

Abstract. Determining semantic similarity between words, concepts and phrases is important in many areas within Artificial Intelligence. This includes the general areas of information retrieval, data mining, and natural language processing. Existing approaches have primarily focused on noun to noun synonym comparison. We propose a new technique for the comparison of general expressions that combines web searching with Latent Semantic Analysis. This technique is more general than previous approaches, as it is able to match similarities between multi-word expressions, abbreviations, and alpha-numeric phrases. Consequently, it can be applied to more complex comparison problems such as ontology alignment.

1 The Problem

In many domains and applications, understanding semantic similarity, or the semantic relationships between expressions is an important problem. With some applications, one may be interested in direct synonymy, and often thesauri like Roget's thesaurus or WordNet [9] are used to try to solve this problem. However, there are different types of semantic relationships one may be interested in. For example, besides direct synonymy, one may be interested in correlations between a term and an instance of that term (*e.g.* "Human" to "Albert Einstein"), or possibly in an association (*e.g.* "Cooking" and "Food"). Previous work has largely been task or domain specific, such as limited to English synonym matching or restricted to a domain such as medical terms. In contrast, we are interested in the general problem of calculating expression similarity and we were led to this problem through our interest in ontology alignment.

An ontology is a conceptualization of a domain [7]. This conceptualization consists of a set of terms with certain semantics and relationships [18]. Generally, the terms are related by the `is_a` relationship, however, other relationships such as `has_a` often exist. In ontology alignment, one wants to map one ontology to another by matching like terms between the two ontologies so that the ontologies can be merged or compared.

Often, ontology alignment tools combine various heuristics to determine similarity between terms. With domain specific ontologies, synonym matching is sometimes performed using databases like UMLS (Unified Medical Language System) [12]. However, it is difficult to apply synonym matching in general, as most existing techniques either depend on a specific domain of knowledge or are restricted to a particular language.

R. Meersman, Z. Tari et al. (Eds.): OTM 2006, LNCS 4275, pp. 1075–1091, 2006.

Moreover, other semantic relationships exist within the ontologies that are equally important to map and understand. For example, by understanding the `has_a` relationship, the internal structure of terms can be mapped. We are interested in human guided ontology alignment, where the alignment algorithm attempts to propose likely matches to the user. We feel that existing techniques can benefit by considering expression similarity along with their other similarity calculations when proposing possible like or related terms.

We propose a data driven approach to solving this problem, which uses the novel idea of web searching to capture relevant instances or data for the terms of interest. To compare two expressions, our algorithm first performs a search using two queries corresponding to the expressions. The results returned represent highly compact and highly relevant instances of the expression. Generally speaking, if two expressions are not related, their corresponding search results have very little in common. Alternatively, if the items are associated with each other, the texts must have certain commonality. This is based on the idea that if the two ontologies in question were used to annotate web search results, then the annotations between matching terms should have a high probability of overlapping.

The results of the web searches get processed by the SVD (Singular Value Decomposition) matrix technique [13]. This is done to change the actual number of each single word occurrence in a single document to its expected number of occurrences based on the relative importance of this word in the given document. The numeric result for semantic similarity is returned using the cosine measure between the vectors of expected word occurrences. For multiple word terms we perform a linear combination of the terms to generate a single vector representing the entire expression.

The advantage of using web search, specifically the Google search engine, is that Google has access to billions of indexed pages in various languages and on various topics. This means that the algorithm is not restricted to a single domain, a single language, and to static information that cannot be easily extended. All of these are vitally important for general ontology alignment. Also, by using only the title of search results along with the page description for a small number of results, we are able to use far less data than traditional Latent Semantic Analysis (LSA), as the data is highly relevant to the given expression. Moreover, by processing smaller amounts of information, our algorithm is able to perform these comparisons quickly, which is important for large ontologies. Finally, searches can also be guided to provide clues as to the relationship between the expressions being compared, such as synonymy versus antonymity.

1.1 Organization of the Paper

The remainder of the paper is organized as follows. In the next section we discuss the set of requirements for a good semantic matcher, which is based on the analysis of common types of term matching divergences. It is followed by a summary of the related work, considered in the following two directions: synonym matchers and ontology alignment algorithms. Afterwards, we formulate our algorithm, LSA-IR, and discuss particulars of its implementation. Next, we introduce the test data sets and follow this with the section describing the experimental results. In the latter, we show our tool's performance in comparison to other techniques, discuss the results, and examine the limitations of

our approach. We conclude with a short summary and a discussion of future research directions.

2 Developing Requirements

2.1 What Causes Alignment Problems?

There are many causes to alignment problems, and in general, it is a very difficult problem to solve. Ontologies can be developed by different groups of individuals and later those groups may want to consolidate their separately built ontologies. Separately built ontologies will most likely have terminology divergences as well as structural ones. For example, in [12] the authors attempt to align two anatomy ontologies, FMA and CRM. FMA contains approximately 59,000 concepts while CRM contains approximately 24,000. Even after the authors performed some basic normalization on the concept names (e.g. remove camel case and split concatenated words), there were only 1834 string matches.

Problems also develop between different versions of an ontology and these versions may later need to be compared or merged. Terminologies may have been enhanced or replaced between versions, making the merge or comparison difficult to perform.

In Table 1 we characterize some of the types of divergences or problems that may occur. This characterization was partly inspired by the divergences discussed in [22], which dealt with data divergences between different data sources where objects are represented by terse English phrases. These problems help drive our requirements which we discuss next.

Table 1. Characterization of term matching divergences

Type	**Example/Explanation**
Structural differences	Differences in hierarchy or internal structure
Missing information	Information contained in ontology not available in other.
Misspellings	
Dropped, transposed substitutions	MICROSOFT *vs* MICOSOFT, MICROSOFT *vs* MICORSOFT, MICROSOFT *vs* MIKROSOFT
Common mispellings	APPARENT *vs* APPARANT
Variant spellings	MODELLING *vs* MODELING
Phonetic spelling	THEIR *vs* THERE
Synonyms	DOG *vs* CANINE
Abbreviations	FEMALE *vs* FML
Acronyms	HTML *vs* HYPERTEXT MARKUP LANGUAGE
Spelling, punctuation, spacing and casing	COLOR *vs* COLOUR ISSN *vs* I.S.S.N PUBLISHED_BY *vs* PUBLISHED BY *vs* "Published by"
Prefixes	xyzHUMAN *vs* HUMAN
Root forms	VALVES *vs* VALVE, RECEIVE *vs* RECEPTION
Alpha/Numeric	PI *vs* 3.141592
Technical/Common	DOG *vs* CANIS FAMILIARIS
Word order	THESIS PHD *vs* PHD THESIS

2.2 Requirements

In Table 1 we characterize different types of data divergences that can occur between two related ontologies. We use these characterizations to help build a set of requirements that any expression comparison technique for terminology alignment must satisfy.

- **Multi-lingual.** Any expression comparison technique cannot be restricted to a single language.
- **Multi-domain.** Expression comparison cannot be restricted to a single domain, as this is far too limiting for general approaches. The technique must be able to potentially support any domain.
- **Multi-word expressions.** Must be able to support comparison between multiple word expressions.
- **Dynamic.** Must be dynamic, in that we do not allow it to be restricted to static data that cannot be easily extended.
- **Multiple-relation comparison.** Cannot restrict similarity comparison to simply synonyms, other relationships are also important.
- **Misspellings:** Must be able to handle the various types of misspellings described in the ontology divergence table.
- **General expressions.** Must be able to compare similarities between various types of expressions, *e.g.* alpha/numeric, non-noun phrases, abbreviations, acronyms, *etc.*

3 Related Work

In this section we briefly discuss related work. This discussion is partitioned into several topics that appear in separate subsections. In the first, we discuss work that has been conducted on the comparison of synonyms. In the second, we discuss existing ontology alignment techniques and discuss how our technique relates to existing approaches. Finally, we compare the set of existing synonym matching techniques to our list of requirements.

3.1 Synonym Matching Approaches

There have been various approaches applied to the problem of matching synonyms. Latent Semantic Analysis (LSA) [10], Roget Thesaurus [9], statistical techniques such as PMI-IR [23], product rule [24], and WordNet similarity measurements [9], [16] have all been applied to this problem. The LSA technique attempts to collect statistics about the relative frequency of a word and its neighboring words. It is based on the assumption that two words are similar if they have similar neighboring content words. One of the limitations of LSA is that it depends on preprocessing a large set of text to build a static representation of words and their relative frequencies. Extending this representation requires re-running the preprocessing. Thus, terms that do not appear in the preprocessed text cannot be later compared. We discuss this method in more detail in the description of our algorithm.

Similar to LSA, PMI-IR is based on co-occurrence of words, however, it computes this in a completely different manner. Rather than preprocessing documents and computing relative frequencies, it applies proximity based web searching and Pointwise Mutual Information (PMI). PMI comes from Information Theory, and it is a technique for calculating the similarity between two words or possibly a word and a document. In PMI-IR, web searching is applied to calculate statistics about the words in question. Two searches are carried out on the Altavista search engine and the number of $hits(query1)$ for the query is used in the calculation. For example, if we wanted to compare $term1$ to $term2$, PMI-IR would perform the following searches and calculation.

$$sim(term1, term2) = \frac{hits(term1 \text{ NEAR } term2 \text{ AND NOT } ((term1 \text{ OR } term2) \text{ NEAR “not”}))}{hits(term2 \text{ AND NOT } (term2 \text{ NEAR “not”}))}$$

The term NEAR is a keyword in the Altavista search engine that only reports sites where the two words separated by the NEAR keyword appear within 10 words of each other.

Similar to PMI-IR, another approach called LC-IR (local context-information retrieval) [8], makes use of the Altavista search engine. While PMI-IR measures similarity based on the proximity of the two words in question, LC-IR restricts searches to results where the two words are exactly adjacent to each other. By adding this restriction, the calculation and search queries are simplified and LC-IR slightly outperforms PMI-IR on the standard synonym data sets.

The product rule is a hybrid approach that uses several individual similarity modules and combines their results into a single calculation. For individual modules, it makes use of LSA, PMI-IR, synonym lists, and a Connector that uses Google summary pages to estimate stem-choice similarity. Each module is trained on a training set and a weight for that module is calculated based on the results from the training set. The weight represents the likelihood that the answer given by the module is correct and is used by the product rule when combining the individual calculations. On the TOEFL synonym test, the product rule has the best known result.

Thesauri are sometimes used for calculating semantic similarity. In the Roget Thesaurus work, paths between individual words are used to calculate similarity. The further apart the words, the smaller the similarity. Similar approaches have been applied to WordNet. These techniques are both limited to calculating synonymy, and WordNet primarily supports only noun to noun comparison. Also, with thesauri, many concepts that could appear in an ontology may not be present due to the thesauri language or domain. Moreover, alpha/numeric comparisons such as PI to 3.14159, are generally not supported.

3.2 Ontology Alignment

As in synonym comparison, there has been a variety of approaches used to automatically or semi-automatically perform ontology alignments. For example, Euzenat *et al.* discuss over 20 different algorithms/tools in [6]. Some of the most widely used methods are based on heuristic techniques.

Heuristics are generally applied at two different levels, the first being to compute syntactic similarity and the second is to measure structural similarity. Both Chimera [1] and PROMPT [14] use syntactic similarities to make suggestions to the user. They first run an ontology alignment algorithm that attempts to find exact matches on concept names, prefixes, suffixes, or word roots. They then use the user's feedback about the suggestions to make further suggestions based on structural similarities.

Structural similarity can be partitioned into two classes: internal and external structure [6]. Internal structural comparisons measure similarity between concept properties, such as cardinality, range, and symmetry. On the other hand, external structural comparisons attempt to find similarities in the `is_a` relationship structure between the two ontologies. Many hybrid approaches exist, such as QOM [5], which employ a large number of heuristics and use a weighted sum to normalize all similarity measurements into a single metric.

Another, less widely used approach is the so called instance-based or instance-level approach [2]. Here, concepts are compared based on their instances rather than their representation. An instance is an actual value of a concept, for example, an instance of a concept "Professor", would be an actual Professor, such as Dr. Donald Knuth. Concept similarity can then be measured by comparing shared instances. Another way to measure the similarity for an instance-based approach is to apply machine learning techniques to build classifiers for concepts. The Glue system is an example of this, it builds learning classifiers for concepts and then evaluates the joint probability distribution of the assigned instances [6].

The semantic similarity technique that we are proposing is more closely related to the instance-based approach to ontology alignment. However, as mentioned, ontology alignment is a very difficult problem and it is unlikely that a single approach is general enough to capture all possible matching scenarios. Tools like PROMPT could be combined or extended to use our proposed technique in order to improve the suggestions made to the user.

3.3 Requirements Table

In Table 2, we compare the previously discussed synonym matchers to our list of term matching requirements. Although each approach partially fulfills the requirements, none of the approaches completely satisfy every criteria that we established as important for semantic comparison in term alignment problem. Of course, this is not terribly surprising as none of these approaches were designed with these requirements in mind, they were designed to solve synonym tests. PMI-IR is the closest to satisfying our requirements, and could possibly be extended to meet the requirements, however the authors did not discuss multiple word or multiple types of comparison as this was not the focus of their work. Also, PMI-IR is not publically available so we could not verify whether this extension is possible. Altavista has also changed considerably over the last few years and PMI-IR may not work with the latest Altavista. Finally, on standard synonym tests, the method we developed (LSA-IR) significantly outperforms PMI-IR. LSA-IR is described in the next section.

Table 2. Requirement satisfaction for existing work

Technique	Multi-lingual	Multi-domain	Multi-word	Dynamic	Multi-comparison	Misspellings	General
LSA	√	√	√	×	√	×	√
Roget	×	√	×	×	×	×	×
PMI-IR	√	√	*	√	*	√	√
LC-IR	√	√	×	√	×	√	√
WordNet Sim	×	√	×	×	×	×	×
Product rule	√	×	√	√	×	√	√
LSA-IR	√	√	√	√	√	√	√

√ - satisfied, × - not satisfied, * - could possibly be extended, but not discussed

4 The Algorithm

Our algorithm combines web searching and LSA, so following the tradition of PMI-IR and LC-IR, we refer to our technique as LSA-IR. By adapting LSA to use web search results rather than a static collection of text, the method is dynamic and not limited to preprocessed data. Also, since the number of results is kept small, and therefore the corresponding text to process is relatively small, LSA can be performed quickly and immediately. Moreover, limiting LSA to processing smaller portions of text helps improve LSA accuracy for certain applications. The authors of [15] introduced a two stage LSA algorithm for finding word aliases. The authors found that when LSA is applied to a general text collection, the most similar words to any given word often do not correspond to proper aliases. In the two stage approach, the authors perform LSA a second time, this time instead of using the general text documents, the documents are constructed by extracting small windows of text that surround the words of interest. That is, the second stage only considers the local context around occurrences of words rather than the global context with respect to the entire document. This extension to traditional LSA greatly improved the performance for extracting word aliases. Similarly, in our tool, since we only process link titles and Google's content summary, we are essentially considering the local context of the word.

The input for our algorithm is a pair of ASCII character strings $String1$ and $String2$ along with optional information that indicates the type of comparison to be made. An example of the type of comparison may be to find the similarity between strings interpreting them as synonyms, antonyms, or homonyms. Such a comparison is guided by a user who points the search in a specific direction. For instance, when one looks for a synonym dependence between two expressions, the query entries are "$String1$ synonym" and "$String2$ synonym". Similarly, with antonym dependence, the query entries are "$String1$ antonym" and "$String2$ antonym". Both the "synonym" and "antonym" context help guide the search when the type of comparison is known.

As a more concrete example, consider the situation where we want to compare "dog" to "canine" as synonyms. We can guide the type of search results by adding "synonym" to the search queries, thus searching for pages containing "dog synonym" and "canine synonym". This context acts as a heuristic which restricts the search results to certain types of websites. Our technique is based on the assumption that similar expressions should yield similar search results, and we calculate this similarity by inspecting the local context as generated by the results.

Moreover, other restrictions are possible, such as restricting the search to a specific web site, ignoring web pages with certain words, or forcing exact phrase matches. While we did not experiment with the other guidelines for the search, it is in our future plans to do so. If no comparison type is specified we assume a general association comparison must be made and prepare the search queries as pure input strings "$String1$" and "$String2$".

The next step is using a web search engine to search for the texts related to the prepared search strings. Search engines have access to billions of indexed web pages, and this provides a very rich data set to work with. Moreover, almost any concept can be found in a variety of different contexts and possible meanings. Our choice of the search engine is Google, which is based partly on the free availability of their search API and also due to Google currently being the most widely used search engine [20]. The result of the search is truncated at the first K hits with all being treated as equal. As discussed in the next section, in our experiments we use $K = 75$. From our initial experiments, this number of search results provides enough generality to find how the terms are being used, and yet is small enough to allow for fast data analysis.

Each of the K search results is treated as a separate document and the total number of documents in both searches equals $2K$. We next stem the words in all $2K$ documents using Porter's Java implementation [17] and delete the most popular words carrying minimal information (stop words). Our list of stop words consists of around 400 words and includes such words as "the", "a", "of", "in", *etc.* The remaining words are used to form a matrix in which there are $2K$ columns that correspond to each of the returned documents and N strings with each corresponding to a single stemmed word that appears at least once in the set of documents. The column vectors are then normalized, meaning we treat all the documents as equal.

The matrix element $m_{i,j}$, where $1 \leq i \leq N$ and $1 \leq j \leq 2K$ is the number of occurrences of the i-th word in the j-th document of the first web search if $j \leq K$ and in the $(j - K)$-th document of the second web search when $j > K$. We weight each element, $m_{i,j}$, as $\log(m_{i,j} + 1)$. This is a common practice in information processing applications [3]. We next process this matrix using the *Singular Value Decomposition* (SVD).

SVD [10], [13] is a popular and efficient technique for text analysis. Its input is a matrix M of word occurrences in the documents constructed as discussed above. This matrix gets decomposed into the product of three matrices $M = U\ D\ V^T$, where matrices U and V have orthogonal columns such that $U^T\ U = V\ V^T = I$ and D is a diagonal matrix. Keeping a percentage of the largest diagonal elements in matrix D transforms it to D'. Next, we construct the matrix M' as a product $M' := U\ D'\ V^T$. Matrix M' differs from the original matrix M. A nice property of M' is that an element $m'_{i,j}$ estimates the number of occurrences of the i-th word in the j-th document. The optimal number of dimensions that must be kept for the best performance is unknown and must be determined through experiments. In our evaluations we chose to keep 70% of the largest coefficients in the diagonal matrix. The details of how we arrived at this percentage are explained in the next section.

Once the SVD transformation is completed, we measure the similarity between the two input strings $String1$ and $String2$. In the case of single word items we compare

the vector of the expected word occurrences of $String1$ with the vector of the expected occurrences of $String2$. The vectors are constructed using matrix M' by considering the strings corresponding to the words $String1$ and $String2$. We use the cosine measure defined as follows

$$\cos(\overline{a}, \overline{b}) := \frac{(\overline{a} \cdot \overline{b})}{\|\overline{a}\| * \|\overline{b}\|}.$$

The cosine measurement defines if the two vectors point in the same direction (meaning if the items are equally used in the similar contexts) without caring which of the two vectors is longer (meaning which of the two search strings is a more commonly used expression/concept/term). When the cosine is close to zero this means that the two items are rarely used in the same contexts and thus we consider such items unrelated.

In the case when the strings $String1$ and $String2$ are composed with multiple words we first stem each word and delete common words from both strings. This results in the construction of strings $String1'$ and $String2'$. If a word appears in both $String1'$ and $String2'$, we apply a weight to this word in order to reduce its importance. For example, if we are interested in the similarity between "computer network" and "computer monitor", we care more about the words "network" and "monitor", than "computer" since this term appears in both strings. Then, we add the vectors corresponding to the expected occurrences of all words in $String1'$, and do the same with the words from $String2'$. Finally, the result is returned as a cosine between these summary vectors.

5 Data Sets

In this section we describe the data sets used in our experiments. The first data set consists of two biomedical ontologies, the NCI (National Cancer Institute) thesaurus and SNOMED-CT (The Systematized Nomenclature of Medicine). The second data set consists of three popular synonym tests that have been used by previous researchers.

5.1 Ontology Data

As mentioned, the ontology data comes from the NCI thesaurus and SNOMED-CT. The authors of [19] used an automated heuristical approach to map ontological terms from the NCI thesaurus and SNOMED-CT to unstructured keyword data within the Stanford Tissue Microarray Database (TMAD). The authors were interested in being able to represent the TMAD data in a structured way, such as in an ontology. They were able to successfully map 80% of the TMAD unstructured annotations. The authors mention that the mapping from the NCI thesaurus to TMAD and SNOMED-CT to TMAD could be used to align terms from the NCI thesaurus to SNOMED-CT.

Using the data generated in [19], we manually extracted 60 expression pairs. Each pair consists of one expression from the NCI thesaurus and its corresponding equivalent expression from SNOMED-CT. The expression pairs were classified based on the types of divergence discussed in Table 1. The primary type of divergence is "Missing Information", that is, one ontology used a more descriptive expression for the term but the two expressions had some overlap in the terms used. The data set also contains abbreviation matches, such as "Malignant_Peripheral_Nerve_Sheath_Tumor" to "MPNST",

synonym matches, such as "Neoplasm" to "Tumor", spelling variations, punctuation changes, and changes in the word order. There are no exact string matches.

In the test, we first normalize the expressions by downcasting all characters, replace underscores with a space, and trim whitespace. We then compute similarities between all pairs of expressions, that is, we compute all 60×60 $(= 3600)$ expression similarities. Afterwards, we greedily select matches by pairing off the two expressions with the highest similarity, then we eliminate these two expressions from further consideration, and continue until all expressions have a match.

5.2 Synonym Data Sets

Although we did not develop LSA-IR with the specific purpose of synonym mining, we chose to evaluate it against some standard synonym data sets in order to compare with existing methods. The data sets used were 80 questions from TOEFL (Test of English as a Foreign Language) [10], 50 questions from ESL (English as a Second Language) [21], and 300 questions from RDWP (Reader's Digest Word Power Game) [4].

The TOEFL and RDWP tests have the same structure. For each question they consist of a single word and four possible matches. One of the four possibilities is a synonym of the given word. The ESL test is slightly different, each question consists of a sentence where the word of interest is used in some context, and there are also four possible matches.

For both the TOEFL and RDWP questions, we compare the given word to all four possibilities and choose the word that yields the highest similarity value. The Google search was guided by using a context of "synonym", that is, to compare "*String1*" to "*String2*", we perform the following two Google searches, "*String1* synonym" and "*String2* synonym". The similarity is returned as the median score between this search and searches "*String1*" and "*String2*".

With the ESL data set, we first preprocess the questions to extract a single word context. We do this by first removing stop words from the questions, and then for each word w_i near the word of interest w, we compute $sim(w, w_i)$. Here *near* is any word within three words of w. The context becomes the word w_i with the highest similarity value to w. As an example, consider the ESL question "Don't forget to vacuum the [rug] before they come home." The word of interest, $w =$ "rug", and the term "vacuum" has the highest degree of similarity to "rug" according to LSA-IR, thus this becomes the context. For each of the four possible synonyms, we compute the similarity between the word of interest and the possible answer using the "synonym" context added together the similarity between the extracted context with the word of interest and the possible answer. The answer with the highest similarity is chosen as the correct synonym.

6 Algorithm Parameters

In the discussion of the algorithm we introduced two constants, the number of search queries K, and the percentage of the diagonal elements of the matrix D that we keep. We must now choose these parameters for optimal performance. Our intention is to keep the number of search results as small as possible such that the amount of processing is kept small, yet yields a good performance. The smaller K is, the less the interaction with

the web, and the less data that needs to be processed afterwards. Secondly, we adjust the number of diagonal elements to be kept. In adjusting the number of dimensions, we are primarily concerned with the quality of the results. This is because the percentage of diagonal elements kept does not greatly affect the speed of our algorithm.

6.1 Setting the Parameters

We conducted an experiment using half of the TOEFL synonym data to find the best values for K and P. Using $10 \leq K \leq 120$ with an increment of 10 and $0.4 \leq P \leq 0.8$ with an increment of 0.05, we evaluated the TOEFL results for every $\{K,\ P\}$ combination. The results are shown in Figure 1. The height map represents the accuracy or percentage evaluated to be correct. Results fluctuate from a low of 0.7 to a high of 0.85. The high of 0.85 was achieved 3 times, with $K = 70$, $P = 0.7$ and with $K = 80$, $P = 0.7$ and $P = 0.75$. Based on this result, for all further experiments, we fixed $K = 75$ and $P = 0.7$.

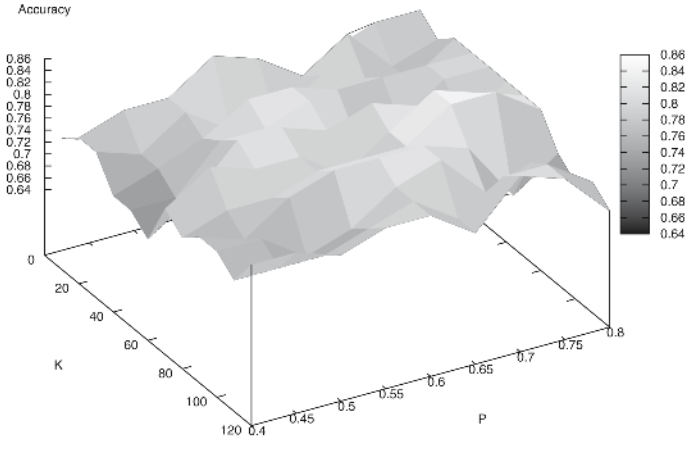


Fig. 1. 3D surface diagram of parameter setting experiment

7 Experiment Results

In the following two sub-sections we illustrate the results of the application of our tool to the data sets discussed previously. When applicable, we compare the results of these experiments with that of other techniques.

7.1 Ontology Test

Table 3 describes the results from our experiment of matching terms from the NCI thesaurus to SNOMED-CT. The "Baseline" represents the results one would expect from random guessing. We included results computed using standard syntactical matching techniques, Levenstein distance [11] and substring distance [6]. We used the same normalization and greedy approach for both syntactical measurements. As previously

mentioned, a large portion of the aligned terms were classified as type "Missing information". Thus, a large number of the expressions contains overlapping substrings, which is most likely why the substring distance measurement achieved greater than 50% accuracy. However, both syntactical measurements were unable to match more complex (semantic) equivalences such as synonymity, so unsurprisingly LSA-IR significantly outperformed syntactical measurements. Moreover, even the incorrect matches computed by LSA-IR made semantic sense, that is, the matched terms were related, but they were not the most related as determined by our data set.

Table 3. Ontology matching experiment results

Technique	NCI to SNOMED-CT
Baseline	$< 2.00\%$
Levenstein distance	$\frac{29}{60} = 48.33\%$
Substring distance	$\frac{32}{60} = 53.33\%$
LSA-IR	$\frac{53}{60} = 88.33\%$

7.2 Synonym Tests

In this experiment we test our technique on some standard synonym data sets. The results are shown in Table 4. Again, the "Baseline" represents the results one would expect from random guessing.

LSA-IR has an overall result of 85.10%, which is the highest overall accuracy. Unfortunately the Product rule, which essentially solved TOEFL, does not have published results for ESL and RDWP. The Product rule would most likely outperform LSA-IR on these two data sets as it is a hybrid approach as opposed to a single technique like LSA-IR. However, we suspect that LSA-IR more efficently computes similarities than the Product rule since the Product rule combines four different computational approaches.

Table 4. Synonym experiment results

Technique	TOEFL	ESL	RDWP	Total
Baseline	$\frac{20}{80} = 25.00\%$	$\frac{12.5}{50} = 25.00\%$	$\frac{75}{300} = 25.00\%$	25.00%
LSA	$\frac{51.5}{80} = 64.40\%$	-	-	-
Product rule	$\frac{78}{80} = 97.50\%$	-	-	-
PMI-IR	$\frac{59}{80} = 73.75\%$	$\frac{37}{50} = 74.00\%$	$\frac{216.83}{300} = 72.30\%$	72.80%
Roget's Thesaurus	$\frac{63}{80} = 78.75\%$	$\frac{41}{50} = 82.00\%$	$\frac{223}{300} = 74.30\%$	76.00%
LC-IR	$\frac{65}{80} = 81.30\%$	$\frac{39}{50} = 78.00\%$	$\frac{224.33}{300} = 74.80\%$	76.40%
LSA-IR	$\frac{71}{80} = 88.75\%$	$\frac{39}{50} = 78.00\%$	$\frac{256}{300} = 85.33\%$	85.10%

7.3 Recommendation Test

In our final test, we used the RDWP synonym data set again, but we randomly removed 50 of the correct answers and replaced them with a random incorrect answer. Moreover, this time, LSA-IR only made a match recommendation if the similarity between the

question word and its most similar answer was above a threshold value of ω. In our test, we experimented with a variety of ω values.

This test becomes a more complicated version of the original, as now LSA-IR must not only determine the correct synonym, but it must also determine if a synonym for the word exists at all. This is an important test because in many matching problems, like ontology alignment, it is very likely that not every term has an appropriate match. We do not wish to make bad suggestions just because it is the only suggestion we can make.

We used the standard Information Retrieval evaluation metrics, precision and recall, as defined in [5] for ontology alignment.

Precision is the measurement of correctly retrieved mappings in proportion to the total number of found mappings.

$$p = \frac{\#correct\_found\_mappings}{\#found\_mappings}$$

Recall is the measurement of correctly retrieved mappings in proportion to existing mappings in the data set.

$$r = \frac{\#correct\_found\_mappings}{\#existing\_mappings}$$

Finally, as is typically done in Information Retrieval, we combine the two metrics into an f-measure.

$$f = \frac{2pr}{p + r}$$

The results are shown in Table 5. We varied ω from 0.1 to 0.4 and recorded the precision and recall for each of these values. As a baseline, we used $\omega = 0.0$. With this threshold, LSA-IR always chooses the term with the highest similarity as the synonym regardless of the value of this measurement. In this case, the recall should be maximized while the precision should be minimized, which the experimental results confirm. The best overall performance, based on the f-measure, occurred with $\omega = 0.2$. With this value, we achieved a very high precision and still correctly found 165 results, which is 83.76% of our original 197 correctly found results. With higher ω values, as expected, our precision continues to climb, however our recall value is much too low.

Table 5. Recommendation test results

ω	**Precision**	**Recall**	f**-measure**
Baseline (0.0)	$\frac{197}{300} = 65.67\%$	$\frac{197}{250} = 78.80\%$	0.7164
0.1	$\frac{197}{299} = 65.89\%$	$\frac{197}{250} = 78.8\%$	0.7177
0.2	$\frac{165}{183} = 90.16\%$	$\frac{165}{250} = 66.00\%$	0.7621
0.3	$\frac{73}{76} = 96.05\%$	$\frac{73}{250} = 29.20\%$	0.4478
0.4	$\frac{27}{28} = 96.43\%$	$\frac{27}{250} = 10.80\%$	0.1942

Figure 2 displays the plot of the f-measure versus the different ω values.

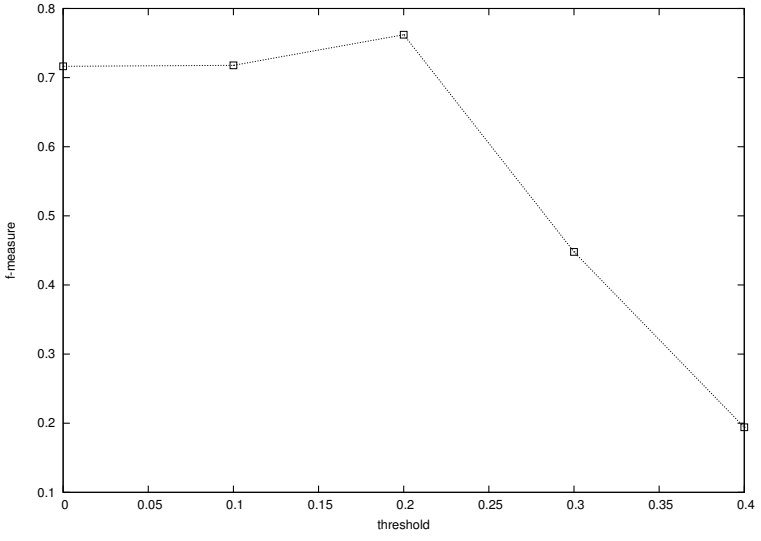


Fig. 2. Plot of the f-measure versus threshold value ($0.0 \leq \omega \leq 0.4$)

7.4 Discussion of Results

We are primarily interested in user-guided ontology alignment, and the results of semantic comparison of the terms from the NCI thesaurus to SNOMED-CT ontologies indicate that LSA-IR is potentially beneficial for computing suggestions during the ontology alignment process. In particular, for ontology alignment it would be extremely helpful to users to be able to suggest matches such as "Olfactory_Neuroblastoma" to "Esthesioneuroblastoma", which many syntactical matchers are not able to recommend.

Although synonym matching is not the primary purpose of our tool, the results of the synonym comparison experiment helped verify the usefulness of the tool for mining strictly synonyms. As discussed, in comparison to previous approaches, LSA-IR had the best overall performance.

In the final experiment, we wanted to test the precision and recall of LSA-IR. To our knowledge, this is the first time this kind of test has been used to evaluate a semantic comparator. Previous research has not reported precision results. However, we feel this is a critical test if our tool is to be used in applications like user-guided ontology alignment. We do not want to make inappropriate suggestions to the user simply because it is the only suggestion we can make. There must be a balance between precision and recall, as we do not want to suggest too many incorrect matches and we also do not want to exclude too many possibly correct matches. Our results indicate that our method is able to correctly identify matches with high precision, and still produce a large number of suggested matches. The results from each of these preliminary experiments are very encouraging and indicate to that we should continue experimenting with this approach.

7.5 Limitations

LSA-IR suffers from several potential limitations. Firstly, since the method depends on the Google search index, it inherits certain limitations that are intrinsic with Google. For example, although Google has billions of pages indexed, it cannot possibly index everything, thus, certain word contexts may be missing or not present within the first 75 Google results. Moreover, we are also limited by the quality of the search results returned by Google. Ideally, we would like to be able to extract the local context of the words of interest from each search result, but performing this extraction from each page would be too costly. In order to bypass this computational cost, we rely on the Google summary and page title, but the proper context of the words of interest may not be present in this result information.

8 Conclusions and Future Work

In this paper we presented an automated, data-driven tool for semantic comparison of phrases. Our tool extends traditional LSA by adapting it to work with web-based search results. By using web search results, we are able to perform comparisons between many types of expressions that previously has been difficult to perform using a single tool. Previous web-based comparison tools have concentrated on computing statistics based on the number of search results returned, whereas with LSA-IR, we concentrated on actually analyzing the text of these returned results.

We demonstrated the usefulness of our approach by conducting experiments on several different data sets. Specifically, we showed how LSA-IR was useful for matching equivalent expressions in two biomedical ontologies, the NCI thesaurus and SNOMED-CT. We also tested the tool on standard synonym benchmarks that had been used by previous researchers and although synonym matching was not our primary goal, LSA-IR outperformed earlier tools that were built specifically to discover synonymity. Finally, we conducted an experiment to test how well LSA-IR can not only choose a correct synonym from a list, but also how well it can determine whether a synonym exists at all.

In the future, we plan to incorporate our semantic comparator into a user-guided ontology alignment tool. Further research is required on how to distinguish between `is_a` and `has_a` relationships, as simply guiding the search via a “is a” or “has a” context will most likely not be enough to distinguish the two concepts.

We also believe LSA-IR could be useful for automated data annotation between ontologies and unstructured data, similar to how it is done in [19]. We are interested in extending our tool to help automated keyword extraction for scientific papers. Many existing approaches to this problem use word frequency counts, however, authors of research papers often use semantically equivalent expressions when discussing a single topic or idea, thus the frequency counts may not be accurate.

Acknowledgements

Firstly, we would like to thank Nigam Shah, Daniel L. Rubin, Kaustubh S. Supekar and Mark A. Musen for use of their ontology matching data. We would also like to

thank Thomas Landauer and the LSA group at the University of Colorado at Boulder for providing us with the TOEFL test questions.

This work was supported in part by the National Center for Biomedical Ontology, under roadmap-initiative grant U54 HG004028 from the National Institutes of Health, and by a PDF grant from the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

1. McGuinness D., Rice J. Fikes R., and Wilder S. An environment for merging and testing large ontologies. 2000.
2. AnHai Doan, Pedro Domingos, and Alon Halevy. Learning to match the schemas of data sources: A multistrategy approach. *Machine Learning*, 50(3).
3. S. Dumais. Enhancing performance in latent semantic indexing. Technical report, 1990.
4. M. Lewis (ed.). Readers digest. 158(932, 934, 935, 936, 937, 938, 939, 940), 159(944, 948), 2000-2001.
5. M. Ehrig and S. Staab. Qom - quick ontology mapping. In *Proc. of the Third International Semantic Web Conference (ISWC2004)*, 2004.
6. Jérôme Euzenat, Thanh Le Bach, Paolo Bouquet Jesús Barrasa, Jan De Bo, Rose Dieng-Kuntz, Marc Ehrig, Manfred Hauswirth, Mustafa Jarrar, Rubén Lara, Diana Maynard, Amedeo Napoli, Giorgos Stamou, Heiner Stuckenschmidt, Pavel Shvaiko, Sergio Tessaris, Sven Van Acker, and Ilya Zaihrayeu. State of the art on ontology. deliverable d2.2.3, 2004.
7. T. R. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):23–28, 1993.
8. Derrik Higgins. Which statistics reflect semantics? rethinking synonymy and word similarity. In *International Conference on Linguistic Evidence*, 2004.
9. M. Jarmasz and S. Szpakowicz. Roget's thesaurus and semantic similarity. In *Proceedings of Conference on Recent Advances in Natural Language Processing (RANLP 2003)*, pages 212–219, September 2003.
10. T.K. Landauer and S.T. Dumais. A solution to plato's problem: The latent semantic analysis: Theory of the acquisition, induction, and representation of knowledge. *Psychological Review*, 104:211–240, 1997.
11. I. Levenstein. Binary codes capable of correction deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8):707–710, 1966.
12. Peter Mork and Philip A. Bernstein. Adapting a generic match algorithm to align ontologies of human anatomy. In *20th International Conference on Data Engineering*. IEEE, 2004.
13. J. C. Nash. *Compact Numerical Methods for Computers: Linear Algebra and Function Minimization*. Adam Hilger, Bristol, England, 2nd edition, 1990.
14. N. Noy and M. Musen. The prompt suite: Interactive tools for ontology merging and mapping. Technical report.
15. Tim Oates, Vinay Bhat, Vishal Shanbhag, and Charles Nicholas. Using latent semantic analysis to find different names for the same entity in free text. In *Proceedings of the 4th international workshop on Web information and data management*, 2002.
16. T. Pedersen, S. Patwardhan, and J. Michelizzi. Wordnet::similarity - measuring the relatedness of concepts, 2004.
17. M. F. Porter. Java implementation of porter's algorithm, 2000.
18. S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, Upper Saddle River, New Jersey, 1995.

19. Nigam H. Shah, Daniel L. Rubin, Kaustubh S. Supekar, and Mark A. Musen. Ontology-based annotation and query of tissue microarray data. In *AMIA 2006 (under review)*, 2006.
20. Danny Sullivan. Nielsen netratings search engine ratings. http://searchenginewatch.com/showPage.html?page=2156451, 2006. Last visited: 08-15-2006.
21. D. Tatsuki. Basic 2000 words - synonym match 1, 1998.
22. David R. Throop. Reconciler: Matching terse english phrases, 2004.
23. P. Turney. Mining the web for synonyms: Pmi-ir versus lsa on toefl. In *Proceedings of the Twelfth European Conference on Machine Learning (ECML-2001)*, pages 491–502, September 2001.
24. P. Turney, M.L. Littman, J. Bigham, and V. Shnayder. Combining independent modules to solve multiple-choice synonym and analogy problems. In *Proceedings of International Conference on Recent Advances in Natural Language Processing (RANLP-03)*, September 2003.

A Web-Based Novel Term Similarity Framework for Ontology Learning

Seokkyung Chung[1,⋆], Jongeun Jun[2,⋆⋆], and Dennis McLeod[2]

[1] Yahoo! Inc., 2821 Mission College Blvd, Santa Clara, CA 95054, USA
[2] Department of Computer Science, University of Southern California, Los Angeles, CA 90089, USA
schung@yahoo-inc.com, {jongeunj, mcleod}@pollux.usc.edu

Abstract. Given that pairwise similarity computations are essential in ontology learning and data mining, we propose a similarity framework that is based on a conventional Web search engine. There are two main aspects that we can benefit from utilizing a Web search engine. First, we can obtain the freshest content for each term that represents the up-to-date knowledge on the term. This is particularly useful for dynamic ontology management in that ontologies must evolve with time as new concepts or terms appear. Second, in comparison with the approaches that use the certain amount of crawled Web documents as corpus, our method is less sensitive to the problem of data sparseness because we access as much content as possible using a search engine. At the core of our proposed methodology, we present two different measures for similarity computation, a mutual information based and a feature-based metric. Moreover, we show how the proposed metrics can be utilized for modifying existing ontologies. Finally, we compare the extracted similarity relations with semantic similarity using WordNet. Experimental results show that our method can extract topical relations between terms that are not present in conventional concept-based ontologies.

1 Introduction

With the rapid growth of the World Wide Web, Internet users are now experiencing overwhelming quantities of online information. Since manually analyzing data becomes nearly impossible, the analysis would be performed by intelligent information management techniques to fulfill users' information requests quickly.

Representation and extraction of semantic meanings from information contents is essential in intelligent information management. This issue has been investigated in diverse research disciplines including artificial intelligence, data mining, information retrieval, natural language processing, etc. One of the widely used approaches to addressing this problem is to exploit ontologies. For example, when users use irrelevant keywords (due to their vague information needs or

⋆ This research was conducted when the author was at University of Southern California.

⋆⋆ To whom correspondence should be addressed.

R. Meersman, Z. Tari et al. (Eds.): OTM 2006, LNCS 4275, pp. 1092–1109, 2006.

unfamiliarity with the domain of interests), query expansion based on ontologies can improve retrieval accuracy by providing an intelligent information selection.

A knowledge acquisition problem (i.e., how to build ontologies) is one of the main bottlenecks in ontology-based approaches. Although there exist handcrafted ontologies such as WordNet [15] or CYC [11], significant amounts of domain-specific terms (e.g., scientific or engineering terms) or neology are not present in general-purpose ontologies. Thus, it is essential to build ontologies that can characterize given applications.

However, although ontology-authoring tools have been developed in the past decades [19,27], constructing ontologies by hand whenever new domains are encountered needs significant amount of time and efforts. Additionally, since ontologies must evolve with time as new concepts or terms appear, it is essential to maintain existing ontologies up-to-date. Therefore, ontology learning, which is a process of integrating knowledge acquisition with data mining, becomes a must. Consequently, a knowledge expert can efficiently build and maintain domain ontologies with the support of ontology learning. Given that computation of the similarity between terms is at the core of ontology learning, we focus our attentions on computing similarity between terms.

As the Web continues to grow as a vehicle for the distribution of information, the massive amounts of useful information can be found on the Web. Given this wide availability of knowledge on the Web, we present WebSim (Web-based Similarity framework), whose feature extraction and similarity model is based on a conventional Web search engine. The proposed approach takes advantage of two main aspects of the Web search engine technology. First, as many thousands of Web pages are published daily on the Web, the Web reflects and characterizes current trend of knowledge. Thus, for each term, we can obtain the freshest content that represents the up-to-date knowledge on the term. This is particularly useful for dynamic ontology management in that ontologies must evolve with time as new concepts or terms appear. Second, because we access as much content as possible using search engines, our method is less sensitive to the problem of data sparseness. Although previous text mining crawls large amounts of Web pages for feature extraction, since crawled one is a small subset of the entire Web contents, it still suffers from a data sparseness problem.

At the core of WebSim, we present two similarity metrics, an information-theoretic metric and a feature-based metric. The former one is a mutual information based measure that utilizes the number of Web pages associating with each term. In contrast, the latter one extracts relevant features for each term, and performs similarity computation based on the extracted features. With the feature-based metric, we present how to deal with ambiguous terms for the similarity computation. We also show how ontologies can be modified with WebSim.

One of the main problems in concept-based ontologies is that topically related concepts and terms are not explicitly linked. For example, consider the Sports domain ontology that we developed in our previous work [9]. In this ontology, "Kobe Bryant", who is an NBA basketball player, is related with terms/concepts in Sports domain. However, for the purpose of query expansion, "Kobe Bryant"

also needs to be connected with a "court trial" concept if a user keeps "Kobe Bryant court trial" in mind. Therefore, it is essential to provide explicit links between topically related concepts/terms. Thus, conventional ontologies have a limitation in supporting a topical search. To address this problem, we also demonstrate how topical relations are generated using WebSim, and compare WebSim with semantic similarity in WordNet. In sum, the purpose of this research is to move one step forward to achieving the development of a novel similarity model that can be utilized for any ontology learning framework.

The remainder of this paper is organized as follows. In Section 2, we briefly review the related work, and highlight the strengths and weaknesses of previous work in comparison with ours. In Section 3, we present an information-theoretic similarity measure. In Section 4, we explain our feature extraction algorithm and explore how to compute similarity between terms based on the extracted features. Section 5 explains how WebSim can be utilized for ontology modification. In Section 6, we discuss the characteristics of WebSim by relating general-purpose ontologies. In Section 7, we present applications to which WebSim is applicable. Finally, we conclude the paper and provide our future plan in Section 8.

2 Related Work

The computation of the similarity between terms is at the core of ontology learning. There have been many attempts to determine similar term pairs from text corpora. One of the widely used approaches in similarity computation is based on distributional hypothesis [6,21]. That is, if terms occur in a similar context, then they tend to have similar meanings. The context for a term t_i can be defined in diverse ways. For example, it can be represented by co-occurrence of words within grammatical relationships (e.g., verbs that take t_i as a subject or object, adjectives that modifies t_i, etc), or co-occurring words with t_i in a certain length of a window. Each context is referred to as features of a term.

Recently, there have been research efforts for building ontologies automatically [16,14,26,1,18,7,25]. In order to obtain context, they usually utilizes certain amount of crawled Web documents as corpus. Our approach is different from the previous work in that a Web search engine is employed to exploit the full content of the Web. Consequently, rather than relying on a small subset of the Web, we can access as much content as possible (depending on how many documents a search engine crawler can index). Thus, our method is less sensitive to the problem of data sparseness. In addition, our feature extraction methodology is different from other approaches in that the context of terms are defined by a set of highly relevant documents returned by a search engine. Note that our research is complementary to the previous ontology learning efforts because the extracted features or term similarity can be utilized for any ontology learning framework. Therefore, WebSim is expected to be a key enabling technique for the tasks where pairwise similarity computations play a central role.

Table 1. Notations used in this paper

Symbol	Definition
t_i	An i-th term
$df(t_i)$	A total number of Web pages that are matched with a query t_i
N	A total number of Web pages that a search engine indexes
D_i	A set of top Web pages (returned by a search engine) for t_i
f_{ij}	A j-th feature of a term t_i
d_k	A k-th document returned by a search engine
l_k	The document length of d_k
$freq_{ijk}$	Term frequency of a feature f_{ij} in d_k
tf_{ijk}	Normalized term frequency of a feature f_{ij} in d_k
v_i	A vector for i-th term
N_i	The size of D_i
n_{ij}	The number of documents in D_i where f_{ij} occurs at least once
w_{ij}	The weight of f_{ij}
$IC(t_i)$	The information content of t_i
$prob(t_i)$	The concept probability of t_i
$count(t_i)$	The term frequency of t_i on corpus
$concept\_freq(t_i)$	The concept frequency of t_i on corpus
C	The size of corpus

3 WebSim: A Simple Mutual Information Approach

In this Section, we present a simple, yet powerful similarity metric. The measure is referred to as an MI-based (Mutual Information) WebSim. Table 1 illustrates the notations that will be used throughout this paper.

The underlying assumption behind the MI-based WebSim is that two terms co-occur frequently if they are similar to each other. Mutual information is an information-theoretic metric that quantifies *relatedness* between two words. The mutual information between t_i and t_j is defined as follows:

$$MI(t_i, t_j) = log \frac{p(t_i, t_j)}{p(t_i) \times p(t_j)} \tag{1}$$

The higher value in $MI(t_i, t_j)$ implies the stronger association between t_i and t_j.

Mutual information has been widely used in previous text mining research as a criteria for measuring term association. The probability is usually defined by term frequency divided by the total number of terms observed in corpus. However, this probability is restricted by the size of corpus. In particular, if a term is a new coined one, then it suffers from a data sparseness problem. To address this problem, WebSim utilizes a search engine to estimate the probability approximately. Figure 1 illustrates the idea. A FE (Front-end) scraper sends a query to a Web search engine, and extracts the number of documents that a

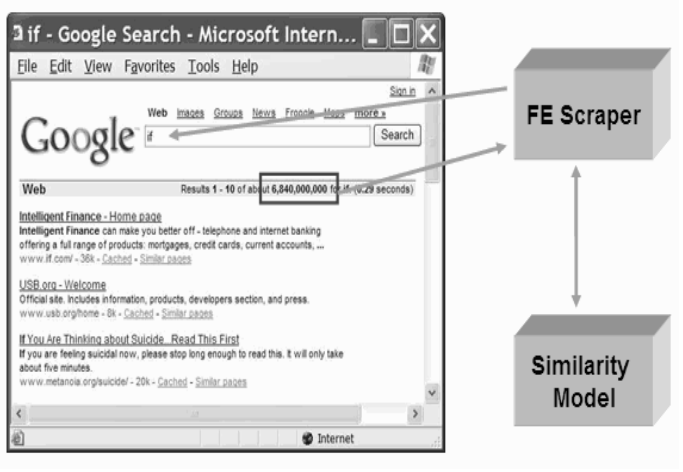


Fig. 1. Overview of an MI-based WebSim

query is matched with [1]. In order to estimate the total number of documents a search engine indexes, based on the fact that most search engines supports a boolean query, the number of documents for two different queries ("t_i" and "NOT t_i") are summed up. Thus, $p(t_i)$ is defined as the number of documents returned by a search engine for a query t_i ($df(t_i)$) divided by total number of document indexed by a search engine (N).

$$p(t_i) = \frac{df(t_i)}{N} \tag{2}$$

Consequently, in contrast to previous text mining approaches, WebSim uses the different notion of $p(t_i)$.

Most Web search engines provide advanced search features in that a user can specify how a query is matched with the page. That is, a user can retrieve the Web pages where a query is matched with title of the page (M_1), text of the page (M_2), URL of the page (M_3), links to the page (M_4), or anywhere in the page (M_5). Although it is worthwhile to investigate how different matching affects the accuracy of WebSim, this is beyond the scope of this paper. However, we briefly compare M_1 and M_5 in Table 2.

Table 2 presents sample results. As shown, in most cases (Type 1), WebSim captures similarity between related term pairs fairly well using both M_1 and M_5. WebSim with M_1 is better than WebSim with M_5 in some cases (Type 2) such as *DAML-OIL*, or *notebook-computer*. Moreover, M_1 is able to adjust

[1] Although Google is used in this paper, because most search engines display the total number of documents that are matched with a query, WebSim can use other search engines as well. It is also worthwhile to investigate how different search engines affect WebSim, but out of scope in this paper.

Table 2. Mutual information for sample term pairs

Type	t_i	t_j	M_5	M_1
1	Natural Language Processing	NLP	7.71	7.31
	Self Organizing Maps	SOM	6.32	3.59
	Artificial Intelligence	AI	5.35	8.04
	Genetic Algorithm	Evolutionary Computation	8.60	8.47
	Data Mining	Clustering	4.18	4.06
	Data Mining	Knowledge Discovery	6.85	12.5
	Text Mining	Ontology	4.40	3.62
	Text Mining	ODBASE	5.33	7.12
	Text Mining	ODBASE	5.33	7.12
	Text Mining	Bioinformatics	4.87	4.19
	Computer Security	Firewalls	3.41	3.31
	Clustering	Classification	4.49	5.34
	Classification	Neural Networks	3.92	6.49
	Machine Learning	Text Mining	6.57	3.77
	Semantic Web	Ontology	5.39	7.33
	Semantic Web	DAML	5.86	4.87
	Bioinformatics	Computational Biology	5.47	9.96
	OWL	DAML	6.17	5.30
	ODBASE	Coopis	15.4	22.7
	ODBASE	DOA	10.0	14.6
	Neural Networks	Perceptron	8.35	6.84
	Neural Networks	Multi Layer Perceptron	8.92	10.4
2	*DAML*	*OIL*	2.89	7.28
	Notebook	*Computer*	0.05	4.38
	Operating System	Unix	1.54	5.28
	Text Mining	*Computational Biology*	3.70	-2.18
3	OWL	metadata	0.71	-3.23
	OWL	*OIL*	-0.16	-3.98
	Apple	*Computer*	-0.90	0.28
	Data Mining	Classification	1.75	1.27

the similarity values that M_5 overestimates (*text mining-computational biology*). This is expected in that M_1 provides more accurate context for a term than M_5 does. In some cases (Type 3), WebSim fails to capture similarity between terms. This is primarily because mutual information prefers high-frequency terms to low-frequency ones (i.e., gives higher similarity values for low-frequency terms). For example, in case of *DAML-OIL* and *OWL-OIL*, because of relatively low frequency of "DAML" in comparison with "OWL", WebSim captures association for *DAML-OIL* while it cannot for *OWL-OIL*. This also holds for *apple-computer*. Thus, although mutual information can detect pairwise similarity fairly well, there is a need to incorporate content (associated with the term) into WebSim, which motivates the necessity of a feature-based similarity metric.

4 WebSim: A Similarity Model Based on Feature Extraction

This section presents another similarity model based on feature extraction, which is referred to as a feature-based WebSim. Section 4.1 discusses feature extraction methodology. Section 4.2 explores similarity computation based on the extracted features.

4.1 Feature Extraction

In this section, we explain how to extract features for each term. The notions of "term" and "word" are firsts defined. Although "term" and "word" generally have same meanings, for the convention, "term" will be used to refer to the entity of similarity computation (i.e., similarity is measured between terms) while "word" will be used to refer to a feature of "term". Extracting meaningful features for WebSim consists of the following three phases:

- Retrieval of Web documents for each term.
- Preprocessing of the retrieved Web documents.
- Construction of a vector space model using a relevant feature extraction method.

A Web search engine is necessary to obtain the initial set of relevant documents for each term. Toward this end, we use the open source software, Google Web API [30] [2]. Consequently, for each t_i, a set of the top most relevant documents (D_i) is obtained by a search engine.

Next, meaningful information are extracted from Web pages in D_i using standard IR tools. This process includes HTML preprocessing (e.g., removing irrelevant HTML tags or Javascript code, etc), tokenization, stemming [22], stopwords removal, etc.

Finally, a term (t_i) is represented as a vector (v_i) in a vector space [24]. Toward this end, we employ a *bag-of-words* approach. That is, we treat each word as a feature of t_i, and represent each term as a vector of certain weighted word frequencies in this feature space. The weight of a word for each term is determined based on the following two heuristics.

- Important words occur more frequently within a document than unimportant words do.
- The more times a words occurs throughout the documents within D_i, the stronger its predicting power becomes.

The term frequency (TF) is based on the first heuristic. In WebSim, term frequency of t_i is counted in a document in D_i. In addition, TF can be normalized to reflect different document length. Let f_{ij} be the j-th feature of t_i, and $freq_{ijk}$

[2] Alternatively, we can use the front-end scraper that is presented in Section 3.

be the number of f_{ij}'s occurrences in a document d_k where $d_k \in D_i$. Then, term frequency (tf_{ijk}) of f_{ij} in d_k is defined as follows:

$$tf_{ijk} = \frac{freq_{ijk}}{l_k} \tag{3}$$

where l_k is the length of d_k.

The second heuristics is related with the document frequency (DF) of the word (the percentage of the documents that contains this word). In WebSim, since only relevant documents with respect to a term are retrieved, if appropriate stopwords are removed in the preprocessing step, then a word with high document frequency within D_i is considered to be of a particular relevant feature for a term.

A combination of TF and DF introduces a new ranking scheme, which is defined as follows:

$$w_{ij} = \frac{n_{ij}}{N_i} \times \frac{\Sigma_{d_k \in D_i} tf_{ijk}}{N_i} \tag{4}$$

where w_{ij} is an weight of f_{ij}, N_i is the total number of documents in D_i, and n_{ij} is the number of documents in D_i where f_{ij} occurs at least once.

By exploiting the fact that top (high-weighted) few features contribute substantially to the norm of a vector, only high-weighted features (that make up most of the norm) are retained. This approach reduces the significant number of features while this minimizes the loss of information.

Table 3 shows sample features. Note that all features are stemmed (e.g., "inform" refers to "information"). As shown, top features for each term characterize descriptive concepts of terms. For example, consider "knowledge discovery" and "association rules". As expected, key concepts that describe both terms are extracted as features. Note that the extracted features sometimes do not always correspond to definitions of terms. For example, for a term "knowledge discovery", "sigkdd" is not a feature for defining the term. However, this is an important feature in that it is one of the largest organizations in data mining. Similarly, "agraw" (Rakesh Agrawal) , who is an inventor of association rule mining, is extracted as a feature for "association rules". Therefore, extracted features by WebSim reflect the current trend on the term besides definition for the term.

4.2 A Similarity Model Based on Extracted Features

Once each term is represented as a vector, the next step is to measure closeness between vectors. Toward this end, we employ a Cosine metric that measures similarity of two vectors according to the angle between them [24]. The cosine of the angle between two vectors t_i and t_j is defined by

$$Sim_1(t_i, t_j) = Cosine(v_i, v_j) = \frac{\sum_{k=1}^{n} w_{ik} \cdot w_{jk}}{||v_i|| \cdot ||v_j||} \tag{5}$$

where v_i and v_j correspond to the vectors of t_i and t_j, respectively. $Cosine(v_i, v_j)$ ranges from 0 (dissimilar) to 1 (similar).

Table 3. Sample features for terms

Term	Features
Knowledge discovery	data mine knowledg discoveri kdd number confer sigkdd acm research inform volum web search scienc
Association rules	rule associ data item mine transact databas inform agraw algorithm confid analysi stream discoveri knowledg itemset
Sequential patterns	pattern mine sequenti data time sequenc databas stream algorithm associ agraw rule transact srikant frequent tempor
Decision trees	tree decis data inform node learn classif algorithm test split predict gain class train text machin model classifi attribut
Data warehouses	data warehous inform wareh busi manag softwar databas integr enterpris solut intellig servic server view decis
Object recognition	object recognit imag model vision base featur comput match visual scene view research geometr invari data shape recogn
Multi layer perceptron	layer perceptron multi network output function input neural weight learn hidden unit model neuron error linear mlp
Self organizing maps	map organ data som neural kohonen network learn vector wsom model visual cluster text similar inform
Parallel computer architecture	comput parallel architectur softwar hardwar design program memori multiprocessor perform morgan kaufmann network
Firewalls	firewal internet secur network comput connect protect softwar servic window filter packet inform server host port
Cryptography	cryptographi secur crypto inform kei privaci encrypt rsa archiv research cryptolog softwar algorithm pgp code
Machine translation	translat machin english languag french onlin text mt spanish comput german public chines associ research
Multimedia	multimedia inform video time grolieronlin photo media histori nation web archiv imag stori real flash audio view
Virtual reality	virtual realiti 3d world vr research model list time comput simul vrml applic interact commun environ motion view
Push down automata	automata push state context languag stack free pda formal grammar program correct inform input finit regular theori
Finite Automata	finit automata state algorithm determinist languag regular machin string dfa comput transit express nfa
Turing machines	machin ture comput state tape program number symbol halt run problem instruct left alan cell blank function

The underlying assumption of the proposed approach is simple yet effective: if t_i (e.g., data mining) and t_j (e.g., knowledge discovery) are similar, then the Web pages returned by t_i and t_j would be somewhat similar, consequently, Cosine value between v_i and v_j becomes high. Table 4 illustrates this. However, Sim_1 sometimes fails to capture similarity relations in case a term is ambiguous.

One of the challenging problems in the feature-based WebSim is how to deal with ambiguity of terms. That is, if a term has multiple meanings, then the returned Web pages are not coherent to a single sense of a term. For example, "clustering" has two meanings in top 10 ranked pages returned by Google (data

Table 4. Sample term pairs that have relatively high $Sim_1(t_i, t_j)$

t_i	t_j	$Sim_1(t_i, t_j)$
Semantic Web	XML	0.405
Genetic algorithms	Evolutionary computation	0.433
Encryption	Computer security	0.535
Information warfare	Computer security	0.444
Parallel computing	Computer architecture	0.623
Parallel programming	MPI	0.383
Data warehouses	Knowledge discovery	0.510
Data mining	Knowledge discovery	0.778
Natural language processing	Computational linguistics	0.439
Natural language processing	NLP	0.583

mining and computer architecture context). Consequently, due to the ambiguity of "clustering", actual similarity between "clustering" and "data mining" becomes low even though clustering is one of the subfields in data mining. This problem is also inherent in Web search. Because user queries usually tend to be short, the queries can be ambiguous, which often leads to irrelevant search results.

Table 5 shows top high-weighted features for three ambiguous terms. For example, consider "classification". Due to the generality of this term, none of the top 10 ranked pages returned by Google is related with classification in data mining context. Additionally, with regard to the top 50 pages for a query"oil" in Google, only 2 pages are about Ontology Inference Layer, and remaining pages are pointers to information on gas oil. Consequently, all extracted features for "oil" are related with gasoline. This is because Web search engines tend to rank a page based on popularity of a page using the notion of authorities and hubs [10,2] as well as how a query is matched with the page (i.e., whether the query is matched with title, etc).

The extracted features for "oil" are useful ones if domain experts intend to add "oil" (in terms of energy context) into ontologies. However, if they consider OIL (Ontology Inference Layer) in Semantic Web context, then the extracted features are problematic. Assuming "oil" in an energy context is already in ontologies (because it was coined a long time ago), OIL in a Semantic Web context is of particular interest in terms of enriching ontologies (because it is neology).

In previous information retrieval research, query expansion has been widely studied in order to provide more useful search results. That is, a query can be refined by adding additional relevant search terms. The key point here is that the added terms should be somewhat related with the original query term. Otherwise, query expansion leads to a degradation of precision [9].

Suppose t_i and t_j are in consideration of similarity computation. If the similarity between t_i and t_j is not high enough, then combined queries (i.e., $t_i t_j$ and $t_j t_i$) are issued as queries to a Web search engine, and top k documents for both terms are retrieved. As discussed, if t_i and t_j are not related with each other,

Table 5. Features for ambiguous terms

Term	Features
Clustering	cluster server softwar data technolog linux base inform window high applic search servic product load analysi releas featur group
Classification	classif link search onlin inform extern econom literatur number org list north web bookmark journal
OIL	oil energi industri shell locat ga chang product price servic bp drill origin compani petroleum

then adding an additional term will not be helpful, consequently, similarity between t_i and t_jt_i (and between t_j and t_it_j) will be still not high. However, if t_i and t_j are related with each other, then expanding t_i with t_j will result in high similarity (i.e., similarity between t_i and t_jt_i is expected to be high) [3]. Thus, if the similarity between t_i and t_j is not high enough, then the similarity will be refined as follows:

$$Sim_2(t_i, t_j) = Average(Cosine(v_i, v_{ji}),\ Cosine(v_j, v_{ij})) \tag{6}$$

where v_{ij} is a vector representation for the documents that are retrieved by issuing a query, t_it_j.

Table 6 shows how term expansion successfully refines the sense of a term. Since "classification" and "clustering" are related in data mining context, adding "clustering" to "classification" will refine the meaning of "classification". This is because there exists a sense that both terms share even though they have multiple meanings. As a result, extracted features on "classification clustering" are on data mining subject. However, expanding "linux" with "automata" destroys the true characteristics of the term rather than refines the meaning. Consequently, resulted features are distorted (distorted features are shown in italic). Therefore, term expansion is helpful only when a relevant term is added.

5 Ontology Modification with WebSim

Ontology modification is composed of two parts: ontology enrichment, and ontology restructuring. In this Section, we explore how to modify ontology with WebSim. Figure 2 provides an overview of the proposed method.

One of the key issues in ontology enrichment is how to identify candidate terms that should be added into an ontology. In our previous work, we presented topic mining, which effectively identifies useful patterns (e.g., news topics or events, key terms at multiple levels of abstraction) from news streams [5,4]. Topic mining is a key enabling technology in ontology enrichment. The idea of coupling topic mining and ontology enrichment is as follows:

Topic mining sends a Web crawler to a collection of key sites that are related with the domain of interest (or a collection of popular Web sites like CNN if

[3] In this step, both t_it_j and t_jt_i are submitted as queries to the Web search engine because the order of query terms affects the search results.

Table 6. Sample features for terms with context

Term	Features
Oil odbase	ontolog oil web semant confer inform logic descript system odbase knowle languag gobl base proceed databas model
Agent coopis	agent system inform confer cooper univers comput coopi base paper web distribut knowledg data model servic
Agent insurance	insur agent life compani agenc state servic term licens inform financi busi brok onlin quot health nationwid auto
Classification clustering	cluster classif data class analysi method algorithm text inform distanc group list network imag fuzzi vector type similar variabl model hierarch program point document
Clustering architecture	cluster architectur manag server applic network databas servic group avail softwar replic microsoft
Linux automata	linux *automata* program softwar version *cellular* simul org game *life* file comput window java *state model* 3d

we want to enrich general-purpose ontologies), and retrieves a set of domain specific documents. One of the main capabilities of topic mining is that the key topical terms are dynamically generated based on incremental hierarchical document clustering. Thus, the topic mining framework can be effectively utilized for ontology enrichment in that it can automatically identify key candidate terms/concepts from Web document streams. The identified candidates can be given to a domain expert.

Moreover, the feature extraction methodology (in Section 4.1) can be effectively used for candidate term generation. That is, as shown in Table 3, since features for each term can be key concepts that describe the main characteristics of the term, WebSim can use a term in the ontology to derive the features for the term [4]. If the obtained features do not exist in the ontology, then domain experts can add the features for the purpose of ontology enrichment. Alternatively, terms identified by topic mining can be given as input to WebSim to populate features, which will be candidate terms for ontology enrichment.

Besides adding a new term into an existing ontology, the ontology should be restructured as time evolves. That is, existing term relationships in ontologies need to be changed. Since it is extremely difficult to update ontology manually, it is necessary to suggest which parts of ontology have possibilities of changes. Toward this end, we present an WebSim-based approach to select candidate term pairs whose relationships should be modified. Due to the large number of terms in ontologies, it is computationally expensive to compute pairwise similarity between all terms in the ontology. Because we cannot predict which part of the ontology should be modified (consider OIL and XML that are far from each other in conventional conceptual ontologies), we need $O(m^2)$ pairwise similarity

[4] Since Porter stemmer is used in WebSim, it is difficult to perform reverse mapping from a stemmed word to original one. To address this problem, rather than relying on Porter stemmer, we can simply remove trailing "s" if the word is not in exception list (i.e., exception list contains words such as news).

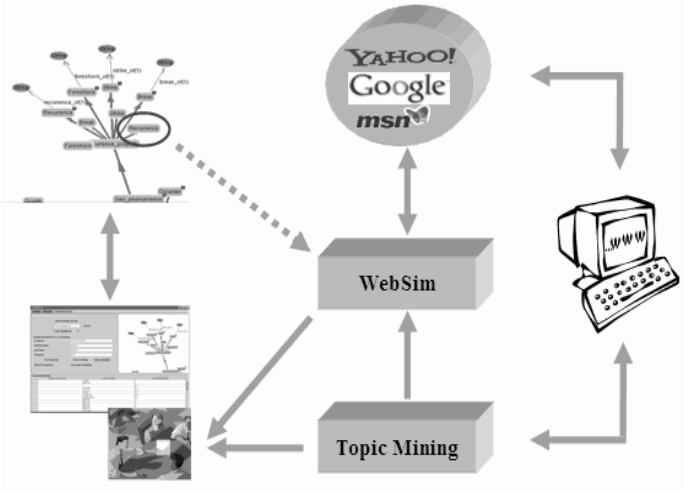


Fig. 2. Overview of the process for ontology modification with WebSim

computations if the size of the ontology is m. However, the number of computations can be significantly reduced if the WebSim is employed. As discussed, for each term, extracted features by WebSim represent the up-to-date knowledge on the term. Thus, rather than examining all term pairs in the ontology, for each term (t_i), we only need to compute similarity between t_i and the features of t_i that are in the ontology. Assuming that the number of the features for each term is constant (because only top weighted features are considered), the complexity can be reduced to $O(m)$ from $O(m^2)$, which is a significant improvement.

It is worthwhile to compare the MI-based and the feature-based WebSim, so domain experts can choose the metrics for their own purpose. In terms of computational cost, the MI-based WebSim is cheaper than the feature-based WebSim. In our simulation, the average difference between the number of documents of $t_i t_j$ and that of $t_j t_i$ is 912.70. This number is negligible considering that the average number of returned documents for the terms in our test is 8,267,363.81. Thus, for the MI-based WebSim, only 3 queries need to be submitted (i.e., t_i, t_j and $t_i t_j$). In contrast, for the feature-based WebSim, both $t_i t_j$ and $t_j t_i$ (as well as t_i and t_j) need to be submitted to a Web search engine. In sum, the MI-based WebSim needs 3 query submissions while the feature-based WebSim needs 4 query submissions. Moreover, the feature-based WebSim needs the feature extraction step, and cosine similarity computations. However, the feature-based WebSim produces more reliable similarity values than the MI-based WebSim does because mutual information is strongly influenced by the marginal probabilities of terms.

6 Semantic Similarity Versus WebSim

In this section, we present a methodology on how to investigate relatedness between WebSim and existing ontologies like WordNet.

Given a pair of terms, t_i and t_j, a simple similarity measure in ontologies is to use an edge counting method. That is, the distance corresponds to the number of edges between terms in the ontology. The shorter the path from one term to another, the more similar they are. However, this approach relies on the assumption that links in the taxonomy represent uniform distances, which does not hold in many existing ontologies. Consequently, it cannot provide correct similarity estimation.

Recently, semantic similarity metrics have been proposed to evaluate similarity between two terms in a taxonomy based on information content [12,23]. These approaches rely on the incorporation of empirical probability estimates into a taxonomic structure. Previous studies have shown that these types of approaches are significantly less sensitive to link density variability. The notions of concept frequency and concept probability are first defined as follows:

$$concept\_freq(t_i) = \sum_{t_j \in C_{t_i}} count(t_j) \tag{7}$$

where C_{t_i} is the set of terms subsumed by a term t_i. Each term that occurs in the corpus is counted as an occurrence of each concept containing it.

$$prob(t_i) = \frac{concept\_freq(t_i)}{C} \tag{8}$$

where C is the size of corpus, which is the total number of terms observed in corpus.

The information content of a term t_i ($IC(t_i)$) can be quantified based on Equation (8).

$$IC(t_i) = -log(prob(t_i)) \tag{9}$$

Equation (9) states that informativeness decreases as concept probability increases. Thus, the more abstract a concept, the lower its information content. This quantization of information provides a new approach to measure similarity between terms in ontology. The more information two terms share, the more similar they are. Resnik [23] defines the information shared by two terms as the maximum information content of the common parents of the terms in the ontology (Equation (10)).

$$Resnik(t_i, t_j) = max_{t \in CP(t_i, t_j)}[-log(prob(t))] \tag{10}$$

where $CP(t_i, t_j)$ represents the set of parents terms shared by t_i and t_j.

Because the value of Equation (10) can vary between 0 to infinity, we use Lin's metric instead [12], which varies between 0 (maximum dissimilarity) and 1 (maximum similarity).

$$Lin(t_i, t_j) = \frac{2 \times max_{t \in CP(t_i, t_j)}[-log(prob(t))]}{(IC(t_i) + IC(t_j))} \tag{11}$$

Table 7 shows semantic similarity and WebSim of selected terms. Semantic similarity is computed by using WordNet::Similarity [20]. As expected, due to

Table 7. WebSim versus semantic similarity. Undef denotes that the term does not exist in WordNet.

t_i	t_j	Lin	MI	Sim_1	Sim_2
Semantics	Metadata	0	2.222	0.171	0.653
Firewall	Encryption	0	4.566	0.218	0.699
Automata	Turing machine	0	6.640	0.078	0.500
Apple	Computer	0.121	0.909	0.153	0.556
Yahoo	Messenger	0.230	3.795	0.102	0.666
Doctor	Nurse	0.797	3.093	0.091	0.637
Microsoft	Windows	Undef	3.559	0.541	0.783
Apple	Ipod	Undef	2.408	0.644	0.876

the lack of ability to express topical relations in WordNet, we observed low semantic similarity for the first five term pairs. In contrast, our WebSim model successfully captured the similarity relations. Even though $Sim_1(t_i, t_j)$ was low for "automata" and "Turing machine", this is because of the ambiguity of "automata", which has two meanings (in theory of computation and computational learning context). For the term pair that was detected as highly similar by semantic similarity (e.g., "nurse" vs "doctor"), our WebSim could also identify high similarity using refinement. Finally, for the terms that do not exist in WordNet (e.g., Ipod or Microsoft), WebSim could capture high similarity. In sum, WebSim performs well on high semantic similarity term pairs using either the MI-based approach or refinement while uncovers topical relations that do not exist in WordNet.

7 Potential Applications of WebSim

In this Section, we explore sample applications to which WebSim is applicable.

- *Ontology matching.* Ontology matching, which aims at identifying mappings between related entities of multiple ontologies, has been widely studied recently [13,3,28]. Many different matching solutions proposed so far exploit the various properties of data such as structures of ontologies, data instances, and string similarity using edit distance. WebSim is expected to reinforce previous matching technologies in that it presents similarity metrics that are orthogonal to existing ones.
- *Recommendation Systems.* With the popularity of online sellers' product recommendation to their customers based on purchasing patterns, ontologies can be utilized to customizing a system to a user's preferences in e-commerce [29]. That is, ontologies can be effectively used for modelling customers' behavior and users' profiles. WebSim is particularly useful in this type of application in that it can extend existing taxonomy maintained by online shopping malls (e.g., amazon.com).

– *Term subspace clustering.* Subspace clustering aims at identifying clusters in subspaces of the original feature space. We recently developed an efficient subspace clustering algorithm that is scalable to the number of dimensions [8]. By coupling with the feature extraction methodology of WebSim, we can perform subspace clustering on terms such that each term can be assigned to different subspace clusters depending on the nature of the term (e.g., "clustering" can be assigned to both "data mining" and "computer architecture" clusters). Given that WebSim produces relevant features for a term with multiple meanings, WebSim can play a key role in term subspace clustering.

8 Conclusion and Future Work

In order to accommodate dynamically changing knowledge, we presented a Web search engine based similarity framework that is referred to as WebSim. WebSim is equipped with two similarity metrics, an MI-based one and a feature-based one. The former is computationally cheap while the latter can produce more reliable similarity values. In addition, we suggested diverse ways on how WebSim can be utilized for ontology modification. Finally, coupling with semantic similarity, we demonstrated how WebSim is able to identify unknown relations in WordNet.

We will extend this work into the following four directions. First, although we demonstrated the usefulness of our feature extraction methodology, not all features of a term represent distinctive characteristics for the term. To address this problem, we will study the methodology on employing different search engines (i.e., a different feature set can be generated by each search engine), and taking intersection on feature sets obtained by multiple search engines. Consequently, the resulting feature set is expected to be more robust than the one with a single search engine. Second, we plan to study a sophisticated feature weighting scheme. Based on the observation that the Web page with higher rank is generally more informative than the ones with lower rank, each Web page can be weighted by order when features are extracted. That is, the features in the first page should be weighted higher than the features in the 20-th page, and so on. Third, given that the MI-based and the feature-based similarity metrics may produce different similarity values for a same term pair, we plan to develop a new similarity measure to combine both metrics. Finally, we plan to investigate the applicability of WebSim to diverse applications such as ontology matching or term subspace clustering.

Acknowledgement

This research has been funded in part by the Integrated Media Systems Center, a National Science Foundation Engineering Research Center, Cooperative Agreement No. EEC-9529152. We also would like to thank the anonymous reviewers for their valuable comments.

References

1. E. Agirre, O. Ansa, E. Hovy, and D. Martinez. Enriching very large ontologies using the WWW. In *Proceedings of the ECAI Workshop on Ontology Learning*, 2000.
2. S. Brin, and L. Page. The anatomy of a large-scale hypertextual web search engine. In *Proceedings of the 7th International World Wide Web Conference*, 1998.
3. S. Castano, A. Ferrara, and S. Montanelli. H-MATCH: an algorithm for dynamically matching ontologies in peer-based systems. In *Proceedings of the 1st VLDB International Workshop on Semantic Web and Databases*, 2003.
4. S. Chung, and D. McLeod. Dynamic topic mining from news stream data. In *Proceedings of the 2nd International Conference on Ontologies, Databases, and Application of Semantics for Large Scale Information Systems*, 2003.
5. S. Chung, and D. McLeod. Dynamic pattern mining: an incremental data clustering approach. *Journal on Data Semantics*, 2:85-112, 2005.
6. I. Dagan, F. Pereira, and L. Lee. Similarity-based estimation of word cooccurrence probabilities. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*, 1994.
7. E.J. Glover, D.M. Pennock, S. Lawrence, and R. Krovetz. Inferring hierarchical descriptions. In *Proceedings of the ACM International Conference on Information and Knowledge Management*, 2002.
8. J. Jun, S. Chung, and D. McLeod. Subspace clustering of microarray data based on domain transformation. To appear in *Proceedings of VLDB Workshop on Data Mining on Bioinformatics*, 2006.
9. L. Khan, D. McLeod, and E.H. Hovy. Retrieval effectiveness of an ontology-based model for information selection. *The VLDB Journal*, 13(1):71-85, 2004.
10. J. Kleinberg. Authoritative sources in a hyperlinked environment. In *Proceedings of ACM-SIAM Symposium on Discrete Algorithms*, 1998.
11. D. Lenat, R. V. Guha, K. Pittman, D. Pratt, and M. Shepherd. Cyc: Toward programs with common sense. *Communications of the ACM*, 33(8):30-49, 1990.
12. D. Lin. An information-theoretic definition of similarity. In *Proceedings of the 15th International Conference on Machine Learning*, 1998.
13. J. Madhavan, P.A. Bernstein, A. Doan, and A.Y. Halevy. Corpus-based schema matching. In *Proceedings of the 21st International Conference on Data Engineering*, 2005.
14. A. Maedche, and S. Staab. Ontology learning for the Semantic Web. *IEEE Intelligent Systems*, 16(2), 2001.
15. G. Miller. Wordnet: An on-line lexical database. *International Journal of Lexicography*, 3(4):235-312, 1990.
16. M. Missikoff, P. Velardi, and P. Fabriani. Text mining techniques to automatically enrich a domain ontology. *Applied Intelligence*, 18(3):323-340, 2003.
17. M. Reinberger, P. Spyns, W. Daelemans, and R. Meersman. Mining for lexons: applying unsupervised learning methods to create ontology bases. In *Proceedings of International Conference on Ontologies, Databases and Applications of SEmantics*, 2003.
18. J. Nemrava, and V. Svátek. Text mining tool for ontology engineering based on use of product taxonomy and web directory. In *Proceedings of the Dateso Annual International Workshop on DAtabases, TExts, Specifications and Objects*, 2005.
19. N.F. Noy, M. Sintek, S. Decker, M. Crubézy, R.W. Fergerson, and M.A. Musen. Creating and acquiring Semantic Web contents with Protégé-2000. *IEEE Intelligent Systems*, 16(2):60-71, 2001.

20. T. Pedersen, S. Patwardhan, and J. Michelizzi. WordNet::Similarity - measuring the relatedness of concepts In *Proceedings of the 5th Annual Meeting of the North American Chapter of the Association for Computational Linguistics*, 2004.
21. F. Pereira, N.Z. Tishby, and L. Lee. Distributional clustering of english words. In *Proceedings of the 30th Annual Meeting of the Association for Computational Linguistics*, 1993.
22. M.F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130-137, 1980.
23. P. Resnik. Semantic similarity in a taxonomy: an information-based measure and its application to problems of ambiguity in natural language. *Journal of Artificial Intelligence Research*, 1999.
24. G. Salton and M.J. McGill. *Introduction to modern information retrieval*. McGraw-Hill, 1983.
25. M. Sanderson, and W.B. Croft. Deriving concept hierarchies from text. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1999.
26. P. Spyns, and M. Reinberger. Lexically evaluating ontology triples generated automatically from texts. In *Proceedings of the 2nd European Semantic Web Conference*, 2005.
27. Y. Sure, M. Erdmann, J. Angele, S. Staab, R. Studer, and D. Wenke. OntoEdit: collaborative ontology development for the Semantic Web. In *Proceedings of International Semantic Web Conference*, 2002.
28. M. Ehrig, and Y. Sure. Ontology mapping - an integrated approach. In *Proceedings of the 1st European Semantic Web Symposium*, 2004.
29. C. Ziegler, G. Lausen, and L. Schmidt-Thieme. Taxonomy-driven computation of product recommendations. In *Proceedings of the ACM International Conference on Information and Knowledge Management*, 2004.
30. Google Web APIs. http://www.google.com/apis/.

Erratum

LNCS 4275 Editorial

In an earlier version by mistake the volume editors have been stated instead of the authors of each paper. The current version is the correct source for any reference made to a paper included in the OTM 2006 Proceedings LNCS 4275-4278.

Erratum: Web Service Mining and Verification of Properties: An Approach Based on Event Calculus

Mohsen Rouached, Walid Gaaloul, Wil M.P. van der Aalst, Sami Bhiri, and Claude Godart

LORIA-INRIA-UMR 7503
BP 239, F-54506 Vandoeuvre-les-Nancy Cedex, France
{rouached, gaaloul, bhiri, godart}@loria.fr
Department of Technology Management, Eindhoven University of Technology
P.O. Box 513, NL-5600 MB, Eindhoven, The Netherlands
w.m.p.v.d.aalst@tm.tue.nl

R. Meersman, Z. Tari et al. (Eds.): OTM 2006, LNCS 4275, pp. 408–425, 2006.

DOI 10.1007/11914853_72

The paper entitled "Web Service Mining and Verification of Properties: An Approach Based on Event Calculus", starting on page 408 of this publication, has been retracted. A significant part of the paper was copied from four pieces of work by the authors K. Mahbub and G. Spanoudakis. The pieces of work in question are:

A Framework for Requirements Monitoring of Service Based Systems
http://dx.doi.org/10.1145/1035167.1035181

Requirements Monitoring for Service-Based Systems: Towards a Framework Based on Event Calculus http://dx.doi.org/10.1109/ASE.2004.1342769

Run-time Monitoring of Requirements for Systems Composed of Web-Services: Initial Implementation and Evaluation Experience
http://dx.doi.org/10.1109/ICWS.2005.100

A Scheme for Requirements Monitoring of Web Service Based Systems
http://www.soi.city.ac.uk/project/DOC_TechReport/TR_2004_DOC_02.pdf

Plagiarism was committed by the first author, Mohsen Rouached. The other authors were not aware of this. Moreover, the contribution of the third author, Wil M. P. van der Aalst, had nothing to do with the part of the work that was plagiarized.

The original online version for this chapter can be found at
http://dx.doi.org/10.1007/11914853_25

Author Index