LNCS 8185

Robert Meersman Hervé Panetto
Tharam Dillon Johann Eder
Zohra Bellahsene Norbert Ritter
Pieter De Leenheer Deijing Dou (Eds.)

On the Move to Meaningful Internet Systems: OTM 2013 Conferences

Confederated International Conferences:
CoopIS, DOA-Trusted Cloud, and ODBASE 2013
Graz, Austria, September 2013, Proceedings

Springer

Robert Meersman   Hervé Panetto
Tharam Dillon   Johann Eder
Zohra Bellahsene   Norbert Ritter
Pieter De Leenheer   Deijing Dou (Eds.)

On the Move to Meaningful Internet Systems: OTM 2013 Conferences

Confederated International Conferences:
CoopIS, DOA-Trusted Cloud, and ODBASE 2013
Graz, Austria, September 9-13, 2013
Proceedings

Springer

Volume Editors

Robert Meersman, Free University of Brussels, Belgium
E-mail: meersman@vub.ac.be

Hervé Panetto, University of Lorraine, Vandoevre-les-Nancy, France
E-mail: herve.panetto@univ-lorraine.fr

Tharam Dillon, La Trobe University, Melbourne, VIC, Australia
E-mail: tharam.dillon7@gmail.com

Johann Eder, University of Klagenfurt, Austria
E-mail: johann.eder@aau.at

Zohra Bellahsene, University of Montpellier II, France
E-mail: bella@lirmm.fr

Norbert Ritter, University of Hamburg, Germany
E-mail: ritter@informatik.uni-hamburg.de

Pieter De Leenheer, VU University Amsterdam, The Netherlands
E-mail: pieter.de.leenheer@vu.nl

Deijing Dou, University of Oregon, Eugene, OR, USA
E-mail: dou@cs.uoregon.edu

ISSN 0302-9743 e-ISSN 1611-3349
ISBN 978-3-642-41029-1 e-ISBN 978-3-642-41030-7
DOI 10.1007/978-3-642-41030-7
Springer Heidelberg New York Dordrecht London

Library of Congress Control Number: Applied for

CR Subject Classification (1998): D.2.2, D.2.11-12, H.3.4-5, I.2.4, C.2.4, J.1, K.4.4, K.6.5, C.2.0, H.4.1-3, F.3.2, K.6.3

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

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper
Springer is part of Springer Science+Business Media (www.springer.com)

General Co-Chairs' Message for OnTheMove 2013

The OnTheMove 2013 event, held 9–13 September in Graz, Austria, further consolidated the importance of the series of annual conferences that was started in 2002 in Irvine, California. It then moved to Catania, Sicily in 2003, to Cyprus in 2004 and 2005, Montpellier in 2006, Vilamoura in 2007 and 2009, in 2008 to Monterrey, Mexico, to Heraklion, Crete in 2010 and 2011, and to Rome in 2012. This prime event continues to attract a diverse and relevant selection of today's research worldwide on the scientific concepts underlying new computing paradigms, which, of necessity, must be distributed, heterogeneous, and supporting an environment of resources that are autonomous and yet must meaningfully cooperate. Indeed, as such large, complex, and networked intelligent information systems become the focus and norm for computing, there continues to be an acute and even increasing need to address the implied software, system, and enterprise issues and discuss them face to face in an integrated forum that covers methodological, semantic, theoretical, and application issues as well. As we all realize, email, the Internet, and even video conferences are not by themselves optimal nor even sufficient for effective and efficient scientific exchange.

The OnTheMove (OTM) Federated Conference series has been created precisely to cover the scientific exchange needs of the communities that work in the broad yet closely connected (and moving) fundamental technological spectrum of Web-based distributed computing. The OTM program every year covers data and Web semantics, distributed objects, Web services, databases, information systems, enterprise workflow and collaboration, ubiquity, interoperability, mobility, grid, and high-performance computing.

OnTheMove does not consider itself a so-called multi-conference event but instead is proud to give meaning to the "federated" aspect in its full title: it aspires to be a primary scientific meeting place where all aspects of research and development of internet- and intranet-based systems in organizations and for e-business are discussed in a scientifically motivated way, in a forum of loosely interconnected workshops and conferences. This year's event provided, for the 11th time, an opportunity for researchers and practitioners to understand, discuss, and publish these developments within the broader context of distributed and ubiquitous computing. To further promote synergy and coherence, the main conferences of OTM 2013 were conceived against a background of three interlocking global themes:

– Cloud Computing Infrastructures emphasizing Trust, Privacy, and Security
– Technology and Methodology for Data and Knowledge Resources on the (Semantic) Web
– Deployment of Collaborative and Social Computing for and in an Enterprise Context.

Originally the federative structure of OTM was formed by the co-location of three related, complementary, and successful main conference series: DOA (Distributed Objects and Applications, held since 1999), covering the relevant infrastructure-enabling technologies, ODBASE (Ontologies, DataBases, and Applications of SEmantics, since 2002) covering Web semantics, XML databases, and ontologies, and of course CoopIS (Cooperative Information Systems, held since 1993), which studies the application of these technologies in an enterprise context through, e.g., workflow systems and knowledge management systems. At the 2011 event, security issues, originally started as topics of the IS workshop in OTM 2006, became the focus of DOA as secure virtual infrastructures, further broadened to cover aspects of trust and privacy in so-called Cloud-based systems.

Each of the main conferences specifically seeks high-quality contributions and encourages researchers to treat their respective topics within a framework that simultaneously incorporates (a) theory, (b) conceptual design and development, (c) methodology and pragmatics, and (d) application in particular case studies and industrial solutions.

As in previous years 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 emergent areas related to the general area of Web-based distributed computing. This year this difficult and time-consuming job of selecting and coordinating the workshops was brought to a successful end by Yan Tang, and we were very glad to see that six of our earlier successful workshops (ORM, EI2N, META4eS, ISDE, SOMOCO, and SeDeS) re-appeared in 2013, in some cases for the fifth or even ninth time, and often in alliance with other older or newly emerging workshops. Two brand-new independent workshops could be selected from proposals and hosted: ACM and SMS. The Industry Track, started in 2011 under the auspicious leadership of Hervé Panetto and OMG's Richard Mark Soley, further gained momentum and visibility.

The OTM registration format ("one workshop buys all") actively intends to stimulate workshop audiences to productively mingle with each other and, optionally, with those of the main conferences. In particular EI2N continues to so create and exploit a visible synergy with CoopIS.

We were most happy to see that in 2013 the number of quality submissions for the OnTheMove Academy (OTMA) again increased. OTMA implements our unique interactive formula to bring PhD students together, and aims to carry our "vision for the future" in research in the areas covered by OTM. It is managed by a dedicated team of collaborators led by Peter Spyns and Anja Metzner, and of course by the OTMA Dean, Erich Neuhold. In the OTM Academy, PhD research proposals are submitted by students for peer review; selected submissions and their approaches are to be presented by the students in front of a wider audience at the conference, and are independently and extensively analysed and discussed in front of this audience by a panel of senior professors. One will readily appreciate the effort invested in this by the OTMA Faculty...

As the three main conferences and the associated workshops all share the distributed aspects of modern computing systems, they experience the application pull created by the Internet and by the so-called Semantic Web. For DOA-Trusted Cloud 2013, the primary emphasis remained on the distributed object infrastructure and its virtual and security aspects. For ODBASE 2013, the focus continued to be the knowledge bases and methods required for enabling the use of formal semantics in web-based databases and information systems. For CoopIS 2013, the focus as before was on the interaction of such technologies and methods with business process issues, such as occur in networked organizations and enterprises. These subject areas overlap in a scientifically natural fashion and many submissions in fact also treated an envisaged mutual impact among them. As with the earlier editions, the organizers wanted to stimulate this cross-pollination by a program of famous keynote speakers around the chosen themes and shared by all OTM component events. We were quite proud to announce this year

– Richard Mark Soley, OMG, USA;
– Manfred Hauswirth, DERI, Ireland;
– Herbert Zimmerman, Metasonic, Germany;
– Ainhoa Uriarte, European Commission.

And additionally, an inspiring dinner keynote by justly famous Prof. Dr. Em. Hermann Maurer, H.C. mult., allowed participants to put everything past, present, and future into a coherent context.

In spite of a general downturn in submissions observed this year for almost all conferences in computer science and IT, we were fortunate to receive a total of 137 submissions for the three main conferences and 131 submissions in total for the workshops. Not only may we indeed again claim success in attracting a representative volume of scientific papers, many from the USA and Asia, but these numbers of course allowed the Program Committees to compose a high-quality cross-section of current research in the areas covered by OTM. In fact, the Program Chairs of the CoopIS and ODBASE conferences decided to accept only approximately one full paper for each four submitted, not counting posters. For the workshops and DOA-Trusted Cloud 2013 the acceptance rate varied but the aim was to stay consistently at about one accepted paper for two to three submitted, this rate as always subordinated to proper peer assessment of scientific quality. As usual we have separated the proceedings into two volumes with their own titles, one for the main conferences and one for the workshops and posters, and we are again most grateful to the Springer LNCS team in Heidelberg for their professional support, suggestions, and meticulous collaboration in producing the files and indexes ready for downloading on the USB sticks. This year, a number of high-quality international journals will select best papers from the conferences and workshops to be extended and further submitted for a few special issues of their journal. This is also a great opportunity for some researchers to disseminate their results worldwide.

The reviewing process by the respective OTM Program Committees was as always performed to professional standards: each paper in the main conferences was reviewed by at least three referees (four for most ODBASE papers), with arbitrated email discussions in the case of strongly diverging evaluations. It may be worth emphasizing once more that it is an explicit OnTheMove policy that all conference Program Committees and Chairs make their selections completely autonomously from the OTM organization itself. As in recent years, proceedings on paper are now only available to be ordered separately.

The General Chairs are once more especially grateful to the many people directly or indirectly involved in the setup of these federated conferences. Not everyone realizes the large number of persons that need to be involved, and the huge amount of work, commitment, and in the uncertain economic and funding climate of 2013 certainly also financial risk, that is entailed by the organization of an event like OTM. Apart from the persons in their roles mentioned above, we wish to thank in particular explicitly our nine main conference PC Co-Chairs:

– CoopIS 2013: Johann Eder, Zohra Bellahsene, Rania Y. Khalaf;
– ODBASE 2013: Pieter De Leenheer, Dejing Dou, Haixun Wang;
– DOA-Trusted Cloud 2013: Norbert Ritter, Makoto Takizawa, Volkmar Lotz.

And similarly we thank the 2013 OTMA and Workshops PC (Co-)Chairs (in order of appearance on the website): Irina Richkova, Ilia Bider, Keith Swenson, Alexis Aubry, Georg Weichhart, J. Cecil, Kasuhiko Terashima, Alok Mishra, Jürgen Münch, Deepti Mishra, Ioana Ciuciu, Anna Fensel, Rafael Valencia García, Mamoun Abu Helu, Stefano Bortoli, Terry Halpin, Herman Balsters, Avi Wasser, Jan Vanthienen, Dimitris Spiliotopoulos, Thomas Risse, Nina Tahmasebi, Fernando Ferri, Patrizia Grifoni, Arianna D'Ulizia, Maria Chiara Caschera, Irina Kondratova, Peter Spyns, Anja Metzner, Erich J. Neuhold, Alfred Holl, Maria Esther Vidal, and Omar Hussain.

All of them, together with their many PC members, performed a superb and professional job in managing the difficult yet existential process of peer review and selection of the best papers from the harvest of submissions. We all also owe a serious debt of gratitude to our supremely competent and experienced Conference Secretariat and technical support staff in Brussels and Guadalajara, respectively Jan Demey and Daniel Meersman.

The General Co-Chairs also thankfully acknowledge the academic freedom, logistic support, and facilities they enjoy from their respective institutions, Vrije Universiteit Brussel (VUB), Belgium, Université de Lorraine, Nancy, France and Latrobe University, Melbourne, Australia without which such a project quite simply 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 at next year's event!

July 2013

Robert Meersman
Hervé Panetto
Tharam Dillon
Yan Tang

Organization

OTM (On The Move) is a federated event involving a series of major international conferences and workshops. These proceedings contain the papers presented at the OTM 2013 Federated conferences, consisting of three conferences, namely, CoopIS 2013 (Cooperative Information Systems), DOA-Trusted Cloud 2013, and ODBASE 2013 (Ontologies, Databases, and Applications of Semantics).

Executive Committee

General Co-Chairs

Robert Meersman	VU Brussels, Belgium
Hervé Panetto	University of Lorraine, France
Tharam Dillon	La Trobe University, Melbourne, Australia
Yan Tang	European Space Agency, Noordwijk, The Netherlands

OnTheMove Academy Dean

Erich Neuhold	University of Vienna, Austria

Industry Case Studies Program Chairs

Hervé Panetto	University of Lorraine, France

CoopIS 2013 PC Co-Chairs

Johann Eder	University of Klagenfurt, Austria
Zohra Bellahsene	University of Montpellier II, France
Rania Y. Khalaf	IBM TJ Watson Research Center, USA

DOA-Trusted Cloud 2013 PC Co-Chairs

Norbert Ritter	Universität Hamburg, Germany
Makoto Takizawa	Seikei University Tokyo, Japan
Volkmar Lotz	SAP Research, Sofia Antipolis, France

ODBASE 2013 PC Co-Chairs

Pieter De Leenheer	VU University Amsterdam, The Netherlands
Deijing Dou	University of Oregon, USA
Haixun Wang	Google Research, USA

Logistics Team

Daniel Meersman
Jan Demey

CoopIS 2013 Program Committee

Aameek Singh
Akhil Kumar
Amarnath Gupta
Andreas Wombacher
Arturo Molina
Barbara Pernici
Barbara Weber
Carlo Combi
Djamal Benslimane
Epaminondas Kapetanios
François B. Vernadat
Frank Leymann
Gerald Oster
Guohui Li
Hervé Panetto
Hiroyuki Kitagawa
Jan Hidders
Jan Mendling
Jianwen Su
John Miller
Joonsoo Bae
Julius Köpke
Lakshmish Ramaswamy
Leo Mark
Manfred Reichert
Marco Aiello
Maria Esther Vidal
Maristella Matera
Massimo Mecella
Mathias Weske
Michele Missikoff
Miyuki Nakano
Mohand-Said Hacid
Nacer Boudjlida
Nirmal Mukhi
Paul Johannesson
Ralf Schenkel
Rik Eshuis
Rong Liu
Sanjay K. Madria
Schahram Dustdar
Selmin Nurcan
Shazia Sadiq
Susan Urban
Ted Goranson
Tiziana Margaria
Werner Nutt
Willem-Jan van den Heuvel
Xiaoping Sun
Xiaoyong Du

DOA-Trusted Cloud 2013 Program Committee

Michele Bezzi
Marco Casassa Mont
David Chadwick
Luwei Cheng
Cheng-Kang Chu
Alfredo Cuzzocrea
Stefan Deßloch
Changyu Dong
Scharam Dustdar
Alex Garthwaite
Ching Hsien (Robert) Hsu
Iulia Ion
Martin Jaatun
Florian Kerschbaum

Ryan Ko
Tim Kraska
Kai Kunze
Zhiqiang Lin
Joe Loyall
Gregorio Martinez
Barzan Mozafari
Jiannan Ouyang
Nohhyun Park
Peter Pietzuch
Siani Pearson
Aravind Prakash
Smriti R. Ramakrishnan
Christoph Reich
Russell Sears
Ramesh K. Sitaraman
Charalabos Skianis
Anthony Sulistio
Andreas Thor
Bhavani Thuraisingham
Venkatanathan Varadarajan
Heng Yin
Yuan Yu

ODBASE 2013 Program Committee

Harith Alani
Yuan An
Sören Auer
Marie-Aude Aufaure
Payam Barnaghi
John Bateman
Zohra Bellahsene
Ladjel Bellatreche
Domenico Beneventano
Sonia Bergamaschi
Sourav S. Bhowmick
Omar Boucelma
Andrea Cali
Delroy Cameron
Kit Yan Chan
Sunil Choenni
Francisco Couto
Philippe Cudre-Mauroux
Sybren De Kinderen
Roberto De Virgilio
Christophe Debruyne
Emanuele Della Valle
Prasad Deshpande
Alicia Diaz
Ying Ding
Johann Eder
Tanveer Faruquie
Walid Gaaloul
Aldo Gangemi
Mouzhi Ge
Michael Grueninger
Christophe Gueret
Peter Haase
Mohand-Said Hacid
Harry Halpin
Takahiro Hara
Andreas Harth
Fedja Hadzic
Cornelia Hedeler
Rinke Hoekstra
Prateek Jain
Ruoming Jin
Amit Joshi
Christian Kop
Manolis Koubarakis
Rajasekar Krishnamurthy
Steffen Lamparter
Paea LePendu
Guoliang Li
Lior Limonad
Frederick Maier
Massimo Mecella
Davor Meersman
Eduardo Mena
Paolo Missier
Anirban Mondal
Sean O'Riain
Matteo Palmonari

Jeff Z. Pan
Maryam Panahiazar
Hervé Panetto
Josiane Xavier Parreira
Carlos Pedrinaci
Dimitris Plexousakis
Ivana Podnar Zarko
Geert Poels
Guilin Qi
Christoph Quix
Benedicto Rodriguez
Prasan Roy
Satya Sahoo
Claudio Sartori
Kai-Uwe Sattler
Christoph Schlieder
Michael Schrefl
Nigam Shah
Amit Sheth
Elena Simperl
Srinath Srinivasa
Yannis Stavrakas
Yizhou Sun
Letizia Tanca
Kunal Verma
Christian von der Weth
Yanghua Xiao
Guo-Qiang Zhang
Kenny Zhu
Lei Zou

Keynotes

Streams, Semantics and the Real World

Manfred Hauswirth

Digital Enterprise Research Institute (DERI),
Galway, Ireland

Short Bio

Manfred Hauswirth is the Vice-Director of the Digital Enterprise Research Institute (DERI), Galway, Ireland and a professor at the National University of Ireland, Galway (NUIG).

His research current research focus is on linked data streams, semantic sensor networks, sensor networks middleware, large-scale semantics-enabled distributed information systems and applications. Manfred has also worked extensively in peer-to-peer systems, Internet of things, self-organization and self-management, service-oriented architectures and distributed systems security.

He has published over 160 papers in these domains, he has co-authored a book on distributed software architectures and several book chapters on data management and semantics.

Manfred is an associate editor of IEEE Transactions on Services Computing, has served in over 180 program committees of international scientific conferences and was program co-chair of the Seventh IEEE International Conference on Peer-to-Peer Computing (IEEE P2P) in 2007, general chair of the Fifth European Semantic Web Conference (ESWC) in 2008, program co-chair of the 12th International Conference on Web Information System Engineering (WISE) in 2011, and program co-chair of the 10th International Conference on Ontologies, DataBases, and Applications of Semantics (ODBASE) in 2011.

He is a member of IEEE and ACM and is on the board of WISEN, the Irish Wireless Sensors Enterprise Led Network, the scientific board of the Corporate Semantic Web research center at FU Berlin, and the Scientific Advisory Board of the Center for Sensor Web Technologies (CLARITY) in Dublin, Ireland.

Talk

"Streams, Semantics and the Real World"

Until recently the virtual world of information sources on the World Wide Web and activities in the real world have always been separated. However, knowledge accessible on the Web (the virtual world) may influence activities in the real world and vice versa, but these influences are usually indirect and not immediate. We still lack general-purpose means to interconnect and link this information in a meaningful and simple way. Additionally, information comes in the form of streams which complicates the data management at all levels - from the Internet of Things (IoT) up to the backend information systems. The increasingly popular Linked Data paradigm provides a great model for data integration. However, supporting this approach from resource-constrained sensors on the IoT to (stream) databases and (stream) reasoning systems, possibly as hosted solutions in the cloud, opens up many genuine research problems that require well-orchestrated and synchronized research efforts in and across research communities. In this talk I will discuss these problems and possible solutions.

Empowering Process Participants - The Way to a Truly Agile Business Process Management

Herbert Kindermann

Metasonic AG, Germany

Short Bio

Since August 2009, Herbert Kindermann, has been the sole member of the Executive Board and CEO of Metasonic AG and responsible for all operative units, from marketing to software development. Kindermann focus on the company's customer orientation and the internationalization of sales and services around Metasonic® Suite. Before joining Metasonic in June 2007 as a member of the board of directors, Herbert held the position of a Member of the Executive Board at IDS Scheer with responsibility for all international business of the IDS Scheer AG. Previously, Herbert held various positions at COMSOFT GmbH (project manager, department manager and building up the SAP consulting business), IBCS S.A. (founder and CEO, building up business with subsidiaries in Germany, Czech Republic and Slovakia). In the beginning of the year 2000, IBCS became a member of the IDS Scheer group, taking over business responsibility for the region of Central and Eastern Europe. In 2003 Herbert Kindermann became a Member of the Extended Board of IDS Scheer AG.

Talk

"Empowering process participants - the way to a truly agile business process management"

Business process management (BPM) is widely adapted in large and mid-sized companies. While the focus is shifting more and more from the modelling of business processes for documentation reasons towards IT-backed business process support for the end-users there are still some open spots to consider to bring real business process management to the business departments of those companies. Herbert Kindermann will shed some light on current tool support for business process participants with respect to BPM, the actual needs of the business departments, the gap in between and how current technologies and trends, like a strong focus on KPIs, semantically enabled user interfaces, big data analytics, gamification and flexible workflow technology could lead to fundamental organizational changes and provide more enterprise agility in the future.

Long-Range Forecasting: Important Yet Almost Impossible

Hermann Maurer

TU Graz, Austria

Short Bio

Professor Dr. Hermann Maurer is Professor Emeritus at Graz University of Technology. He started his career at the University of Calgary as Assistant and Associate Professor, was appointed full professor at Karlsruhe just before he turned 30, and has been now Professor and Dean in Computer Science at Graz University of Technology since 1978, with some interruptions, like guest-professorships of more than a year at Denver University, University of Auckland, and shorter visits to Edith Cowan University in Perth, SMU in Dallas, Waterloo, Brasilia and others. Chair of the Informatics Section of Academia Europaea, "The Academy of Europe" since April 2009, and receiver of many national and international distinctions, Professor Maurer is author of over 650 papers and 20 books, founder of a number of companies, supervised some 60 Ph.D. and over 400 M.Sc. students and was leader of numerous multi-million Euro projects

Talk

"Long-range forecasting: important yet almost impossible"

In this talk I will first explain why we desperately need long range forecasts; then I present arguments (far beyond what comes to ones mind immediately) why such forecasts are in general impossible. Some of the arguments are also important for our own life and for society in general. I conclude this section, however, with one dramatic long range prediction. In the rest of the talk I discuss some important aspects of WWW, smart phones and e-Learning and conclude by showing why the main 5 theses of Spitzer's book "Digital dementia: how we ruin us and our children" are (fortunately) only partially correct, but why the impact of this book (in German, no translation exists so far) is potentially dangerous.

The Model-Driven (R)evolution

Richard Mark Soley

OMG

Short Bio

Dr. Richard Mark Soley is Chairman and Chief Executive Officer of OMG®.

As Chairman and CEO of OMG, Dr. Soley is responsible for the vision and direction of the world's largest consortium of its type. Dr. Soley joined the nascent OMG as Technical Director in 1989, leading the development of OMG's world-leading standardization process and the original CORBA® specification. In 1996, he led the effort to move into vertical market standards (starting with healthcare, finance, telecommunications and manufacturing) and modeling, leading first to the Unified Modeling Language TM (UML®) and later the Model Driven Architecture® (MDA®). He also led the effort to establish the SOA Consortium in January 2007, leading to the launch of the Business Ecology Initiative (BEI) in 2009. The Initiative focuses on the management imperative to make business more responsive, effective, sustainable and secure in a complex, networked world, through practice areas including Business Design, Business Process Excellence, Intelligent Business, Sustainable Business and Secure Business. In addition, Dr. Soley is the Executive Director of the Cloud Standards Customer Council, helping end-users transition to cloud computing and direct requirements and priorities for cloud standards throughout the industry.

Dr. Soley also serves on numerous industrial, technical and academic conference program committees, and speaks all over the world on issues relevant to standards, the adoption of new technology and creating successful companies. He is an active angel investor, and was involved in the creation of both the Eclipse Foundation and Open Health Tools. Previously, Dr. Soley was a cofounder and former Chairman/CEO of A.I. Architects, Inc., maker of the 386 Humming-Board and other PC and workstation hardware and software. Prior to that, he consulted for various technology companies and venture firms on matters pertaining to software investment opportunities. Dr. Soley has also consulted for IBM, Motorola, PictureTel, Texas Instruments, Gold Hill Computer and others. He began his professional life at Honeywell Computer Systems working on the Multics operating system.

A native of Baltimore, Maryland, U.S.A., Dr. Soley holds bachelor's, master's and doctoral degrees in Computer Science and Engineering from the Massachusetts Institute of Technology.

Talk

"The Model-Driven (R)evolution"

All sorts of promises of a revolution in software development accompany the phrase "model-driven" these days. Model Driven Architecture, Model Driven Development, Model Driven Enterprise – there must be something to these ideas, but is "model driven" the key to a revolution, or just the newest buzz word? Will we have to completely change the way we develop systems? Is code dead?

Richard Soley, Chairman of the Object Management Group (stewards of the Model Driven Architecture Initiative) will dispel some rumors about the model driven approach, and put it in the context of computing history. While there are some important implications for how complex systems are built, like most revolutions in software, Model Driven Architecture has straightforward underpinnings and represents a direct evolution from where we have been.

Overview of European Commission R&D Activities on Net Innovation

Ainhoa Uriarte

Short Bio

Ainhoa Uriarte is Project Officer in the unit "Net Innovation" of the Communications Networks, Content and Technology Directorate General in the European Commission. She has a degree in Industrial Engineering and a Postgraduate diploma in Business and Management. Previous to working for the Commission Mrs. Uriarte hold a position as research programme manager in the Spanish National Research Council (CSIC) and she has over eight years of experience working as a research project manager in several public and private institutions. She joined the European Commission in 2012 where she contributes to the implementation of the Net Innovation domain of the Information and Communication Technologies area of the 7th Framework Programme for research and technological development.

Talk

"Overview of European Commission R&D activities on Net Innovation"

The talk will give an overview of the research projects funded by the European Commission on Sensing Enterprises. Also future relevant activities under the next H2020 Framework Programme for Research and Innovation will be presented. The aim of the Future Internet challenge in the new programme is three fold; addressing the limitations of an Internet which was not designed to support the very large set of requirements imposed by an ever more diversified usage; supporting the advent of more efficient computational and data management models that respond to the challenges posed by increased device / object connectivity and data-intensive applications; and leveraging the Internet to foster innovative usages of social and economic value.

Table of Contents

OnTheMove 2013 Keynotes

Streams, Semantics and the Real World XV
Manfred Hauswirth

Empowering Process Participants - The Way to a Truly Agile Business Process Management XVII
Herbert Kindermann

Long-Range Forecasting: Important Yet Almost Impossible XVIII
Hermann Maurer

The Model-Driven (R)evolution XIX
Richard Mark Soley

Overview of European Commission R&D activities on Net Innovation XXI
Ainhoa Uriarte

Cooperative Information Systems (CoopIS) 2013

CoopIS 2013 PC Co-Chairs Message 1
Johann Eder, Zohra Bellahsene, and Rania Y. Khalaf

Business Process Management

Dynamic Weaving in Aspect Oriented Business Process Management ... 2
Amin Jalali, Petia Wohed, Chun Ouyang, and Paul Johannesson

Formal Modeling and Evaluation of Stateful Service-Based Business Process Elasticity in the Cloud 21
Mourad Amziani, Tarek Melliti, and Samir Tata

Controllability of Time-Aware Processes at Run Time 39
Andreas Lanz, Roberto Posenato, Carlo Combi, and Manfred Reichert

Process Modelling

Investigations on User Preferences of the Alignment of Process Activities, Objects and Roles 57
Agnes Koschmider, Simone Kriglstein, and Meike Ullrich

On-the-Fly Change Propagation for the Co-evolution of Business Processes 75
Shamila Mafazi, Georg Grossmann, Wolfgang Mayer, and Markus Stumptner

Online Process Discovery to Detect Concept Drifts in LTL-Based Declarative Process Models 94
Fabrizio Maria Maggi, Andrea Burattin, Marta Cimitile, and Alessandro Sperduti

Service Management

Determining the Link and Rate Popularity of Enterprise Process Information 112
Bernd Michelberger, Bela Mutschler, Markus Hipp, and Manfred Reichert

Integrated Cloud Application Provisioning: Interconnecting Service-Centric and Script-Centric Management Technologies 130
Uwe Breitenbücher, Tobias Binz, Oliver Kopp, Frank Leymann, and Johannes Wettinger

Social Networking

SCaaS: A Platform for Managing Adaptation in Collaborative Pervasive Applications 149
Muhammad Ashad Kabir, Jun Han, Alan Colman, and Jian Yu

Simulation-Based Modeling and Evaluation of Incentive Schemes in Crowdsourcing Environments 167
Ognjen Scekic, Christoph Dorn, and Schahram Dustdar

$\mathcal{DS}^4$: Introducing Semantic Friendship in Distributed Social Networks 185
Paraskevi Raftopoulou, Christos Tryfonopoulos, Euripides G.M. Petrakis, and Nikos Zevlis

Models and Schemas

Model-Driven Composition of Information Systems from Shared Components and Connectors 204
Stefania Leone, Alexandre de Spindler, and Dennis McLeod

Collaborative Schema Matching Reconciliation 222
Hung Quoc Viet Nguyen, Xuan Hoai Luong, Zoltán Miklós, Tho Thanh Quan, and Karl Aberer

Completeness and Ambiguity of Schema Cover 241
Avigdor Gal, Michael Katz, Tomer Sagi, Matthias Weidlich, Karl Aberer, Hung Quoc Viet Nguyen, Zoltán Miklós, Eliezer Levy, and Victor Shafran

Short Papers

Adaptive Consistency and Awareness Support for Distributed Software Development (Short Paper) 259
André Pessoa Negrão, Miguel Mateus, Paulo Ferreira, and Luís Veiga

Semantic Enrichment of Models to Assist Knowledge Management in a PLM environment 267
Yongxin Liao, Mario Lezoche, Eduardo Loures, Hervé Panetto, and Nacer Boudjlida

Discovering High-Level Performance Models for Ticket Resolution Processes (Short Paper) 275
Francesco Folino, Massimo Guarascio, and Luigi Pontieri

Change Patterns for Supporting the Evolution of Event-Based Systems 283
Simon Tragatschnig, Huy Tran, and Uwe Zdun

Subject-Oriented Semantic Knowledge Warehouse (SSKW) to Support Cognitive DSS 291
Tasneem Memon, Jie Lu, Farookh Khadeer Hussain, and Rajan Rauniyar

Improving Web Service Composition with User Requirement Transformation and Capability Model 300
Wenbin Li, Youakim Badr, and Frédérique Biennier

Secure Virtual Infrastructures (DOA-Trusted Cloud) 2013

DOA-Trusted Cloud 2013 PC Co-Chairs Message 308
Norbert Ritter, Makoto Takizawa, and Volkmar Lotz

Technical Advances in Cloud Computing

RouPar: Routinely and Mixed Query-Driven Approach for Data Partitioning 309
Ladjel Bellatreche, Amira Kerkad, Sebastian Breß, and Dominique Geniet

Multi-tenant Network Monitoring Based on Software Defined Networking 327
Aryan TaheriMonfared and Chunming Rong

Towards Trusted Cloud Computing

Federated Authorization for Software-as-a-Service Applications 342
Maarten Decat, Bert Lagaisse, Dimitri Van Landuyt, Bruno Crispo, and Wouter Joosen

Policy4TOSCA: A Policy-Aware Cloud Service Provisioning Approach to Enable Secure Cloud Computing 360
Tim Waizenegger, Matthias Wieland, Tobias Binz, Uwe Breitenbücher, Florian Haupt, Oliver Kopp, Frank Leymann, Bernhard Mitschang, Alexander Nowak, and Sebastian Wagner

SLA-Based Reputation Life Cycle 377
Kahina Hamadache

Privacy for the Cloud

Towards a Self-protecting Cloud 395
Anthony Sulistio and Christoph Reich

Cloud Audits and Privacy Risks 403
Thomas Rübsamen and Christoph Reich

On the Relationship between the Different Methods to Address Privacy Issues in the Cloud 414
Siani Pearson

Machine-Readable Privacy Certificates for Services 434
Marco Anisetti, Claudio A. Ardagna, Michele Bezzi, Ernesto Damiani, and Antonino Sabetta

Ontologies, DataBases, and Applications of Semantics (ODBASE) 2013

ODBASE 2013 PC Co-Chairs Message 451
Pieter De Leenheer and Deijing Dou

Querying and Mining Semantic Information

Efficient Parallel Processing of Analytical Queries on Linked Data 452
Stefan Hagedorn and Kai-Uwe Sattler

Analysis of User Editing Patterns in Ontology Development Projects ... 470
Hao Wang, Tania Tudorache, Dejing Dou, Natalya F. Noy, and Mark A. Musen

Complexity of Inconsistency-Tolerant Query Answering in Datalog+/– ... 488
Thomas Lukasiewicz, Maria Vanina Martinez, and Gerardo I. Simari

Preference-Based Query Answering in Probabilistic Datalog+/– Ontologies ... 501
Thomas Lukasiewicz, Maria Vanina Martinez, and Gerardo I. Simari

Towards a Cooperative Query Language for Semantic Web Database Queries ... 519
Stéphane Jean, Allel Hadjali, and Ammar Mars

Semantic Matching and Mapping

The AgreementMakerLight Ontology Matching System ... 527
Daniel Faria, Catia Pesquita, Emanuel Santos, Matteo Palmonari, Isabel F. Cruz, and Francisco M. Couto

Flexible Matchmaking for RESTful Web Services ... 542
Fatma Slaimi, Sana Sellami, Omar Boucelma, and Ahlem Ben Hassine

Mix'n'Match: An Alternative Approach for Combining Ontology Matchers (Short Paper) ... 555
Simon Steyskal and Axel Polleres

Evaluation of Technologies for Mapping Representation in Ontologies ... 564
Olga Kovalenko, Christophe Debruyne, Estefanía Serral, and Stefan Biffl

Semantic Information Management 1

Incremental Maintenance of RDF Views of Relational Data ... 572
Vania Maria P. Vidal, Marco Antonio Casanova, and Diego Sá Cardoso

Semantically Interlinked Notification System for Ubiquitous Presence Management ... 588
Qaiser Mehmood, Muhammad Intizar Ali, Ollie Fagan, Owen Friel, and Alessandra Mileo

Semantic Measures Based on RDF Projections: Application to Content-Based Recommendation Systems (Short Paper) ... 606
Sébastien Harispe, Sylvie Ranwez, Stefan Janaqi, and Jacky Montmain

Analyzing Dimension Mappings and Properties in Data Warehouse Integration 616
Domenico Beneventano, Marius Octavian Olaru, and Maurizio Vincini

Ontology Engineering

Semantic Enrichment of OLAP Cubes: Multi-dimensional Ontologies and Their Representation in SQL and OWL 624
Bernd Neumayr, Christoph Schütz, and Michael Schrefl

Ontology Driven Information Extraction from Tables Using Connectivity Analysis 642
Ashwin Bahulkar and Sreedhar Reddy

Efficient Projection of Ontologies 659
Julius Köpke, Johann Eder, and Michaela Schicho

Using a Reputation Framework to Identify Community Leaders in Ontology Engineering (Short Paper) 677
Christophe Debruyne and Niels Nijs

Semantic Information Management 2

Dealing with Context Ambiguity in Context-Based Information Re-finding 685
Tangjian Deng, Liang Zhao, and Ling Feng

Cognitive Modeling for Topic Expansion (Short Paper) 703
Sumant Kulkarni, Srinath Srinivasa, and Rajeev Arora

Extended Tversky Similarity for Resolving Terminological Heterogeneities across Ontologies 711
DuyHoa Ngo, Zohra Bellahsene, and Konstantin Todorov

Semantics in Use

Ontology-Based Semantic Annotation of Documents in the Context of Patient Identification for Clinical Trials 719
Peter Geibel, Martin Trautwein, Hebun Erdur, Lothar Zimmermann, Stefan Krüger, Josef Schepers, Kati Jegzentis, Frank Müller, Christian Hans Nolte, Anne Becker, Markus Frick, Jochen Setz, Jan Friedrich Scheitz, Serdar Tütüncü, Tatiana Usnich, Alfred Holzgreve, Thorsten Schaaf, and Thomas Tolxdorff

Can Ontologies Systematically Help in the Design of Domain-Specific Visual Languages? 737
Maria das Graças da Silva Teixeira, Ricardo de Almeida Falbo, and Giancarlo Guizzardi

A Data Space System for the Criminal Justice Chain 755
Jan van Dijk, Sunil Choenni, Erik Leertouwer, Marco Spruit, and Sjaak Brinkkemper

Exploiting Social Tagging in Web API Search 764
Devis Bianchini, Valeria De Antonellis, and Michele Melchiori

Erratum

On the Move to Meaningful Internet Systems: OTM 2013 Conferences E1
Robert Meersman, Hervé Panetto, Tharam Dillon, Johann Eder, Zohra Bellahsene, Norbert Ritter, Pieter De Leenheer, and Deijing Dou

Author Index 773

CoopIS 2013 PC Co-Chairs Message

Cooperative Information Systems (CIS) enable, support, and facilitate cooperation between people, organizations, and information systems. CIS provide enterprises and user communities with flexible, scalable and intelligent services to work together in large-scale networking environments. The CIS paradigm integrates several technologies: distributed systems technologies (such as middleware, cloud computing), coordination technologies (such as business process management) and integration technologies (such as service oriented computing, semantic web).

The CoopIS conference series has established itself as a major international forum for exchanging ideas and results on scientific research in all aspects of cooperative information systems such as computer supported cooperative work (CSCW), middleware, Internet & Web data management, electronic commerce, business process management, agent technologies, and software architectures, to name a few.

As in previous years, CoopIS'13 is part of a joint event with other conferences, in the context of the OTM ("OnTheMove") federated conferences, covering different aspects of distributed information systems. The call for papers attracted 65 submissions this year. In a thorough evaluation process where each paper was reviewed by at least 3 reviewers, the program committee selected 15 papers as full papers and 6 as short papers (acceptance rate 23% for full papers and 32% for short papers) which are contained in these proceedings. In addition, 6 submissions accepted as posters are published in the workshop proceedings of OTM 2013.

We extend our dear thanks to all who made CoopIS 2013 possible: Robert Meersman and his team for organizing OTM 2013 with experience and dedication, all the PC members and external reviewers who worked hard to meet the tight deadline and who provided insightful and constructive reviews for sharing their time and expertise. And last but not least to all the authors and presenters who made CoopIS again a thriving scientific event.

July 2013

Johann Eder
Zohra Bellahsene
Rania Y. Khalaf

Dynamic Weaving in Aspect Oriented Business Process Management

Amin Jalali[1], Petia Wohed[1], Chun Ouyang[2,⋆], and Paul Johannesson[1]

[1] Department of Computer and Systems Sciences, Stockholm University, Sweden
{aj,petia,pajo}@dsv.su.se

[2] Science and Engineering Faculty, Queensland University of Technology, Australia
c.ouyang@qut.edu.au

Abstract. Reducing complexity in Information Systems is an important topic in both research and industry. One strategy to deal with complexity is separation of concerns, which results in less complex, easily maintainable and more reusable systems. Separation of concerns can be addressed through the Aspect Oriented paradigm. Although this paradigm has been well researched in programming, it is still at the preliminary stage in the area of Business Process Management. While some efforts have been made to extend business process modelling with aspect oriented capability, it has not yet been investigated how aspect oriented business process models should be executed at runtime. In this paper, we propose a generic solution to support execution of aspect oriented business process models based on the principle behind dynamic weaving of aspects. This solution is formally specified using Coloured Petri Nets. The resulting formal specification serves as the blueprint to the implementation of a service module in the framework of a state-of-the-art Business Process Management System. Using this developed artefact, a case study is performed in which two simplified processes from real business in the domain of banking are modelled and executed in an aspect oriented manner. Through this case study, we also demonstrate that adoption of aspect oriented modularization increases the reusability while reducing the complexity of business process models in practice.

Keywords: Business Process Management, Aspect Oriented, Weaving, Service Oriented Architecture, Reusability, Coloured Petri Nets.

1 Introduction

Reducing the complexity of models is an important issue in the Business Process Management (BPM) area. Business process models tend to become complex quickly [4], which makes them difficult to communicate, use, maintain and validate [28]. Various approaches have been proposed to reduce the complexity of

⋆ Involvement in this work is supported by an ARC Discovery grant with number DP120101624.

process models (e.g. [15,18,32,33]). Some of these approaches have been analysed and systemised as a collection of patterns [28]. One of the patterns is called *orthogonal modularization*, which aims to reduce the complexity of a model by separating different aspects of a process, such as security and privacy. Traditionally, these aspects are defined in a single process model, hence adding to the complexity of the model [34]. In contrast, orthogonal modularization advocates modelling the aspects as separate processes. These processes are related to the main process, where they represent different pieces of the puzzle. The business process is described as a result of putting together all pieces of the puzzle. The mechanism that puts all aspects and the main process model together is called *weaving*, while the whole technique is called *aspect oriented modularization*.

Aspect oriented modularization so far has been realised as extensions to current business process modelling techniques such as Aspect Oriented Business Process Modeling Notation (AO4BPMN) [7,13,14,18]. Existing work on AO4BPEL [12] proposed an aspect-oriented extension to Business Process Execution Language for Web Services (BPEL), and their approach for weaving of apsects was defined for that specific language only. How to support the enactment of aspect oriented business process models in general is still an open issue.

In this paper, we present a solution to runtime execution of aspect oriented process models based on the principle behind dynamic weaving of aspects. It is defined in a generic manner, i.e. independent of any specific business process modelling technique or Business Process Mangement System (BPMS). The proposed solution, in the form of a so-called *Aspect Service*, is formally specified using Coloured Petri Nets (CPNs). We select CPN as it is a widely-used formal technique for system design and verification, and its application in the domain of workflow management has been well-established [1]. The CPN specification of Aspect Service serves as a blueprint for design and implementation of a service module which extends the capability of a state-of-the-art BPMS with support to aspect oriented business process enactment. The developed artefact has been used in a real banking case study to validate our solution. The case study also demonstrates that by adopting aspect oriented modularization one can reduce the complexity while increase the reusability of process models in practice.

The remainder of this paper is structured as follows. Section 2 provides a background of the aspect oriented business process modelling. Section 3 describes an overview of our solution to support weaving of aspects during process enactment. Section 4 presents the formalization of the solution in CPN. This is followed by the description of a supporting implementation within an open source BPMS environment in Section 5. Section 6 describes a case study that is conducted using the implemented artefact. Section 7 discusses related work, and finally Section 8 concludes the paper and outlines directions for future work.

2 Background

Process models encompass different activities, which address different concerns of business processes. Charfi et al. enumerate compliance, auditing, business monitoring, accounting, billing, authorization, privacy and separation of duties as examples of concerns [13]. It is common that some of these concerns are scattered across several business processes.

As a real example in Swedish public organizations, it is compulsory to inform citizens if a decision is made on their applications. Accordingly, an `inform` activity is required in all business processes that contain a `decide` activity. Moreover, a process may contain several `decide` activities, implying the need for several `inform` activities. If the `inform` activity is changed, or if the policy regarding the informing concern is modified, we have to find and update all business processes containing a `decide` activity. To be conformed to the law, when designing a new business process one has to remember to add the `inform` activity after each `decide` activity. Such efforts add costs in designing, updating and monitoring business processes, and increase the risk of inconsistency in the resulting process models. Moreover, concerns are tightly coupled with individual business processes and could not be reused. As a result, models of business processes become more complex, less reusable and more costly to design and maintain [30].

The Aspect Oriented Paradigm addresses these problems by separating different concerns from the main process. Concerns are captured in terms of *aspect*s associated with a business process and thus can be handled outside the main process. There are various works (e.g [7,13,14,18]), which provide means for aspect oriented business process modelling. Aspect Oriented Business Process Modeling Notation (AO4BPMN) [13] is one such approach that defines the terminology and suggests a notation based on BPMN for modelling processes according to the aspect oriented principle.

Let's consider an example of a business process involving different concerns. Fig. 1 describes a simplified version of a `Transfer Money Process` in the banking domain using BPMN[1]. First, a customer fills in information. Next, if the money is transferred to the customer's own account, the transfer is performed straight away; otherwise, the transaction needs to be signed beforehand. Finally, the transaction is archived. The `Sign Transaction` activity is part of the *security aspect*, and the `Archive Information` activity is part of the *logging aspect.* These aspects describe different concerns related to the `Transfer` activity.

Fig. 2 depicts the above process using AO4BPMN [13]. The concerns are removed from the main process and modelled separately through aspects. Hence, the main process contains only the `Fill Information` and `Transfer` activities. The two additional models annotated with an `Aspect` label are called *aspect models.* They capture the `Logging Aspect` and `Security Aspect`, respectively. An aspect consists of one or more *advices*, which are specified by individual process models annotated with an `Advice` label. An advice captures a specific part of a concern under a certain condition called *pointcut.* A pointcut indicates when and where the advice should be integrated with the main process. The possible points of integrations are called *join points.* A join point can be related to an aspect via a pointcut, and in such case, it is called an *advised joint point.* In AO4BPMN, join points are activities. A pointcut condition, on the one hand,

[1] For simplicity, we omit pools/lanes in this process model.

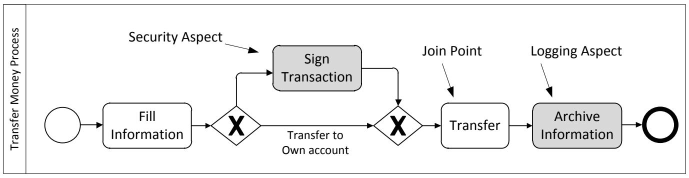


Fig. 1. Transfer Money Process model in BPMN

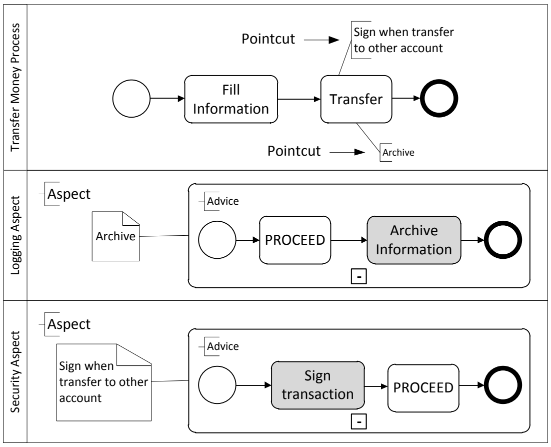


Fig. 2. Transfer Money Process model in AO4BPMN

is captured in an annotation associated with an advised joint point activity in the main process, and on the other hand, is specified in a data object as input to the corresponding advice process model. In Fig. 2, the `Transfer` activity is an example of an advised joint point with two pointcuts: one referring to the advice related to information archiving in the `Logging Aspect`, the other to the advice related to transaction signing in the `Security Aspect`.

Next, a `PROCEED` activity within an advice model acts as a "placeholder". This placeholder is for carrying out the relevant advised joint point activity during the execution of the advice. As such, the position of a `PROCEED` activity determines how the execution of an advice interleaves with the related advised joint point activity. There are three scenarios: an advice occurring *before*, *after* or *around* an advised joint point. For example, in Fig. 2, the `Archive Information` activity occurs after the `Transfer` activity, while the `Sign Transaction` activity occurs before the `Transfer` activity. It is possible that an advice does not have a `PROCEED` activity. Such advice is called an *implicit advice*, and is executed *before* the related advised joint point activity.

Comparing the process models in Fig. 1 and Fig. 2 shows that aspect oriented process modelling increases reusability because an aspect can be related to different activities and even different processes. It also makes the maintenance of process models easier, since any change to a concern would only affect the relevant aspect. Finally, it reduces the complexity of process models that capture the core processes.

3 Overview of the Solution

In the area of BPM, the existing aspect oriented modularisation approaches, such as AO4BPMN, only support process modelling. These models should be weaved in order to be executable. The weaving can be performed at design time or at runtime, which are called static or dynamic weaving correspondingly. Static weaving does not resilient to change, since for every change the process model should be weaved and loaded into BPMS. However, dynamic weaving is the approach that is flexible and solve such problem. Dynamic weaving of aspects are investigated in different areas like programming(e.g. [27,26]), service composition [11] and etc. However, it is still an open issue in BPM area.

In this section, we propose a generic solution in form of a so-called *Aspect Service*, which extends current BPMSs with runtime support to aspect oriented modularization.

According to the principle of weaving, the aspects and their advices are to be executed together with the main process, and synchronisations are to be made at the `Proceed` placeholder as well as at the end of each advice. To explain such requirements of weaving in more detail, we use an abstract example of an aspect oriented process model shown in Fig. 3. There is a main process with four aspects, which are associated to one of the activities (activity B) in the process, and each aspect contains a single advice. Based on the fact that aspect Y has an *after* advice, aspect Z has a *before* advice, aspect X has an *around* advice, and aspect W has an implicit advice, it can be determined when activity B should take place together with the four advices. Moreover, activity C, which is the activity that follows activity B, in the main process, cannot occur until the executions of *all* the advices associated with activity B are completed. Hence, the valid execution sequence of activities in the process in Fig. 3 will be A, followed by D, G and H in parallel, then B, followed by E and F in parallel, and finally C. Using regular expression this can be written as $A(D||G||H)B(E||F)C$.

At runtime, the enactment of business processes, including executions of activities in the main process and those in the associated aspects, is managed through a BPMS. The Aspect Service controls and coordinates the interactions between (the advices in) the aspects and the main process. We propose a generic approach to support dynamic weaving of aspects during process enactment. It consists of the following four steps. Note that for an implicit advice, a pre-processing is required which adds an (empty) Proceed placeholder to the end of the advice, and as such an implicit advice is treated as before advice during dynamic weaving.

1 *Launching: before executing an advised joint point, the Aspect Service shall launch all valid advices associated with that joint point.* Each valid advice is determined by the Aspect Service through evaluation of the corresponding pointcut condition. If the pointcut condition holds, the Aspect Service will initiate one instance for that valid advice.

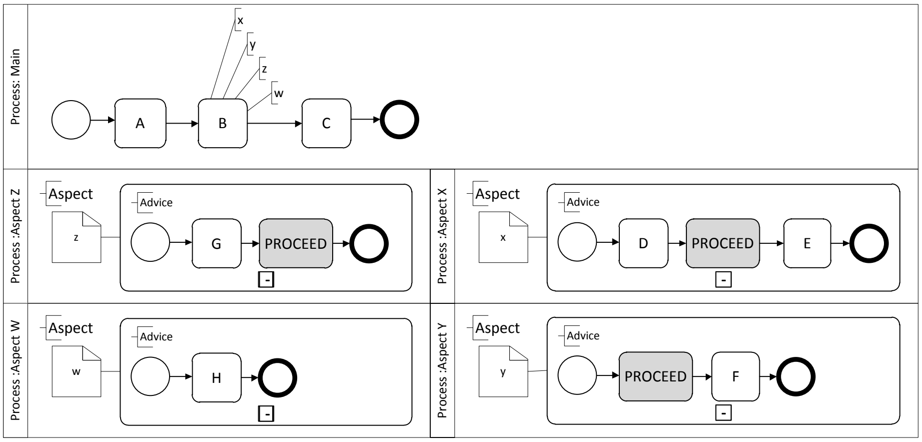


Fig. 3. An abstract example of an aspect oriented process model

> 2 *Pausing*: *the Aspect Service shall pause the execution of an advice when reaching the Proceed activity in the advice and at the same time enable the execution of the corresponding advised joint point in the main process.* It is possible that multiple advices exist for one advised join point, in which case, the `Proceed` activities in these advices should be synchronized to ensure a single execution of the advised joint point.

> 3 *Resuming*: *when the execution of the advised join point is completed, the Aspect Service shall resume the enactment of those advice instances that are interrupted by the execution of the advised joint point.* Note that for each launched advice (regardless it being a before, after, or around advice), the control of execution will be returned to the corresponding advice instance after the enactment of the advised joint point.

> 4 *Finalizing*: *the Aspect Service shall complete the executions of all the launched advice instances before the enactment of the main process can continue.* Only when the executions of all the launched advice instances finish, the control will be returned to the main process to continue its enactment (to subsequent activities enabled after the advised joint point). This signals the end of the weaving of aspects associated with that advised joint point.

Fig. 4 illustrates the dynamic weaving of the aspects with the main process using the example shown in Fig. 3. To reflect runtime nature of weaving, we use the notation of Yet Another Workflow Language (YAWL). YAWL is based on Petri nets but extended with advanced control-flow constructs to facilitate workflow modelling. In the left-hand side of Fig. 4, there are four YAWL nets capturing the main process and the three advice processes, respectively. Between each advice net and the main net, there are highlighted annotations capturing the above four-phased weaving approach. For illustration purpose, the overall behaviour resulting from the weaving of three aspects with the main process of the example in Fig. 3 is specified as the YAWL net in the right-hand side of Fig. 4. Tasks that belong to the same advice share the same graphical annotation.

Next, we look into specific states and state changes during process enactment to elaborate the above weaving approach. At runtime, an instance of an activity is called work item. Russell et al. define a general lifecycle of a work item, which describes the possible states and relation between these states during the execution of a work item [29] (See Fig. 5). In each state, different information is available concerning different perspectives . For example, the data of the resource perspective is not available when a work item is in the `Created` state. Instead, it is available when the work item is in the `Started` state.

To separate cross-cutting concerns, we need to have information regarding different perspectives. The states in which a work item has all information are `Started`, `Suspended`, `Completed` and `Failed` states. As it can be seen in Fig. 5, a work item can be alternate between `Started` state and `Suspended` state. Hence, these states are central for the weaving of advice to a main process. Therefore, the following states changes can be defined for instances of activities in both main and advices models during the weaving of aspects.

For *Launching*, the state of an advised join point shall be changed from `started` to `Suspended`. Then, all advices shall be launched. For *Pausing*, the `Suspended` state of the advised join point should be changed to `Started`, i.e. ready to be executed. For *Resuming*, the remainder activities of the main process should not be executed, and the advised join point state remains in `Completed`. The solution is to prevent instances of other activities to be `Completed`. Therefore, other instances of all other activities should be changed to `Suspended` if they reach to the `Started` state. For *Finalizing*, the suspended work items in previous step should be changed to `Started` again.

During all these steps, the data should also be synchronised between the main process and its advices. In case several advices operate on the same data simultaneously (see for example activities D and G in Fig. 4), the last event

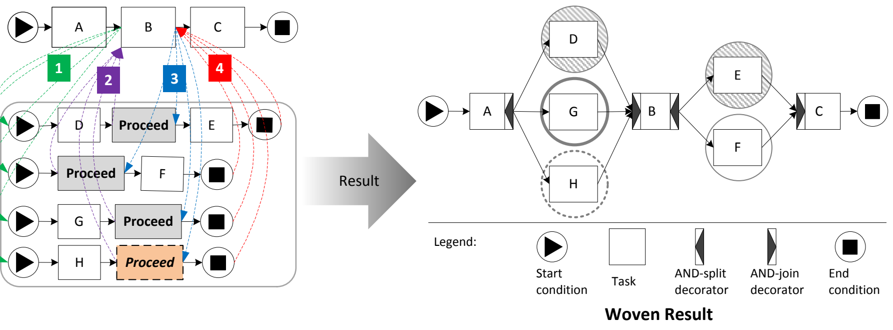


Fig. 4. Dynamic weaving of aspects using the example in Fig. 3

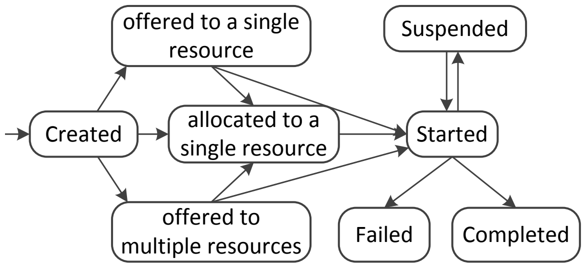

offered to a single resource
Suspended
Created
allocated to a single resource
Started
offered to multiple resources
Failed
Completed

Fig. 5. Work item Lifecycle [29]

will overwrite the data that are stored by the work items of the advised joint point, that has already been completed. In the next section, we describe the CPN model that shows the operational semantics of the weaving at runtime.

4 Formal Specification

The formalisation of the Aspect Service is specified using hierarchical Coloured Petri Nets. The solution is a three-level model. The top-level module captures the behaviour of the initiation of the service (see Fig. 6). The second level captures the weaving behaviour (see Fig. 7). This model contains four modules capturing the requirements related to weaving described in the previous section. It also contains a module for communicating with the BPMS and performing actions for data persistence, which is needed for the weaving. These five modules constitute the third level of the CPN Model. The model defines 57 colour sets and 33 functions. We re-used some of the colour sets, variables and functions from the Worklet Service CPN model [5]. A preliminary version of the CPN model from our previous work is reported in [19]. In that version, the semantic was not developed based on workitem lifecycle; while, this version is refined to be compatible with the lifecycle. This compatibility makes the semantic general for all workflow management systems.

The interaction of the Aspect Service and a BPMS is realized through passing a number of messages. These messages are named constraints and commands. *Constraints* are the messages raised by the BPMS, and *Commands* are the messages invoked by the Aspect Service.

We used standard constraints and messages according to the BPMS reference model defined by the Workflow Management Coalition (WfMC). For concrete names of messages, we adopted those in the YAWL system which is known as conforming to the WfMC's reference model. Hence, the solution is general and can be adapted to any BPMS. The constraint messages are *WorkitemConstraint* and *CaseConstraint*. The raising of one of these messages is signified in the CPN model in Fig. 6 as a token arriving in the `workitemConstraint` or `caseConstraint` places correspondingly (the places are highlighted in the figure). In other words, these places are the starting points of the net.

As it can be seen in Fig. 6, the service recognizes if the WorkitemConstraint is related to a normal activity or to a `Proceed` activity (see the `matchPointcut` and `isProceedCmd` transitions). If it is a *normal activity*, the `matchPointcut` investigates whether a pointcut is defined for the activity or not. If yes, the pointcut is evaluated to see which advices should be launched using `selectPointcut` subnet. If the pointcut is fulfilled, the `enableAspect` transition is enabled, else the `notFulfilled` transition is enabled. The result leads to *Launching* if the advised join point is just started (the `enableAspect` transition produces a token in `AspectInfo` place). However, if the advised join point is completed, the *Resuming* should be started (the `enableAspect` transition produces a token in `AdvisedJP` place). If it is a *Proceed activity*, no Pointcut is needed to be checked (the `isProceedCmd` transition produces a token in `Proceed` place). As a result, the *Pausing* can be started. The *Finalizing* can be started if all advices are finished. This condition is checked using `endAdvice` transition, which produces a token in `completedAdvice` place if a token representing the end of a launched advice appears in `caseConstraint`.

The net in Fig. 6 shows how messages received from the BPMS should be processed to enable weaving. The weaving is described by the `weaveAspect` sub-net shown in Fig 7. The `weaveAspect` net contains five subnets such as `Launching`, `Pausing`, `Resuming`, `Finalizing` and `Core`. The first four subnets fulfils the four weaving requirements, and the last one persists the data which are required to perform weaving.

The CPN model allowed us to verify the design of the Aspect Service using state space analysis. This analysis showed that the nets were free of deadlocks. Moreover, the Strongly Connected Component (SCC) graph of the model has the

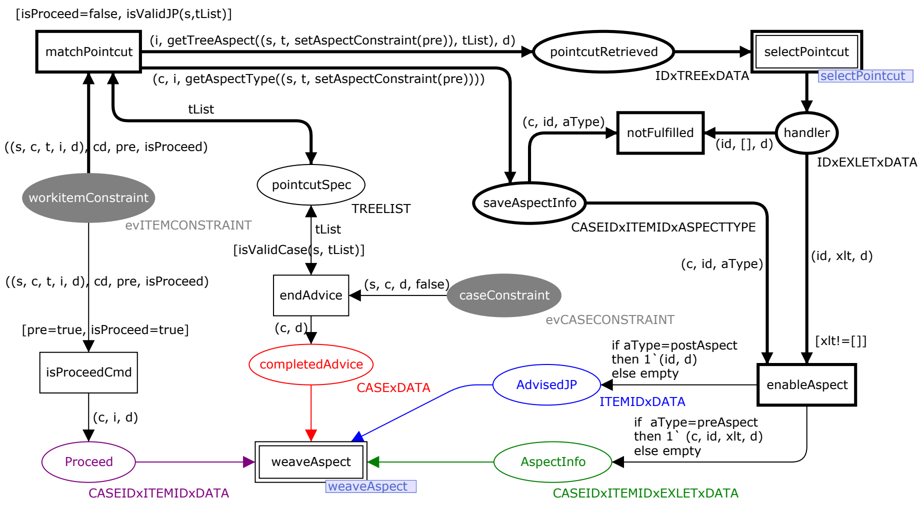


Fig. 6. CPN: Aspect Service

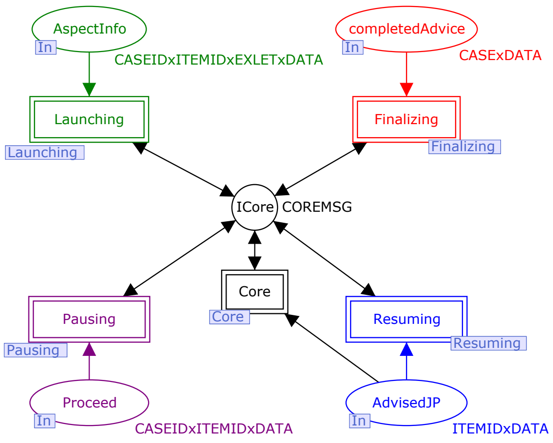


Fig. 7. CPN: weaveAspect

same number of nodes and arcs as the corresponding state space. This indicates that there is no cycle in the CPN model, which is expected since a weaving process taken into account within the scope of current work is free of cycles.

Due to space consideration the rest of the CPN model is not described here. Full details of the model with definition of the colour sets, variables, functions and analysis are publically available and can be downloaded[2].

5 Implementation

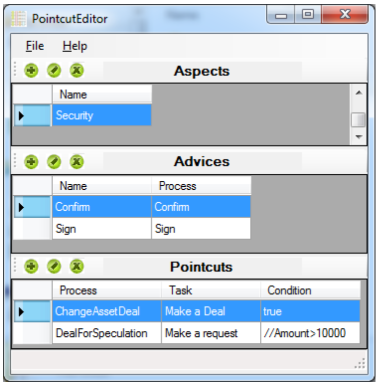
Using the formal CPN specification of our proposed solution for dynamic weaving of aspects for process enactment, we implemented two artefacts, a Pointcut Editor and an Aspect Service[3], in the framework of the YAWL system[4]. We chose YAWL because: 1) it provides support to the full workitem life cycle; 2) it has formal foundation; and 3) it is open-source and based on Service Oriented Architecture [2,3].

The Pointcut Editor (see a screenshot of the GUI in Fig. 8(a)) enables the definition of aspects, advices and pointcuts. Each aspect can have several advices, and each advice can have several pointcuts. These pointcuts consist of the name of the process and activity for which the advice should be weaved. The pointcut also includes a condition. The condition is used to define data constraints, which controls the enactment of an advice (e.g. transfer to non own account). If the condition is fulfilled, the advice will be weaved. The condition can be written using XPath language.

[2] `http://www.aobpm.com`

[3] The artefacts, with examples and case studies, can be downloaded from `http://www.aobpm.com`

[4] An open source BPMS, see `http://www.yawlfoundation.org`



PointcutEditor

File Help

Aspects

	Name
▶	Security

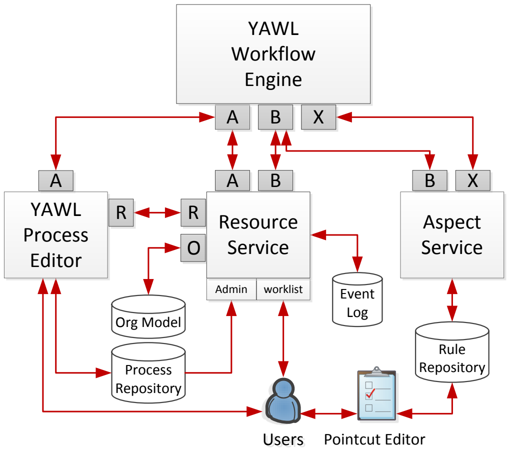
Advices

	Name	Process
▶	Confirm	Confirm
	Sign	Sign

Pointcuts

	Process	Task	Condition
▶	ChangeAssetDeal	Make a Deal	true
	DealForSpeculation	Make a request	//Amount>10000

(a) Pointcut Editor



(b) Aspect Service Architecture

Fig. 8. Implemented artefacts to enable dynamic weaving

Fig. 8(b) shows the architecture of the Aspect Service and its relation to the BPMS. The service is connected to the YAWL Engine through two interfaces, i.e. B and X. Interface X and B are used to capture case and workitem level events correspondingly. The service also reads the rules (composed by the Pointcut Editor) from the Rule Repository. The rules specify which events should be captured. During implementation, the example in Fig. 3 was used for testing, since it contains all combinations of advice types.

6 Case Study

In this section, we apply the aspect oriented approach to a real case from the financial domain[5]. The case demonstrates how the aspect oriented modularization can be used to capture cross-cutting concerns in two banking process models and thus enactment of the two aspect oriented process models using the implemented artefacts in the previous section.

These processes were modelled in a traditional way first (see Fig. 9). Then, cross-cutting concerns were separated from them by applying AO4BPMN (see Fig. 10). Afterwards, the AO4BPMN model (see Fig. 10) was manually converted to a YAWL model for execution in the YAWL system. The `Pointcut Editor` was used to define pointcuts, and the `Aspect Service` was used to enact processes.

The banking case was selected due to our previous knowledge in that domain. To choose appropriate processes, i.e. fairly simple yet representative processes, we conducted an interview with a domain expert from a bank. For the confidentiality reason, the bank asked to remain anonymous. Two processes were selected, i.e. the *Deal for speculation* process and the *Change Asset Deal* process. Detailed

[5] Translation of the banking terminology to English is done by the authors.

information about the processes was derived through a follow-up interview with the same domain expert.

The goal of the *Deal for speculation* process is to make a profit; however, sometimes money is lost. Hence, there is a limit on the amount of money that a junior and a chief dealer can trade in a deal. If a junior dealer wants to use a higher amount, an approval from his chief is needed. If the amount exceeds the limit of the chief dealer, an approval from the general manager (GM) is also needed. This approval needs to be archived by the `Office Employee`. If an approval is obtained or a deal is within one's limit, a *junior dealer* opens a position, makes the deal, and fills in a deal slip. Next, both a *chief dealer* and the *general manager* sign the deal slip, after which the deal slip is archived. After this, two parallel sets of activities are performed. On one hand, the dealt amount of money is sent to the external partner of the deal. For this, first an *employee of the Swift department* provides a swift draft for sending the money. Then, for security purposes, the *dealer*, *chief dealer* and *general manager* sign the swift draft. Finally, an *employee of the Swift department* sends out the swift. In parallel, the dealt amount of money should be received. This part starts when an *employee of the Swift department* receives an MT300 swift message. The *employee* sends this message to the general manager to control. The *general manager* makes an order to the Back office department and to the dealer to control the swift message. These orders are issued separately, according to the security policy at the bank. The results from both controls are archived separately. When the deal is made, a *back office employee* registers a voucher in the accounting system. The process ends with archiving the voucher. Fig. 10 shows the process modeled with the aspect oriented approach. In this model, the security concern is separated from the main process and captured in the Security Aspect with a number of advices, i.e. `Confirm`, `Control` and `Sign`.

The goal of the *Change Asset Deal* process is to change some asset of the bank from one currency to another. The process starts by `Backoffice Employee` who fills in the position sheet. The `General Manager` confirms the position sheet, and `Office Employee` archives it. Next, the `Junior Dealer` makes the deal and fills in dealslip. The rest of the process is the same as *Deal for speculation* process.

These processes are implemented with limited information from the data perspective. This limitation does not affect on validating the artefacts, since the current approach focuses on the execution of control-flow perspective.

Below we summarise our experiences from carrying out this case study.

- *The aspect oriented approach truly enables separation of concerns.* Separation of concerns like security or privacy increases the reusability, since they are defined once (at organisation level) and applied across the organisation where needed. It also *facilitates the maintenance* of a system. If a policy is changed, the changes are reflected in one model rather than in all business processes utilising it. In our case, if the control routines at the bank are increased, the updates are reflected in the corresponding advice(s) instead of in the processes implementing them.

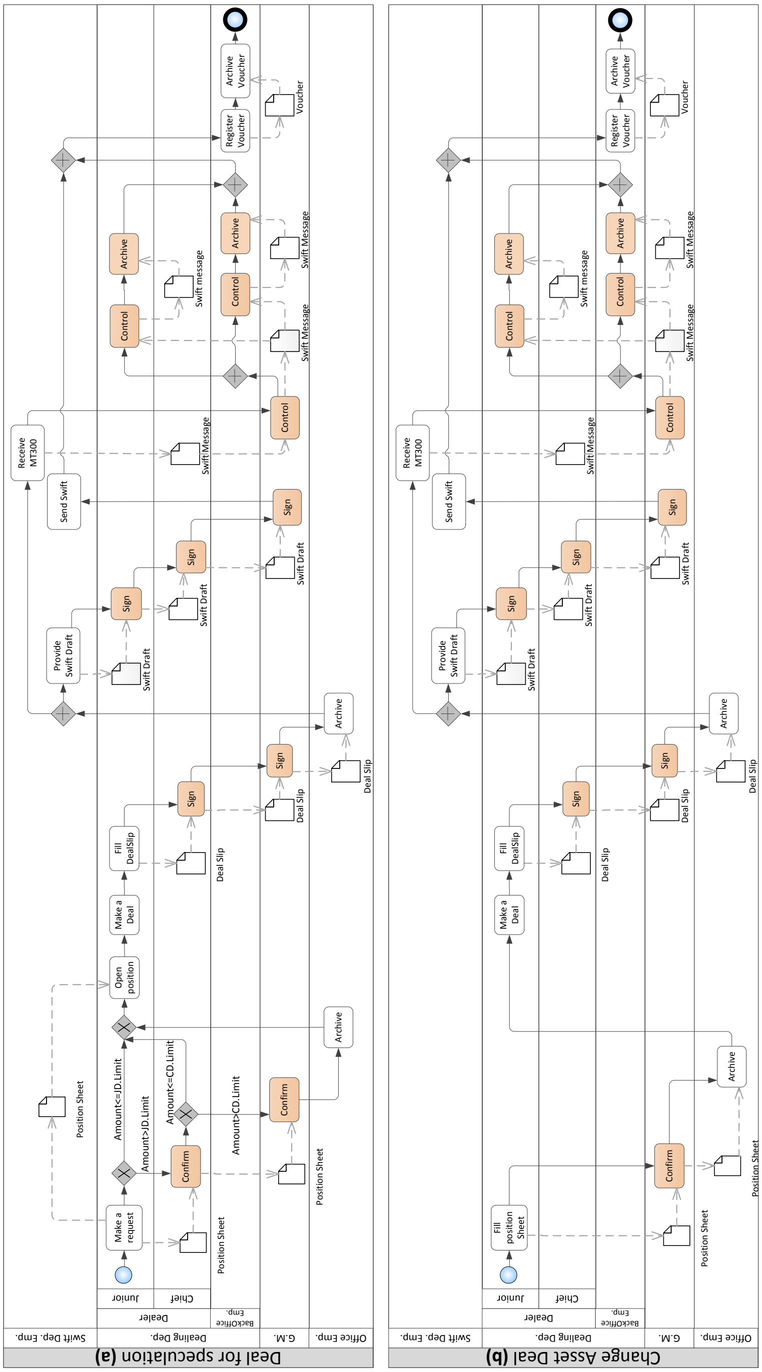


Fig. 9. The AS-IS case study processes

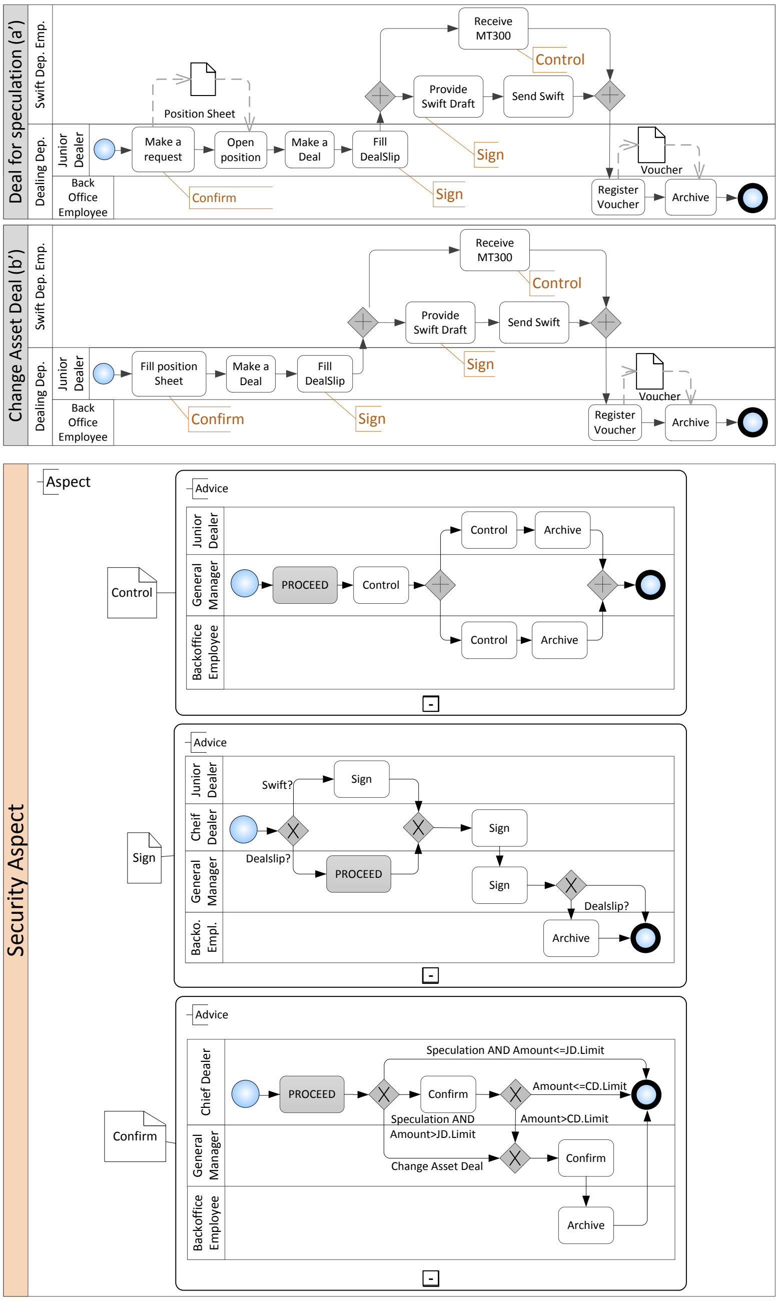


Fig. 10. The TO-BE case study processes

- *Aspect oriented decomposition decreases the complexity of process models through decreasing the overall size of models.* Hence, communicating models to business users is expected to be easier [23]. The sizes of process models are decreased both in deal processes and in overall(considering advice activities). The *deal processes* in Fig. 10 contain only half the number of activities compared to the original (not aspect oriented) model. The overall size of the set of models is also decreased since the repetitive parts in both processes are modelled once. The total size of the models applying the aspect oriented approach, were more than 25 percent less than the models not using aspect oriented decomposition. Furthermore, swimlanes which represents people who are only involved in security aspect are disappeared from the main processes, i.e. `General Manager` and `Chief Dealer`. This also adds the readability of models. It should be noted that the process models in the case study are fairly small, so the expected effect of applying aspect oriented modularization on larger process models will be even more significant.
- *Aspect oriented modelling documents additional knowledge about the business processes.* This knowledge specifies the relation between cross-cutting concerns and activities. For example, in this case study, we can see that the `Security` aspect (more precisely the `Sign` advice) is associated to the `Send Swift` activity. This information is not captured in the process modelled with traditional approaches, where we cannot interpret whether `Provide Swift Draft` or `Send Swift` activity is related to the security aspect.
- *Guidelines on how to apply aspect oriented decomposition are needed.* Sometimes different design choices are possible. For instance, the `Archive` activities in advices might also be considered as different aspects, i.e. logging. Guidelines supporting such design choices would help business process analysts in applying the aspect oriented approach.
- *The possibility to define the sequential order of advices associated to the same activity should be offered.* The approach offered by Charfi et al [13] or Cappelli et al [7] do not consider the definition of precedence for advices. This limitation enforces us to dismiss separation of some aspects from the main process or merge aspects together (like having `Archive` and `Sign` activities in `Sign` advice in our case). The first solution limits the aspect oriented modelling to separate all cross-cutting concerns from process models; while, the second one makes aspects more coupled together, which decreases the reusability of them for different process models. Hence, the Aspect Service should support the definition of precedence between advices [18]. This is to be considered as an extension to our current work.
- *The advice type should be explicitly defined in pointcut rather implicitly using a `Proceed` placeholder.* This study shows that the way that AO4BPMN proposes the definition of advice types can reduce reusability of advices. For example, a sign advice which is defined as *before* advice (using the `Proceed` placeholder) cannot be used as *after* advice later. The solution is to define the advice type in pointcut to increase the reusability of advices for both scenarios, as what is proposed in [13] and [18].

7 Related Work

To support the Aspect Oriented paradigm two components are needed: *decomposition* for capturing separation of concerns, and integration i.e. the *weaving* of aspects with processes. In the process modelling area, there are some attempts for process decomposition, e.g. [9,13,18,22,24,31]. Despite these numerous attempts, we could not find any work which shows how *weaving should be performed* in BPM area. The weaving can be performed in design time or run-time, namely static or dynamic weaving correspondingly. Static weaving suffers from lack of flexibility, since it needs models to be weaved and uploaded into business process for every change. In contrast, dynamic weaving covers this lack, and it provides flexibility to change aspects at runtime [25]. Therefore, in the work presented here, we elaborate on the *Dynamic Weaving* for BPM. The weaving is inspired by the work on weaving in programming e.g. [8,16,20,21].

Moreover, it should be mentioned that there is one implementation of weaving for service orchestration, namely AO4BPEL [12]. AO4BPEL is an extension to the Business Process Execution Language (BPEL) to support aspect orientation. This extension is defined based on soap message lifecycle [10]. This means that the BPEL4People activity lifecycle is not considered at designing this extension [6], which makes the approach specific to service decomposition. Such a limit disables AO4BPEL to address the need of separation of cross-cutting concerns in BPM area. This need is even reflected by Charfi A. where he mentions "These security concerns will not be shown in BPEL code because BPEL does not support human participants. There is however, a recent proposal for such an extension". To consider the proposal (BPEL4People), the solution (AO4BPEL) should be changed to comply with BPEL4People activity lifecycle. However, to the best of our knowledge, no research has been done to address this issue.

Furthermore, AO4BPEL cannot be used to study the needs of separation of concerns for other business process perspectives since BPEL does not support all of business process perspectives. Such a need can be exemplified as the situation where a senior employee in the bank shall confirm all activities of newly employed clerk at the first week. This separation needs definition of pointcuts to be specific for resource perspective. Such a separation cannot be investigated by AO4BPEL since it is not developed based on workitem or BPEL4People activity lifecycles [17].

8 Conclusions and Future Work

In this paper, we presented a generic solution to address how the weaving of aspects to business processes can be done. The solution is designed and implemented in form of a service, namely the Aspect Service, which extends a BPMS to support enactment of aspect oriented business process models. We provided a formalisation of the Aspect Service using CPNs and verified the soundness of the design of this service (using state space analysis). The Aspect Service is implemented in YAWL based on defined semantic. The implemented service shows that aspect oriented business process modelling increases the reusability, reduces the complexity and facilitates the maintenance of process models. The artefact is also inspected through implementing two processes of a case study from banking domain. The implementation not only shows the relevancy of the artefact to solve the separation of cross-cutting concerns, but it also reveals limitations of current aspect oriented modeling techniques. Therefore, a direction for future work is defined based on real application of aspect orientated business process modeling and enactment.

The solution is currently limited to weaving advices in which the Proceed placeholder is enabled only once. This means Proceed cannot be included in loops. Moreover, if several Proceed placeholders are defined within the same advice, care must be taken that only one of them is enabled during the execution of the advice (e.g. as a result of an XOR split). The impact of these limitations, i.e. how frequent such scenarios occur in real life, needs to be studied further.

Other directions for future work include: (i) a comparison of Aspect Orientation in the programming and BPM areas. Such comparison would fortify Aspect Oriented BPM, as the Aspect Orientation is more mature in the programming area; (ii) a definition of a pointcut language which captures other business process perspectives such as the resource perspective; (iii) an investigation on how the resource patterns [29], e.g., separation of duties and retain familiar, should be captured in orthogonal modularization; (iv) an extension to the semantic and implementation of Aspect Service to support weaving of ordered aspects in BPM area; (v) an investigation on the possibility to define nested aspects, i.e. an aspect that is related to other aspects; and (vi) extending the implementation of Aspect Service to support other WfMSs as well.

References

1. van der Aalst, W.M.P.: Three Good reasons for Using a Petri-net-based Workflow Management System. In: Information and Process Integration in Enterprises, vol. 428, pp. 161–182. Springer, US (1998)
2. van der Aalst, W.M.P., Aldred, L., Dumas, M., ter Hofstede, A.H.M.: Design and implementation of the YAWL system. In: Persson, A., Stirna, J. (eds.) CAiSE 2004. LNCS, vol. 3084, pp. 142–159. Springer, Heidelberg (2004)
3. van der Aalst, W.M.P., ter Hofstede, A.H.M.: Yawl: yet another workflow language. Information Systems 30(4), 245–275 (2005)
4. van der Aalst, W.M.P., ter Hofstede, A.H.M., Weske, M.: Business Process Management: A Survey. In: van der Aalst, W.M.P., ter Hofstede, A.H.M., Weske, M. (eds.) BPM 2003. LNCS, vol. 2678, pp. 1–12. Springer, Heidelberg (2003)
5. Adams, M.J.: Facilitating Dynamic Flexibility and Exception Handling for Workflows. PhD thesis, Faculty of Information Technology, Queensland University of Technology (November 2007)
6. Agrawal, A., et al.: WS-BPEL extension for people (BPEL4People), version 1.0. OASIS Cover Pages (2007), `http://xml.coverpages.org/BPEL4People-V1-200706.pdf`
7. Cappelli, C., et al.: Reflections on the modularity of business process models: The case for introducing the aspect-oriented paradigm. BPM Journal 16, 662–687 (1995)

8. Belblidia, N., Debbabi, M.: Formalizing AspectJ Weaving for Static Pointcuts. In: SEFM, pp. 50–59. IEEE Computer Society (2006)
9. Cappelli, C., Leite, J.C.S.P., Batista, T., Silva, L.: An aspect-oriented approach to business process modeling. In: Proceedings of the 15th Workshop on Early Aspects, EA 2009, pp.7–12. ACM (2009)
10. Charfi, A.: Aspect-oriented workflow languages: AO4BPEL and applications. PhD thesis, der Technischen Universitat Darmstadt, Darmstadt (November 2007)
11. Verheecke, B., Cibrán, M.A., Jonckers, V.: Aspect-Oriented Web Service Composition with AO4BPEL. In (LJ) Zhang, L.-J., Jeckle, M. (eds.) ECOWS 2004. LNCS, vol. 3250, pp. 168–182. Springer, Heidelberg (2004)
12. Charfi, A., Mezini, M.: AO4BPEL: An Aspect-oriented Extension to BPEL. World Wide Web 10, 309–344 (2007)
13. Charfi, A., Müller, H., Mezini, M.: Aspect-Oriented Business Process Modeling with AO4BPMN. In: Kühne, T., Selic, B., Gervais, M.-P., Terrier, F. (eds.) ECMFA 2010. LNCS, vol. 6138, pp. 48–61. Springer, Heidelberg (2010)
14. Di Francescomarino, C.: Supporting Documentation and Evolution of Crosscutting Concerns in Business Processes. In: Motahari Nezhad, H.R., et al. (eds.) ICSOC PhD Symposium. CEUR Workshop Proceedings, vol. 421 (2008)
15. Gruhn, V., Laue, R.: Reducing the cognitive complexity of business process models. In: IEEE ICCI, pp. 339–345 (2009)
16. Ho, W.-M., Jézéquel, J.-M., Pennaneac'h, F., Plouzeau, N.: A Toolkit for Weaving Aspect Oriented UML Designs. In: AOSD, pp. 99–105 (2002)
17. Jalali, A., Johannesson, P.: Multi-Perspective Business Process Monitoring. In: Nurcan, S., Proper, H.A., Soffer, P., Krogstie, J., Schmidt, R., Halpin, T., Bider, I. (eds.) BPMDS 2013 and EMMSAD 2013. LNBIP, vol. 147, pp. 199–213. Springer, Heidelberg (2013)
18. Jalali, A., Wohed, P., Ouyang, C.: Aspect oriented business process modelling with precedence. In: Mendling, J., Weidlich, M. (eds.) BPMN 2012. LNBIP, vol. 125, pp. 23–37. Springer, Heidelberg (2012)
19. Jalali, A., Wohed, P., Ouyang, C.: Operational semantics of aspects in business process management. In: Herrero, P., Panetto, H., Meersman, R., Dillon, T. (eds.) OTM 2012 Workshops. LNCS, vol. 7567, pp. 649–653. Springer, Heidelberg (2012)
20. Jézéquel, J.-M.: Model Driven Design and Aspect Weaving. Software and System Modeling 7(2), 209–218 (2008)
21. Klein, J., Fleurey, F., Jézéquel, J.-M.: Weaving Multiple Aspects in Sequence Diagrams. In: Rashid, A., Aksit, M. (eds.) Transactions on AOSD III. LNCS, vol. 4620, pp. 167–199. Springer, Heidelberg (2007)
22. Machado, I., Bonifácio, R., Alves, V., Turnes, L., Machado, G.: Managing variability in business processes: an aspect-oriented approach. In: Proceedings of the 2011 International Workshop on Early Aspects, EA 2011, pp. 25–30. ACM (2011)
23. Mendling, J., Reijers, H.A., Cardoso, J.: What Makes Process Models Understandable? In: Alonso, G., Dadam, P., Rosemann, M. (eds.) BPM 2007. LNCS, vol. 4714, pp. 48–63. Springer, Heidelberg (2007)
24. Perin-Souza, A., Cappelli, C., Santoro, F.M., Azevedo, L.G., do Prado Leite, J.C.S., Batista, T.V.: Service identification in aspect-oriented business process models. In: SOSE, pp. 164–174 (2011)
25. Pinto, M., Fuentes, L., Fayad, M.E., Troya, J.M.: Separation of coordination in a dynamic aspect oriented framework. In: Proceedings of the 1st International Conference on Aspect-Oriented Software Development, pp. 134–140. ACM (2002)

26. Popovici, A., Alonso, G., Gross, T.: Just-In-Time Aspects: Efficient Dynamic Weaving for Java. In: Proceedings of the 2nd International Conference on Aspect-Oriented Software Development, pp. 100–109. ACM Press (2003)
27. Popovici, A., Gross, T., Alonso, G.: Dynamic weaving for aspect-oriented programming. In: Proceedings of the 1st International Conference on Aspect-Oriented Software Development, AOSD 2002, pp. 141–147. ACM (2002)
28. La Rosa, M., Wohed, P., Mendling, J., ter Hofstede, A.H.M., Reijers, H.A., van der Aalst, W.M.P.: Managing Process Model Complexity Via Abstract Syntax Modifications. IEEE Trans. Industrial Informatics 7(4), 614–629 (2011)
29. Russell, N., van der Aalst, W.M.P., ter Hofstede, A.H.M., Edmond, D.: Workflow Resource Patterns: Identification, Representation and Tool Support. In: Pastor, O., Falcão e Cunha, J. (eds.) CAiSE 2005. LNCS, vol. 3520, pp. 216–232. Springer, Heidelberg (2005)
30. SantÁnna, C., Garcia, A., Chavez, C., Lucena, C., von Staa, A.: On the reuse and maintenance of aspect-oriented software: An assessment framework. In: Proceedings of Brazilian Symposium on Software Engineering, pp. 19–34 (2003)
31. Santos, N., Jack, F., do Prado Leite, S., Cesar, J., Claudia, C., Batista, T., Santoro, F.M.: Using goals to identify aspects in business process models. In: Proc. of the 2011 International Workshop on Early Aspects, EA 2011, pp. 19–23. ACM (2011)
32. Streit, A., Pham, B., Brown, R.: Visualization Support for Managing Large Business Process Specifications. In: van der Aalst, W.M.P., Benatallah, B., Casati, F., Curbera, F. (eds.) BPM 2005. LNCS, vol. 3649, pp. 205–219. Springer, Heidelberg (2005)
33. Weber, B., Reichert, M., Mendling, J., Reijers, H.A.: Refactoring large process model repositories. Computers in Industry 62(5), 467–486 (2011)
34. Weske, M.: Business Process Management: Concepts, Languages, Architectures. Springer (2007)

Formal Modeling and Evaluation of Stateful Service-Based Business Process Elasticity in the Cloud

Mourad Amziani[1,2], Tarek Melliti[2], and Samir Tata[1,⋆]

[1] Institut Mines-Telecom, TELECOM SudParis, UMR CNRS Samovar, Evry, France
[2] University of Evry Val d'Essonne, IBISC, Evry, France

Abstract. Cloud environments are being increasingly used for deploying and executing business processes and particularly Service-based Business Processes (SBPs). One of the expected features of Cloud environments is elasticity at different levels. It is obvious that provisioning of elastic platforms is not sufficient to provide elasticity of the deployed business process. Therefore, SBPs should be provided with elasticity so that they would be able to adapt to the workload changes while ensuring the desired functional and non-functional properties. In this paper, we propose a formal model for stateful SBPs elasticity that features a duplication/consolidation mechanisms and a generic controller to define and evaluate elasticity strategies.

Keywords: Cloud computing, stateful service-based business processes, elasticity, evaluation of elasticity strategies.

1 Introduction

Based on the pay-as-you-go business principle, the Cloud computing is a new model for provisioning of dynamically scalable and often virtualized IT services. Several types of services are delivered at different levels: infrastructure, platform, software, etc. These services use cloud components (such as databases, containers, VMs etc.) which themselves use cloud resources (such as CPU, memory, network).

Among other properties cloud environments provide elasticity. The principle of elasticity is to ensure the provisioning of necessary and sufficient resources such that a cloud service continues running smoothly even as the number or quantity of its use scales up or down, thereby avoiding under-utilization and over-utilization of resources [10].

Provisioning of resources can be made using vertical or horizontal elasticity [17]. Vertical elasticity increases or decreases the resources of a specific cloud service while the horizontal elasticity replicates or removes instances of cloud services [15]. Our work is mainly concerned with providing horizontal elasticity for Services-based Business Processes (SBPs). This paper does not discuss all the

⋆ The work presented in this paper was partially supported by the OpenPaaS project.

aspects that are relevant to elasticity. For example, we do not deal with ensuring vertical elasticity of cloud services and infrastructure services. While we believe that these issues are important, the provisioning of horizontal elasticity of SBPs discussed here is complex enough in itself to deserve separate treatment.

Cloud environments are increasingly being used for deploying and executing business processes and particularly SBPs. One of the expected facilities of Cloud environments is elasticity at the service and process levels.

It is obvious that provisioning of elastic platforms, *e.g.* based on elasticity of process engines or service containers [21], is not sufficient to provide elasticity of the deployed business process. Therefore, SBPs should be provided with elasticity so that they would be able to adapt to the workload changes while ensuring the desired functional and non-functional properties.

In this paper we address elasticity at the level of SBPs that mainly raises the following questions.

– What mechanisms should be developed to perform elasticity of SBPs?
– How to define and evaluate elasticity strategies of SBPs?

Among others, there are two main approaches for describing elasticity of SBPs. For a given SBP model, the first approach consists in producing a model for an elastic SBP which is the result of the composition of the SBP model with models of mechanisms for elasticity. This approach dedicates a controller for each SBP deployed but changes the nature of these latter.

The second approach that we adopt in this paper consists in setting up a controller that enforces elasticity of deployed SBPs. One can assign a single controller for all deployed processes, a controller for each subset (that corresponds to an enterprise) or even a controller for each deployed process. Actually, we have introduced a generic controller for the elasticity of business processes based on stateless services [1]. In addition, we have formally described the controller and shown how it is used for the evaluation of elasticity strategies [2]. In this paper we go further in considering the elasticity of stateful SBPs. In addition, we provide two approaches for the evaluation of elasticity strategies.

Many strategies that decide on when SBP elasticity is performed can be proposed. They use the load in each business service, in terms of the the number of current invocations, as a metric to make elasticity decisions. Some of them are reactive and some others are predictive. In this paper, we propose formal descriptions and an evaluation framework of reactive strategies.

The rest of this paper is organized as follows. Section 2 presents the state of the art. In section 3 we propose a deployment model for SBPs. In section 4, which is dedicated to the first question we raised above, we propose a formal model for elasticity of stateful SBPs. In section 5, which is dedicated to the second question we raised above, we propose a framework for the definition and evaluation of elasticity strategies using two evaluation approaches. An example, for a proof of concept, is also detailed. Section 6 concludes and suggests directions for our future work.

2 Related Work

One of the most relevant issues raised by the Cloud environment is the elasticity at different levels. Elasticity is the ability to determine the amount of resources to be allocated as efficiently as possible according to user requests. Many approaches based on predictive or reactive strategies have been proposed to address this issue [9,20]. Reactive strategies [5,13,4] are based on Rule-Condition-Action mechanisms. While predective strategies [18,8] are based on predictive-performance models and load forecasts.

At the Infrastructure level, generally two approaches are used to perform elasticity: Vertical elasticity which consists in adding or removing resources to virtual machines (VMs) to prevent over-loading and under-loading [6,18,8]. Horizontal elasticity on the other hand consists of adding or removing instances of VMs according to demands variations [12,3,8]. These approaches ensure the elasticity at the infrastructure level, but they are not sufficient to ensure the elasticity of deployed process. At the platform level, elasticity mechanisms have been proposed to ensure containers elasticity [4,21]. Nonetheless, provisioning of elastic platforms is not sufficient to provide elasticity of deployed SBPs since they do not take into account the nature of the application *e.g.,* SBPs. In fact, each application has a maximal capacity, beyond this capacity the QoS decreases and can make the container unresponsive and consequently, crash the application. Giving to the container more resources will not solve the problem [21].

At the Software level, SBPs mechanisms must be provided to ensure the elasticity of SBPs. In [7], the authors propose an approach to ensure elasticity of processes in the Cloud by adapting resources and their non-functional properties with respect to quality and cost criteria. Nevertheless, the authors addressed elasticity of applications in general rather than processes particularly. In [19], the authors consider scaling at both the service and application levels in order to ensure elasticity. They discuss the elasticity at the service level as we did in our approach. Nevertheless, the proposed approach is not based on a formal model. In [15], the authors present *ElaaS*, a service implemented as a SaaS application for managing elasticity in the Cloud. While the idea of pushing elasticity management to the applications is in line with our approach, the proposed approach is difficult to use since it requires an effort from the application designer to provide the necessary information for elasticity enforcement.

In [1] we considered the elasticity of stateless SBPs and provided duplication and consolidation mechanisms. In [2] we formally proved the correctness of our elasticity mechanisms. In addition, we have provided a framework to evaluate strategies based on duplication/consolidation. In this work we go further by considering the elasticity of stateful SBPs. We also propose two approaches for the evaluation of elasticity strategies.

At the best of our knowledge, the approaches for elasticity mainly those we cite above, focus on the IaaS level. As stated before, ensuring elasticity at the IaaS level is not sufficient to provide users with elasticity of deployed SBPs. Similarly, ensuring elasticity at the PaaS level is not enough to ensure elasticity of deployed SBPs. We believe that elasticity should be handled and tuned at different levels of Cloud environments. We have already contributed to the elasticity of platforms at the PaaS level [21]. The work we present in this paper is novel in the sense that it (1) tackles the problem of elasticity at the SaaS level (particularly for stateful SBPs) and (2) is based on a formal model (3) proposes a framework for defining and evaluating elasticity strategies and (4) proposes two approaches for the evaluation of elasticity strategies.

3 Model for SBPs Deployment

A SBP is a business process that consists in assembling a set of elementary IT-enabled services. These services carry out the business activities of the considered SBP. Assembling services into a SBP can be ensured using any appropriate service composition specifications (*e.g.* BPEL). In Figure 1-(a) we presents an example of SBP composed by eight services modeled in BPMN.

To model SBPs, several techniques can be used (BPEL, BPMN, Petri nets). In our work, we are interested in the formal aspect of the model. So, we choose Petri nets to model SBPs. Generally the modeling of SBPs using Petri nets represents the SBPs execution model.

3.1 SBPs Execution Model

The SBPs execution model specifies how the processes and their services need to be executed and in what order. In this model, each service is represented by a transition. The places represent the states between services.

The execution model of the SBP of Figure 1-(a) gives the Petri net shown in Figure 1-(b).

The SBPs execution model is suitable to verify behavioral properties. Nevertheless, with this model we can not verify non-functional properties e.g. QoS properties. Indeed, the execution model does not provide a view of the evolution of loads on services which is necessary to verify this kind of properties. Therefore, it would be interesting to have a view of the way services are deployed and their loads. For that reason, we propose, using a transformation procedure [1], to automatically derive a deployment model from the execution model of a SBP.

3.2 SBPs Deployment Model

The obtained SBPs deployment model is also modeled using Petri nets. In this model, each service is represented by a place. The transitions represent calls transfers between services according to the behavior specification of the SBP. In fact, instead of focusing on the execution model of the process and its services, we focus on the dynamic (evolution) of loads on each basic service participating in the SBP.

[1] Due to the lack of space and the heaviness of notations, the transformation rules are not given in this paper.

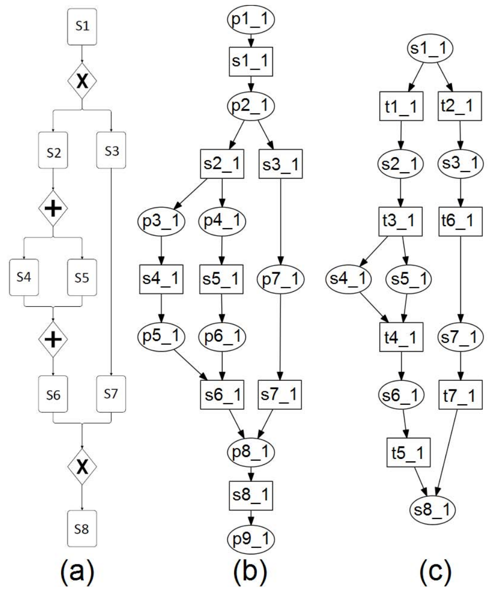


Fig. 1. An example of the transformation of a SBP execution model (b) to a deployment model (c)

The SBP deployment model of the SBP execution model of Figure 1-(b) gives the Petri net shown in Figure 1-(c).

The SBPs deployment model represents the way a process and its services are deployed and the load on each services of the SBP. The advantage of using this deployment model is to be able to represent information that are inexpressible on the execution model. This would allow verifying some properties that cannot be verified in the execution model e.g. QoS, deployment properties. Using the deployment model we can, for example, monitor the load of a service (the number of current invocations of a service) which is represented by the marking of its corresponding place. The marking of places represents load distribution over services of the process. This facilitates the implementation of load-based mechanisms e.g. elasticity and load balancing mechanisms.

In the rest of paper we will focus on the elasticity of SBPs. For that reason, we will use the deployment model to represent SBPs.

4 Formal Model for Stateful SBPs Elasticity

Elasticity of a SBP is the ability to duplicate or consolidate as many instances of the process or some of its services as needed to handle the dynamics of the received requests. Indeed, we believe that handling elasticity does not only operate at the process level but it should operate at the level of services too. It is not necessary to duplicate or consolidate all the services of a considered SBP while the bottleneck comes from some services of the SBP.

Services involved in a SBP can be stateless or stateful services. A stateless service is a service that does not store its state between two service invocations. Each service invocation is completely independent of previous invocations. On the other hand, a stateful service is a service designed to store its state between invocations. Interactions and events occurring during service execution are taken into account to manage the service invocations. The state of a stateful service is represented by the user sessions and the data values specific to this service.

Performing elasticity on stateless services can be done using a service duplication/consolidation approach without taking into account the state of the duplicated/consolidated service [1]. However, performing elasticity on stateful service is more complicated. In fact, in a duplication/consolidation approach it is necessary to ensure that the state of the stateful service is taken into account in the elasticity mechanisms at each duplication or consolidation. To solve this problem, we propose to model stateful SBPs using Colored Petri Nets (CPN). In our model, the management of user sessions is allowed by the use of colors. Each user session represents a state of the service, and so, represents a color. On the other hand, to model the data values specific to a stateful service, we propose to model each stateful service by a stateless service and a database deployed as a service in which the data values of the service are stored during its execution. Each stateful service of the SBP will have its specific database service that models the data values of all user sessions. Note that this database service can be also duplicated/consolidated as other services that compose the SBP in order to ensure its elasticity.

4.1 Stateful SBP Modeling

To model stateful SBP we use Colored Petri Nets (CPN). Classical Petri nets does not allow the modeling of data. CPN have been proposed to extend Petri nets by modeling data with color. A Petri net is a colored Petri net if its tokens can be distinguished by colors. Each place has an associated type determining the kind of data that this place may contain. The marking of a given place is a multi-set of values of the associated type. Arcs constraints are expressions that extract or produce multi-sets with respect to the sources of target types.

In order to give a definition of the CPN, we give here, without a loss of generality, a simple syntax and semantics for expressions.

- Types: Noted by Π, we range over by using π. Types are defined by the set of values that compose them, $\pi = \{v_0, ..., v_i, ...\}$. Also, types can be defined by applying set operations on them.
- Variables: Noted by $\mathcal{X}$, we range over by $\mathcal{X}_i$, Variables are typed and as usual we use $Type(\mathcal{X})$ to obtain the type of $\mathcal{X}$.
- Function: Denoted by F, for a function $f \in F$ with $f : \pi \to \pi'$ we use $Type(f)$ to define its range type.

Definition 1. *(Multi-set) : Let E be a set, a multi-set m on E is an application from E to $\mathbb{N}$, we write such a multi-set using the formal sum notation i.e $m = \sum_{0<i\leq|E|} q_i' e_i$ (with $q_i \in \mathbb{N}$ and $e_i \in E$)*[2]*. We denote by $\mathcal{M}(E)$ the set of multi-sets of E.*

We use $\mathcal{E}$ to define a color expression which can be a color constant, variable, or a color function. Given an expression $e \in \mathcal{E}$, we use $Var(e)$ to denote the set of variables which appear in e.

Definition 2. *(CPN graph) : A stateful SBP deployment model is a Colored Petri Net graph (CPN graph) $N = \langle \Sigma, P, T, cd, Pre, Post, \equiv_P, \equiv_T \rangle$, where:*

- *Σ is a set of non-empty types, also called color sets (represents the set of user sessions).*
- *P is a set of labeled places (represents the set of services/activities involved in a SBP);*
- *T is a set of labeled transitions (represents the call transfers between services according to the SBP behavioral specification);*
- *$cd : P \rightarrow \Pi$ is a function that associates to each place a color domain. Intuitively, this means that each token in place p must have a data value that belongs to $cd(p)$;*
- *Pre (resp. $Post$): are forward (resp. backward) matrices, such that $Pre : P \times T \rightarrow \mathcal{M}(\mathcal{E})$ (resp. $Post : P \times T \rightarrow \mathcal{M}(\mathcal{E})$, represent the input (resp. output) arc expressions.*
- *$\equiv_P \subseteq P \times P$: an equivalence relation over P. An equivalence relation between copies of the same place: $[p]_{\equiv_P} = \{p' | (p, p') \in \equiv_P\}$.*
- *$\equiv_T \subseteq T \times T$: an equivalence relation over T. An equivalence relation between copies of the same transition: $[t]_{\equiv_T} = \{t' | (t, t') \in \equiv_T\}$.*

In our model, each service is represented by a place with a session identifier as an associated type. Each service call is typed with its session identifier. The transitions represent calls transfers between services according to the behavior specification of the SBP while respecting the different user sessions.

As stated above, in order to manage the data values of stateful services, we add a place (database service) for each stateful service of the SBP to model the data values related to this stateful service. If the SBP contains a certain number of stateful services, we will have the same number of database services so each database service manage the data values of its corresponding stateful service. For each stateful service $s \in P$:

- *$P = P \cup \{sDB\}$ (sDB: database service of the stateful service s)*
- *$\forall t \in T : Pre(sDB, t) = Pre(s, t) \wedge Post(t, sDB) = Post(t, s)$*

For a place p and a transition t we denote ${}^\bullet p$ and $p^\bullet$ as the input and output transitions set of place p, ${}^\bullet t$ and $t^\bullet$ as the input and output places set of transition t.

[2] For simplicity we keep only the terms with $q_i \neq 0$.

The $^\bullet$ notation can also be naturally extended to equivalent classes of places and/or transitions as the union of its application to all the elements of the class *e.g.* $[p]^\bullet = \bigcup\limits_{p' \in [p]} p'^\bullet$. We extend the notation [] to a set of places and transitions *e.g.* for some $P' \subseteq P$, $[P']_{\equiv_P} = \{[p]_{\equiv_P} | p \in P'\}$. We ignore the $_{\equiv_T}$ and $_{\equiv_P}$ if it is clear from the context.

Definition 3. *(Well-formed graph) A CPN graph* N=$\langle \Sigma$,P, T,cd,Pre,$Post$, $\equiv_P$,$\equiv_T\rangle$ *is well formed iff:* $\forall t \in T, \forall p \in t^\bullet$, *we have* $Var(Post(p,t)) \subseteq Var(Pre(.,t))$ *with* $Var(Pre(.,t)) = \bigcup\limits_{p' \in {}^\bullet t} var(Pre(p',t))$.

In a well-formed CPN graph, we restrict that for each transition, the output arc expressions must be composed by the variables which are in the input arcs expressions. To each CPN graph, we associate its terms incidence Matrix C $(P \times T \rightarrow \mathcal{M}(\mathcal{E}))$ with $C = Post - Pre$.

In the following, we define the behaviors (the dynamics) of a CPN System.

Definition 4. *(CPN Marking) A marking* M *of a CPN graph is a multiset vector indexed by* P, *where* $\forall p \in P, M(p) \in \mathcal{M}(cd(p))$. *The marking is also extended to equivalent classes i.e.* $M([p]) = \sum\limits_{p' \in [p]} M(p')$. *The marking of a CPN represents a distribution of calls over the set of services that compose the SBP.*

Definition 5. *(CPN system) A Colored Petri Net system (CPN system) is a pair* S=$\langle N, M\rangle$ *where* N *is a CPN graph and* M *is one of its marking. A CPN system models a particular distribution of calls over the services of a deployed SBP.*

We use $u : Var(Pre(.,t)) \rightarrow \Sigma$ with $M \geq Pre(.,t)^u$ to denote a binding of the input arcs variables. [3]

Definition 6. *Given a CPN system* $S = \langle N, M\rangle$ *and a transition* t, *we use* $M[t\rangle^u$ *to denote that the transition* t *is fireable in the marking* M *by the use of* u, *and we use the classic notation* $M[t\rangle$ *if* u *is not important (e.g. when* u *is unique). A class of transitions is fireable in* M, $M[t\rangle^u$, *iff* $\exists t' \in [t] : M[t'\rangle^u$

Definition 7. *Let* M *be a marking and* t *a transition, with* $M[t\rangle^u$ *for some* u. *The firing of the transition* t *changes the marking of CPN from* M *to* $M' = M + C(.,t)^u$. *We note the firing as* $M[t\rangle^u M'$.

The transition firing represents the evolution of the load distribution after calls transfer. The way that calls are transferred between services depends on the behavior specification (workflow operators) of the SBP.

[3] u must respect the color domain of the places, *i.e.* , $\forall p \in^\bullet t$, $x \in var(Pre(p,t))$, we have $u(x) \in cd(p)$.

4.2 Elasticity Operations

Place Duplication

Definition 8. *Let $S = \langle N, M\rangle$ be a CPN system and let $p \in P$, the duplication of p in S by a new place p^c ($\notin P$), noted as $D(S, p, p^c)$, is a new CPN system $S' = \langle N', M'\rangle$ s.t*

- $\Sigma' = \Sigma$
- $P' = P \cup \{p^c\}$
- $T' = T \cup T''$ *with* $T'' = \{t^c | t \in ({}^\bullet p \cup p^\bullet) \wedge t^c = \eta(t)\}$ *($\eta(t)$ generates a new copy of t which is not in T).*
- $cd' : P' \to \Sigma'$ *with* $cd'(p') = cd(p')$ *for all* $p' \in P$ *and* $cd'(p^c) = cd(p)$
- Pre' *(resp.* $Post'$*):* $P' \times T' \to \mathcal{M}(\mathcal{E})$ *(resp.* $P' \times T' \to \mathcal{M}(\mathcal{E})$*)*
- $\equiv_{P'} \subseteq P' \times P'$ *with* $\equiv_{P'} = \equiv_P \cup \{(p, p^c)\}$. *The place p and its copy are equivalent.*
- $\equiv_{T'} \subseteq T' \times T'$ *with* $\equiv_{T'} = \equiv_T \cup \{(t, t^c) | t^c \in T''\}$. *Each transition is equivalent to its copy.*
- $M' : P' \to \mathcal{M}(cd(p))$ *with* $M'(p') = M(p')$ *if* $p' \neq p^c$ *and* $\emptyset$ *otherwise.*

The Pre' (resp. $Post'$) functions are obtained by extending the Pre (resp. $Post$) to the new added places and transitions as follow:

$$Pre'(p', t') = \begin{cases} Pre(p', t') & p' \in P \wedge t' \in T \\ Pre(p', t) & t \in T \wedge t' \in (T' \setminus T) \wedge t' \in [t]_{\equiv_{T'}} \wedge p' \in (P \setminus \{p\}) \\ Pre(p, t) & t \in T \wedge t' \in (T' \setminus T) \wedge t' \in [t]_{\equiv_{T'}} \wedge p' = p^c \\ \emptyset & otherwise. \end{cases}$$

$$Post'(t', p') = \begin{cases} Post(t', p') & p' \in P \wedge t' \in T \\ Post(t, p') & t \in T \wedge t' \in (T' \setminus T) \wedge t' \in [t]_{\equiv_{T'}} \wedge p' \in (P \setminus \{p\}) \\ Post(t, p) & t \in T \wedge t' \in (T' \setminus T) \wedge t' \in [t]_{\equiv_{T'}} \wedge p' = p^c \\ \emptyset & otherwise. \end{cases}$$

Place Consolidation

Definition 9. *Let $S = \langle N, M\rangle$ be a CPN system and let p, p^c be two places in N with $(p, p^c) \in \equiv_P \wedge p \neq p^c$, the consolidation of p^c in p, noted as $C(S, p, p^c)$, is a new CPN system $S' = \langle N', M'\rangle$ s.t*

- N'*: is the net N after removing the place p^c and the transitions* $(p^c)^\bullet \cup {}^\bullet p^c$
- $M' : P' \to \mathcal{M}(cd(p))$ *with* $M'(p) = M(p) + M(p^c)$ *and* $M'(p') = M(p')$ *if* $p' \neq p$.

Example 1. Figure 2-(a) represents the deployment model (empty marking) of the stateful SBP of Figure 1-(a). In this SBP, $s3\_1$ is a stateful service and all others are stateless services. Figure 2-(b) is the resulting system from the duplication of the service $s3\_1$ in (a), $D((a), s3\_1, s3\_2)$. Figure 2-(c) is the consolidation of the service $s3\_1$ in its copy $s3\_2$, $C((b), s3\_2, s3\_1)$.

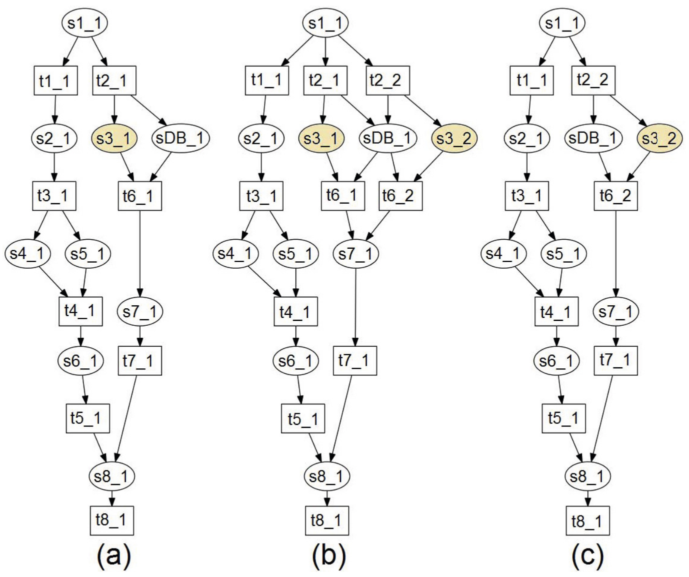


Fig. 2. An example of the elasticity of stateful SBP

4.3 Correctness of Elasticity Operations

In the previous paper [2] we applied the same structural duplication and consolidation operations on classical Petri-net. We proved that this two operations preserve the structural and dynamical properties of the net modulo $\equiv_T$ and $\equiv_P$ relations. This means that the two following properties are still valid for colored Perti-nets:

Property 1. By any transformation of the net using duplication/consolidation operators, we do not lose or create SBP invocations *i.e.,* the load in terms of the number of requests of all the copies of a given service is the same as the load of the original one without duplications/consolidations.

Property 2. The dynamics in terms of load evolution of the original process is preserved in the transformed one *i.e.,* for any reachable load distribution in the original net there is an equivalent (according to property 1) reachable load distribution in the transformed net.

We can also easily deduce that from properties 1 and 2 that duplication/consolidation properties preserve the call sessions dynamics.

5 Framework for the Evaluation of Elasticity Strategies

In order to manage the SBPs elasticity, several strategies can be used [11,9,20]. The strategy is responsible of making decisions on the execution of elasticity

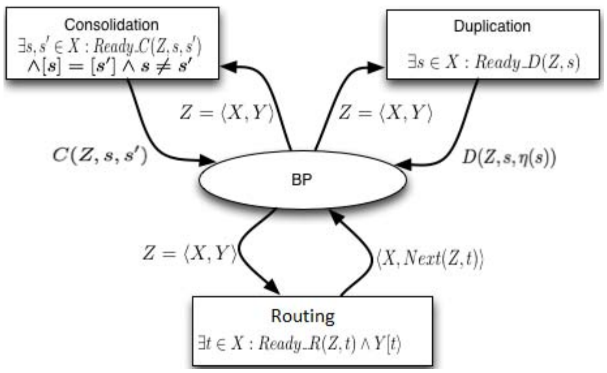

Consolidation
$\exists s, s' \in X : Ready\_C(Z,s,s')$
$\wedge [s] = [s'] \wedge s \neq s'$
Duplication
$\exists s \in X : Ready\_D(Z,s)$
$Z = \langle X,Y \rangle$
$Z = \langle X,Y \rangle$
$C(Z,s,s')$
$D(Z,s,\eta(s))$
BP
$Z = \langle X,Y \rangle$
$\langle X, Next(Z,t) \rangle$
Routing
$\exists t \in X : Ready\_R(Z,t) \wedge Y[t]$

Fig. 3. High level Petri net (HLPN) of the generic controller

mechanisms *i.e.,* deciding when and how to use these mechanisms. So, it is necessary to ensure the precision of a strategy before using it to guarantee its effectiveness. The abundance of possible strategies requires their evaluation and validation. For this reason, we propose a framework, called generic controller, for the evaluation of SBPs elasticity strategies. This generic controller allows the implementation and execution of different elasticity strategies in order to analyze the behavior and the impact of these strategies on SBPs elasticity. Our controller has the capability to perform three actions:

– Routing: Is about the way a load of services is routed over the set of their copies. It determines under which condition we transfer a call. We can think of routing as a way to define a strategy to control the flow of the load *e.g.,* transfer a call iff the resulted marking does not violate the capacity of the services.
– Duplication: Is about the creation of a new copy of an overloaded service in order to meet its workload.
– Consolidation: Is about the removing of an unnecessary copy of a service in order to meet its workload decrease.

If we consider the three actions that can be performed by the elasticity controller, any combination of conditions associated with a decision of routing, duplication and consolidation is an elasticity strategy.

5.1 Formal Description of the Generic Controller

To model our generic controller we used high level Petri nets (HLPN). Due to the lack of space and the heaviness of notations of high level Petri nets, we give here, an informal definition; a more rigorous one can be found in [14]. As classic Petri nets, HLPN is a place-transition bipartite graph. The places are typed, a type can be any set of values (we denote by $type(p)$ the type of the place p). An arc connecting a place p and a transition t is labeled by a multiset of expressions of type $type(p)$. Expression of a type $type(p)$ can be any values of $type(p)$, a variable or any function with domain $type(p)$. The transitions in HLPN can be guarded by a condition *i.e.,* expression of boolean type. The variables that appear in a transition condition and the expressions of its output arcs must be restricted to the variables that appear in the expressions of the input arcs. A marking of HLPN is any function that associates to each place p a multiset of $type(p)$. As in classical Petri nets, a HLPN system is composed of a HLPN and a marking. A transition is fireable, given a marking, iff there is a binding of the variables of its input arcs that validate the condition. The firing of a transition, given a binding, removes the instantiated multisets from input places and adds the instantiated multiset to the output places. Let us mention that the dynamics of an HLPN system can be obtained by computing the reachability graph exactly as classical Petri nets.

The structure of the controller is shown in Figure 3. The controller contains one place (BP) of type CPN system. The marking of this place is modified by the transitions of the controller after each firing:

– Routing: This transition is fireable if we can bind the variable Z to a CPN system $S = \langle N, M \rangle$ where there exists a transition t fireable in S and the predicate $Ready\_R(S, t)$ is satisfied. The firing of the Routing transition adds the CPN system S after the firing of t ($Next(Z, t)$ returns the marking after the firing of t).
– Duplication: This transition is fireable if we can bind the variable Z to a CPN system $S = \langle N, M \rangle$ where there exists a place s and the predicate $ready\_D(Z, s)$ is satisfied. The firing of the Duplication transition adds a new system resulted from the duplication of s in S.
– Consolidation: This transition is fireable if we can bind the variable Z to a CPN system $S = \langle N, M \rangle$ where there exists two copies of the same service, s and s', and the predicate $ready\_C(Z, s, s')$ is satisfied. The firing of the Consolidation transition adds a CPN system resulted from the consolidation of s' in S.

The elasticity conditions that decide when duplicate/consolidate a service are implemented in predicates $ready\_D$ (for duplication) and $ready\_C$ (for consolidation) while the condition that decides on how the service calls are routed is implemented in the predicate $ready\_R$. The execution of controller actions (Duplication/Consolidation and Routing) is performed after checking the guards of the execution of these actions ($ready\_D$, $ready\_C$, $ready\_R$). In our controller, the conditions are generic to allow the use of different elasticity strategies. By instantiating our generic controller, one can analyze and evaluate behaviors and performances of the implemented strategies.

5.2 How to Evaluate Elasticity Strategies with the Framework

The controller has been designed to offer developers a framework to define and evaluate elasticity strategies. In this section, we will show how a strategy developer can instantiate our controller to define elasticity strategies and evaluate their behavior and performance. The execution of the instantiated controller generates the reachability graph of the controller which contains all the possible evolutions of the SBP with respect to the implemented strategy. As we will see, many properties can be checked and many indicators can be observed. The only restriction is to limit the number of calls during the analysis phase. Otherwise this would generate an infinite reachability graph. Note that there are tools to analyze unbounded HLPN nets but do not support any property. In this paper, we propose two kinds of evaluation:

Model Checking Evaluation. Using a HLPN tool, the developer can generate the reachability graph of the controller which can then be analyzed using any model-checker and any temporal logic. Some significant examples of properties are given below:

- QoS violation: Let us assume that we associate for each service a maximal threshold over which its QoS will decrease drastically. Using temporal logic, one can check whether it is possible to reach a situation where one or some services have exceeded their thresholds *i.e.,* transfer a call to a copy of service that has already reached its maximal capacity.
- Blocked services: Let us suppose a routing strategy that allows only transition firing iff the next marking does not exceed the thresholds of some services. We can check if this strategy, coupled with a duplication strategy, would not cause a deadlock in the call transfer *i.e.,* there are fireable transitions in the SBP whereas the routing condition is no longer satisfied.
- Elasticity loop: Duplication and consolidation are costly activities. Given an elasticity strategy, one can check if this strategy can provoke a loop of elasticity *i.e.,* a duplication followed by consolidation of the same service while there is no (or few) calls arrival which means that the strategy causes unnecessary duplication of services.

Performance Evaluation. The developer can also define a set of indicators to evaluate strategies' performance. For example an indicator that computes the number of copies of each service, etc. The value of these indicators will be calculated according to the evolution of the controller *i.e.,* each state of the reachability graph will contain the values of the indicators. The analysis of these indicators allows us to evaluate strategies' performance.

Many parameters can be evaluated, we will focus here on two parameters in order to answer two questions:

- How does the strategy influence the workload of the SBP according to the solicitations?
- How efficient is the resources allocation by the strategy to face the variation of the SBP solicitations?

We measure the workload of the SBP as the average of workloads of its basic services. To do so, we implemented an indicator which stores, at each step of the SBP evolution, the average of the number of running instances on each of its basic services which can be obtained by dividing the number of tokens in the SBP net by the number of places. Concerning resources we consider the number of deployed services copies. We define two indicators. In the first indicator we store, at each step of the SBP evolution, the minimum number of each service copies needed to handle the current number of instances. Note that each copy of services can handle its maximum threshold instances. The second indicator will store the real number of the SBP services produced by a strategy.

5.3 Example of an Application of the Framework

We present hereafter an example, for a proof of concept, of strategies definition and evaluation with the framework. For that, we implemented the controller using the *SNAKES* toolkit. *SNAKES* is a Python library that allows the use of arbitrary Python objects as tokens and arbitrary Python expressions in transitions guards, etc [16].

Experimental Setup. In order to illustrate the feasibility of our approach, we propose here to implement two elasticity strategies inspired from the literature [13,5]. We applied such two strategies on the same SBP system $S = \langle N, M \rangle$ where N is the Petri net of an SBP composed by 3 services $(s1\_1, s2\_1, s3\_1)$ executed in sequence and a data providing service sDB_1 for managing the state of the stateful service $s2\_1$. $M_0 = (0, 0, 0, 0)$ is its initial marking. An invocation (a call) of the SBP is represented by adding a token to a copy of the place $s1\_1$, the invocation takes end by removing a token from a copy of the place $s3\_1$.

We assume in this example that each service of the SBP is provided by a maximum and minimum threshold capacities. Above the maximum threshold the QoS would no longer be guaranteed and under the minimum we have an over allocation of resources. Here are the thresholds:

– Max_t($s1\_1$) = 5. Max_t($s2\_1$) = 3. Max_t($s3\_1$) = 5. Max_t(sDB_1) = 5.
– Min_t($s1\_1$) = 1. Min_t($s2\_1$) = 1. Min_t($s3\_1$) = 1. Min_t(sDB_1) = 1.

Note here that these thresholds represent the maximum number of running instances (calls) on each service. These thresholds are used as scaling indicators by the strategies in order to make their elasticity decisions.

Elasticity Strategies. As we explained previously, the definition of a strategy consists in instantiating the three generic predicates *ready_R*, *ready_D* and *ready_C*. We use two threshold-based scaling algorithms that use the concept of maximum and minimum thresholds to make elasticity decisions. Note that initially these algorithms do not deal directly with the SPB elasticity but use a reasoning that can be used to manage the SPB elasticity. Here after the strategies:

Strategy 1. In [13] an algorithm is proposed to scale up or down an application instance by replication in response to a change in the workload.

- $Ready\_D(S, s) : M(s) \geqslant Max\_t(s) \wedge \nexists s' \in [s] : M(s') < Max\_t(s') \wedge \exists t \in^{\bullet} [s] : M[t\rangle$. It duplicates a copy s of service if all copies of this service have already reached theirs maximal threshold. In addition, there is a service call waiting to be transferred to this copy s.
- $Ready\_C(S, s', s) : M(s') = 0 \wedge M(s) \leqslant Min\_t(s) \wedge \nexists t \in^{\bullet} [s] : M[t\rangle$. It consolidates a copy s' of service if this copy does not contain calls (empty copy) and there is another copy s of the service that has not reached its minimum threshold. In addition, there is not service call waiting to be transferred to this copy s.
- $Ready\_R(S, t) : \forall s \in P : M'(s) < Max\_t(s)$ with $M[t\rangle M'$. It routes a call if this call transfer does not cause a violation of the maximum thresholds of services.

Strategy 2. In [5] a scaling algorithm is proposed to scale up or down the number of instances according to a threshold in each instance.

- $Ready\_D(S, s) : M(s) \geqslant Max\_t(s) \wedge \nexists s' \in [s] : M(s') < Max\_t(s')$. It duplicates a copy s of service if all copies of this service have already reached theirs maximal threshold.
- $Ready\_C(S, s', s) : M(s') = 0 \wedge M(s) \leqslant Min\_t(s)$. It consolidates a copy s' of service if this copy does not contain calls (empty copy) and there is another copy s of the service that has not reached its minimum threshold.
- $Ready\_R(S, t) : \forall s \in P : M'(s) < Max\_t(s)$ with $M[t\rangle M'$. Same routing strategy that strategy 1.

Strategy 3. To illustrate the elasticity impacts, we define also a third strategy that implements only a routing strategy.

- $Ready\_R(S, t) : \forall s \in P : M'(s) < Max\_t(s)$ with $M[t\rangle M'$. Same routing strategy that strategy 1 and strategy 2.

Evaluation of Strategies. In our experiment, we used a Poisson process (with mean 2) to define a scenario of calls arrival on the SBP. This scenario was applied on the three strategies. For each strategy we generate, using the *SNAKES* tool, the reachability graph of the instantiated controller. This graph represents all the possible evolutions of the SBP in terms of routing, duplication and consolidation actions. Hereafter, we present the results of our experiment:

Analysis of Model Checking Evaluation. The analysis of the reachability graph generated by the instantiated controller allows us to deduce some behavioral properties of the execution of the SBP controlled. These properties are summarized in the table below:

	Strategy 1	Strategy 2	Strategy 3
Qos violation	No	No	No
Blocked services	No	No	Yes
Elasticity loop	No	Yes	-

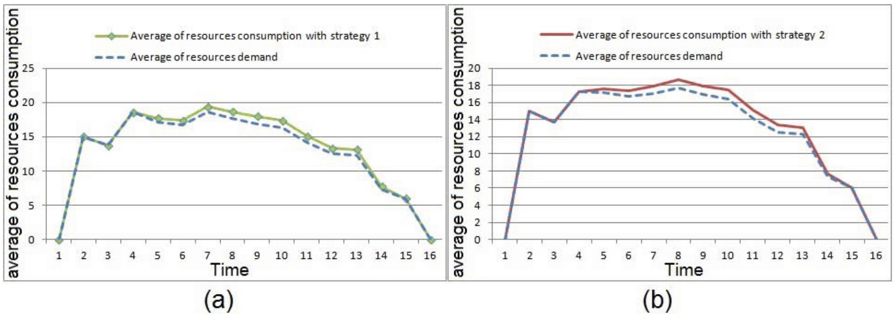


Fig. 4. The average evolution of resources consumption (a) with strategy 1 (b) with strategy 2

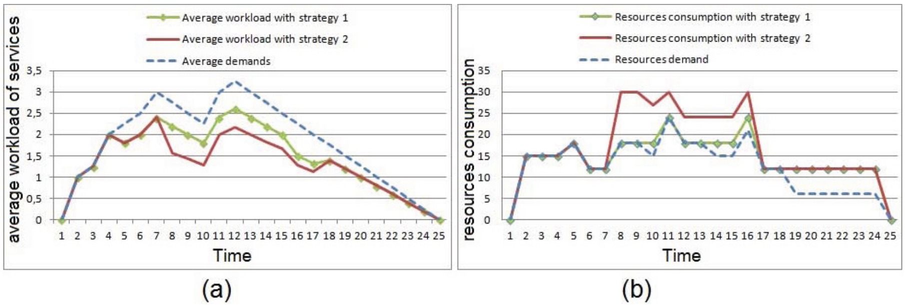


Fig. 5. (a) The evolution of average workload of services (b) The evolution of resources consumption

The analysis of this table allows us to deduce some properties:

- All three strategies avoid QoS violations thanks to the routing strategy used by the three strategies.
- Unlike Strategy 3 that does not implement elasticity mechanisms, Strategies 1 and 2 avoid blocking states by duplicating overloaded services.
- We notice also a difference between the strategies 1 and 2 in the presence of a loop of elasticity. This difference is explained by the conditions of duplication used by both strategies. Indeed, the conditions of duplication used in strategy 1 are more difficult to verify than the conditions of the strategy 2. So, the controller using the strategy 2 will react faster to load increases. This fast reaction in some cases can cause unnecessary elasticity loops.

Analysis of Performance Evaluation. The average evolution of resources consumption with strategies 1 and 2 on all possible executions of the SBP (about 6000 possible executions) is shown in Figure 4. The analysis of this figure shows that both strategies provide the elasticity of SBP by adapting its resources consumption according to the variation of resource demands which avoids resources oversizing. Also, the resources demand never exceeds the resources consumption. This guarantees the availability of resources to provide required QoS and avoid resources over-utilization.

The Figure 5-(a) represents the evolution of average workload of services on one possible execution of the controller. We notice a difference between the strategies in the reactivity to the requests variation. We can see that the strategy 2 is more reactive than the strategy 1. Indeed, the strategy 2 causes more duplication/consolidation than strategy 1. The evolution of resources consumption on one possible execution of the controller is shown in Figure 5-(b). We can see that both strategies adapt the resources consumption according to the resources demand. Using both strategies allows a better efficiency in resources consumption, but there is an under-utilization of resources in some periods.

The analysis of these figures shows a difference between the two strategies. This difference is explained by the conditions of elasticity used in these strategies. Indeed, the conditions of strategy 1 are more difficult to verify than the conditions of strategy 2 (the condition on the existence of service call waiting to be transferred). So, the controller using strategy 2 reacts faster. We can see that the reactivity of strategy 2 does not always mean better efficiency. In fact, this reactivity can cause unnecessary duplication of services.

6 Conclusion

This paper addresses the problem of elasticity of stateful SBPs deployed in Cloud environments. Unlike existing work, our approach tackles the elasticity at the level of SBPs. To perform stateful SBPs elasticity we proposed and formalized using colored Petri nets two operations: Duplication and consolidation. In addition, we have proposed a framework to define elasticity strategies and two approaches to evaluate elasticity strategies. Moreoever, we presented an example for the proof of concept. As perspectives of this work, we are working on the integration of the temporal aspect in our model. We also consider the implementation of the elasticity operations into *CloudServ* (a PaaS under development).

References

1. Amziani, M., Melliti, T., Tata, S.: A generic framework for service-based business process elasticity in the cloud. In: BPM 2012. LNCS, vol. 7481, pp. 194–199. Springer, Heidelberg (2012)
2. Amziani, M., Melliti, T., Tata, S.: Formal modeling and evaluation of service-based business process elasticity in the cloud. In: WETICE (2013)
3. Bi, J., Zhu, Z., Tian, R., Wang, Q.: Dynamic provisioning modeling for virtualized multi-tier applications in cloud data center. In: IEEE CLOUD (2010)
4. Calheiros, R.N., Vecchiola, C., Karunamoorthy, D., Buyya, R.: The aneka platform and qos-driven resource provisioning for elastic applications on hybrid clouds. Future Gener. Comput. Syst. (2012)
5. Chieu, T.C., Mohindra, A., Karve, A.A., Segal, A.: Dynamic scaling of web applications in a virtualized cloud computing environment. In: ICEBE (2009)

6. Duong, T.N.B., Li, X., Goh, R.S.M.: A framework for dynamic resource provisioning and adaptation in iaas clouds. In: CloudCom (2011)
7. Dustdar, S., Guo, Y., Satzger, B., Truong, H.-L.: Principles of elastic processes. IEEE Internet Computing (2011)
8. Dutta, S., Gera, S., Verma, A., Viswanathan, B.: Smartscale: Automatic application scaling in enterprise clouds. In: IEEE CLOUD (2012)
9. Galante, G., de Bona, L.: A survey on cloud computing elasticity. In: IEEE International Conference on Utility and Cloud Computing, UCC (2012)
10. Geelan, J., Klems, M., Cohen, R., Kaplan, J., Gourlay, D., Gaw, P., Edwards, D., de Haaff, B., Kepes, B., Sheynkman, K., Sultan, O., Hartig, K., Pritzker, J., Doerksen, T., von Eicken, T., Wallis, P., Sheehan, M., Dodge, D., Ricadela, A., Martin, B., Kepes, B., Berger, I.W.: Twenty-One Experts Define Cloud Computing
11. Ghanbari, H., Simmons, B., Litoiu, M., Iszlai, G.: Exploring alternative approaches to implement an elasticity policy. In: IEEE CLOUD (2011)
12. Han, R., Guo, L., Guo, Y., He, S.: A deployment platform for dynamically scaling applications in the cloud. In: CloudCom (2011)
13. He, S., Guo, L., Guo, Y., Wu, C., Ghanem, M., Han, R.: Elastic application container: A lightweight approach for cloud resource provisioning. In: AINA (2012)
14. Jensen, K.: Coloured Petri Nets, Basic Concepts, Analysis Methods and Practical Use. Springer, USA (1997)
15. Kranas, P., Anagnostopoulos, V., Menychtas, A., Varvarigou, T.A.: ElaaS: An Innovative Elasticity as a Service Framework for Dynamic Management across the Cloud Stack Layers. In: CISIS (2012)
16. Pommereau, F.: Nets in nets with snakes. In: Int. Workshop on Modelling of Objects, Components, and Agents (2009)
17. Rimal, B.P., Choi, E., Lumb, I.: A taxonomy and survey of cloud computing systems. In: International Joint Conference on INC, IMS and IDC. NCM (2009)
18. Roy, N., Dubey, A., Gokhale, A.: Efficient autoscaling in the cloud using predictive models for workload forecasting. In: IEEE CLOUD (2011)
19. Tsai, W.-T., Sun, X., Shao, Q., Qi, G.: Two-tier multi-tenancy scaling and load balancing. In: ICEBE (2010)
20. Vaquero, L.M., Rodero-Merino, L., Buyya, R.: Dynamically scaling applications in the cloud. SIGCOMM Comput. Commun. Rev (2011)
21. Yangui, S., Mohamed, M., Tata, S., Moalla, S.: Scalable service containers. In: CloudCom (2011)

Controllability of Time-Aware Processes at Run Time

Andreas Lanz[1], Roberto Posenato[2], Carlo Combi[2], and Manfred Reichert[1]

[1] Institute of Databases and Information Systems, University of Ulm, Germany
[2] Department of Computer Science, University of Verona, Italy

Abstract. Companies increasingly adopt process-aware information systems (PAISs) to analyze, coordinate, and monitor their business processes. Although the proper handling of temporal constraints (e.g., deadlines, minimum time lags between activities) is crucial for many applications, contemporary PAISs lack a sophisticated support of the temporal perspective of business processes. In previous work, we introduced *Conditional Simple Temporal Networks with Uncertainty (CSTNU)* for checking controllability of time constraint networks with decision points. In particular, controllability refers to the ability of executing a time constraint network independent of the actual duration of its activities, while satisfying all temporal constraints. In this paper, we demonstrate how CSTNUs can be applied to time-aware business processes in order verify their controllability at design as well as at run time. In particular, we present an algorithm for ensuring the controllability of time-aware process instances during run time. Overall, proper run-time support of time-aware business processes will broaden the use of PAIS significantly.

Keywords: Process-aware Information System, Temporal Perspective, Temporal Constraints, Process Execution, Controllability.

1 Introduction

To stay competitive in their market, companies strive for improved life cycle support of their business processes. In this context, sophisticated IT support for analyzing, modeling, executing, and monitoring business processes becomes crucial [17]. Process-aware information systems (PAISs) offer promising perspectives regarding such a process automation. In particular, a PAIS allows defining a business process in terms of an explicit *process schema*, based on which *process instances* may be created and executed in a controlled and efficient manner [17].

As it has been shown in [12], contemporary PAISs lack a more sophisticated support of the temporal perspective of business processes. However, properly integrating temporal constraints with the design- and run-time components of a PAIS is indispensable to be able to support a greater variety of business processes [3]. Furthermore, in many application domains (e.g., flight planning, patient treatment, and automotive engineering), the proper handling of temporal constraints is crucial for the proper execution and completion of a process [9,4].

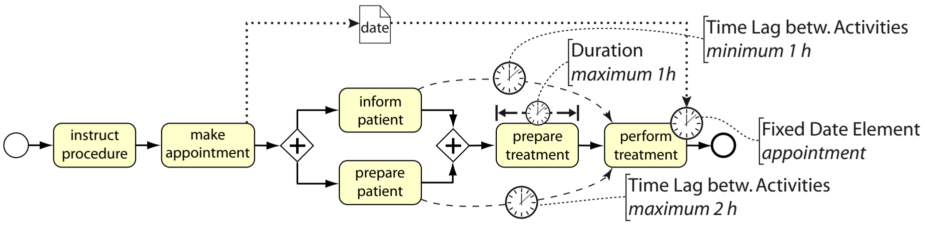


Fig. 1. Illustrating process example with temporal constraints

A fundamental concept related to temporal constraints of process schemas is *controllability* [6]. Controllability is the ability of executing a process schema for all allowed durations of activities and satisfying all temporal constraints. In particular, this ensures that it is possible to execute a process schema without ever having to restrict the duration of an activity to satisfy one of its temporal constraints. Note that this is of paramount importance since activity durations are usually *contingent*. Indeed, it is possible to set up a duration range for any activity, but the PAIS is aware of the effective duration only after activity completion. Checking controllability is especially important at the presence of alternative *execution paths* (e.g., exclusive choice and loops) as each execution path may lead to different temporal properties of the remaining process.

Checking controllability of a process schema solely at design time, however, is not sufficient. In particular, during the execution of corresponding process instances, temporal constraints need to be continuously updated according to the actual durations of already completed activities as well as the decisions made during run time. Further, note that temporal constraints might not be always known at design time. For example, an appointment with a third party (i.e., the date of a respective activity) is usually made during run time (e.g., in the context of a preceding activity) and is specific for each process instance. As example take the patient treatment process depicted in Fig. 1.[1] When considering the temporal perspective of this simplified process, a number of temporal constraints can be observed. In particular, the date for executing activity perform treatment is set by preceding activity `make appointment` and needs to be monitored during run time. In turn, this affects the scheduling of preceding activities due to the other temporal constraints defined, e.g., the patient needs to be prepared *at most 2 hours before* the actual treatment takes place. Hence, activity prepare patient needs to be scheduled in accordance with the appointment of the treatment.

Obviously, the temporal constraints of this process schema are not very strict, i.e., the temporal perspective of the schema is not over-constrained. Nevertheless, when not meeting these constraints, severe consequences might result. For example, if the patient is not informed about the treatment *at least 1 hour* before performing the treatment, the latter must not take place as scheduled for legal reasons and the process has to be aborted. We denote processes obeying a set of defined temporal constraints as *time-aware*, i.e., the execution of a *time-aware process* is

[1] Note that we use an extension of BPMN to visualize temporal constraints in processes (cf. Sect. 3 for details).

Table 1. Process Time Patterns [12]

Category I: Durations and Time Lags	
TP1	Time Lags between two Activities
TP2	Durations
TP3	Time Lags between Events

Category II: Restricting Execution Times	
TP4	Fixed Date Elements
TP5	Schedule Restricted Elements
TP6	Time-based Restrictions
TP7	Validity Period

Category III: Variability	
TP8	Time-dependent Variability

Category IV: Recurrent Process Elements	
TP9	Cyclic Elements
TP10	Periodicity

driven by a set of temporal constraints. In particular, for a time-aware process it is necessary to continuously monitor and update its temporal constraints during run time and hence to re-check controllability of the respective process instance. Accordingly, the contribution of this paper is threefold. First, we discuss fundamental requirements for modeling time-aware processes. In particular, we provide a basic set of modeling elements required for specifying time-aware processes, as well as for executing corresponding instances during run time. Second, we present a mapping of time-aware process schemas to *Conditional Simple Temporal Networks with Uncertainty* (CSTNU) [5], which allows checking their controllability at build time. Third, we present a sophisticated algorithm that enables flexible controllability checking of time-aware processes during run time as well.

Sect. 2 considers existing proposals relevant in the context of time-aware processes. Sect. 3 provides background information on modeling time-aware processes. In Sect. 4, we show how to check controllability of time-aware processes at both design and run time. Sect. 5 provides a short discussion and evaluation of the proposed approach. Finally, Sect. 6 concludes with a summary and outlook.

2 Related Work

In literature, there exists considerable work on temporal constraints for business processes [10,2,4,13]. However, these approaches focus on design time issues, i.e., issues related to the modeling and verification of time-aware processes. By contrast, run-time support for time-aware processes has been neglected by most approaches so far. The mayor novelty of our work is to explicitly address run-time issues of time-aware processes and to elicit requirements emerging in this context.

In [12], 10 time patterns (TP) are presented, which represent temporal constraints relevant for time-aware processes (cf. Table 1). Further, [11] provides a formal semantics of these time patterns. In particular, time patterns facilitate the comparison of existing approaches based on a universal set of notions with well-defined semantics. Moreover, [11,12] elaborate the need for explicitly considering run-time support for time patterns and time-aware processes, respectively.

Marjanovic et al. [13] define a conceptual model for temporal constraints on a process schema. When taking the time patterns as benchmark, [13] considers *time lags between activities* (TP1), activity and process *durations* (TP2), and *fixed date elements* (TP4). Further, a set of rules for verifying time-aware process schemas is presented. However, no run-time support is considered.

Eder et al. [10] use Timed Workflow Graphs (TWG) to represent temporal properties of activities and their control flow relations. [10] considers *time lags between activities* (TP1), activity *durations* (TP2), *fixed date elements* (TP4), and *schedule restricted elements* (TP5). Further, activity durations are assumed to be deterministic, i.e., be the same for all process instances. In [9], same authors suggest a basic run-time support for time-aware processes assuming that the value of a fixed date element is known when creating the process instance, i.e., setting the particular date during run time is not considered. Based on this, "internal deadlines" are calculated for each activity making use of the available temporal information.

Bettini et al. [2] suggest an approach quite different from the above ones. As basic formalism *Simple Temporal Network (STN)* [8] are used. In an STN, nodes represent time points, while each directed edge $a \xrightarrow{v} b$ between time points a and b represents a temporal constraint $b - a \leq v$, where v is a real value. Note that if $v \geq 0$ holds, the constraint represents the maximum allowed delay between b and a; if $v < 0$ holds, it represents the minimum time span elapsed after a before the occurrence of b. Regarding the approach suggested by [2], each activity is represented by two nodes in an STN; i.e., its starting and ending time point. In turn, the edges of the STN represent temporal constraints and precedence relations between the corresponding nodes. [2] considers *time lags between activities* (TP1), activity *durations* (TP2), and *fixed date elements* (TP4). However, run-time support of time-aware processes is not considered.

Combi et al. [4] propose a temporal conceptual model for specifying time-aware process schemas. In particular, *time lags between activities* (TP1), activity *durations* (TP2), *fixed date elements* (TP4), *schedule restricted elements* (TP5), and *periodicity* (TP10) are considered. Additionally, [4] discusses how to check consistency of time-aware processes at design time and argues that different strategies for ensuring consistency of a process instance during run time may be applied, depending on the current kind of consistency of a process schema.

The concept of controllability has been mainly investigated in the AI area in connection with temporal constraint networks: [15] proposes an extension of the STN [8], the *Simple Temporal Network With Uncertainty* (STNU), where the constraints are divided into two classes, the *contingent links* (not under the control of the system) and *requirement links*. In [6], Combi et al. transferred the concept of controllability to time-aware process schemas. In the latter context, informally, controllability is the capability of executing a process schema for all possible durations of all activities and satisfying all temporal constraints. Recently, [5] extended STNU to *Conditional Simple Temporal Network with Uncertainty* (CSTNU) that additionally consider alternative execution paths.

3 Modeling Time-Aware Processes

This section provides basic notions needed for understanding this paper. It further defines a basic set of elements for modeling time-aware processes, which allow for a flexible execution of respective process schemas.

3.1 Process Schema

For each business process to be supported, a *process schema* needs to be defined (cf. Fig. 2). In this work, a process schema corresponds to a directed graph, which comprises a set of *nodes* – representing *activities* and *control connectors* (e.g., Start-/End-nodes, XOR-splits, or AND-joins) – and a set of *control edges* linking these nodes and specifying precedence relations between them as well as loop backward relations. We assume that a process schema is well-structured, i.e., sequences, branchings (i.e., parallel and exclusive choices), and loops are specified in terms of blocks with unique start and end nodes of same type. These blocks—also known as SESE regions [19]—may be arbitrarily nested, but must not overlap; i.e., their nesting must be regular [16]. Fig. 2 depicts an example of a well-structured process schema with the grey areas indicating corresponding blocks. Each process schema contains a unique start and end node and may be composed of the following control flow patterns [1] (cf. Fig. 2): sequence, parallel split (AND-split), synchronization (AND-join), exclusive choice (XOR-split), simple merge (XOR-join), and structured loops. Note that these patterns constitute the core of any process meta model and allow for the flexible composition of more complex structures [14]; further, they cover most processes found in practice [14]. We further assume that the start and the end nodes of a structured loop are distinct from normal XOR-join and XOR-split nodes, i.e., there is an explicit loop construct in the process meta model (like in ADEPT [7]).[2] Finally, to be able to reason about the temporal properties of a loop and to ensure termination of any process schema execution, each loop-end node is augmented with a minimum and maximum number of possible iterations of the respective loop. Note that this does not pose an actual restriction as it is always possible to find a maximum number of iterations high enough to cover any possible case.

In addition to the described control flow elements, a process schema contains process-relevant *data objects* as well as *data edges* linking activities with data objects. More precisely, a data edge either represents a read or write access of the referenced activity to the referred data object.

Process activities may either be *atomic* or *complex.* While an *atomic activity* is associated with an application service, a complex activity refers to a *sub-process.* In our work, we consider complex activities as self-contained, i.e., there is no direct relation between a sub-process and the respective parent process. Therefore, we do not differentiate between atomic and complex activities.

Even though we mostly use the notation defined by BPMN for illustration purpose, the approach described in the following is not specific to BPMN. To set a focus we restrict ourselves to a set of basic modeling elements found in almost every process meta model. Furthermore, to graphically distinguish between loop-blocks and XOR-blocks we use the exclusive gateway symbol with an “X” to represent an XOR-split/-join and the symbol without an “X” to represent loop-start and loop-end nodes.

[2] Note that this does not apply to BPMN causing additional complexity when analyzing processes.

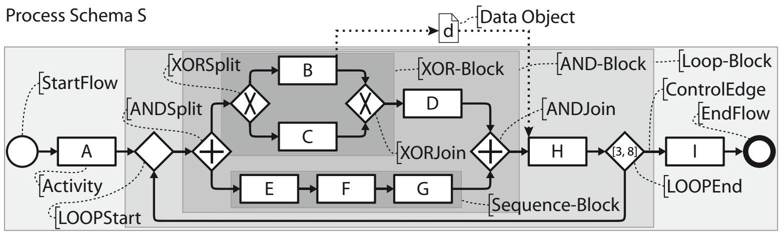


Fig. 2. Core Concepts of a Process Meta Model

At run time, *process instances* are created and executed according to the defined process schema. In turn, *activity instances* represent executions of single process steps (i.e., activities) of such a process instance. If a process schema contains one or more XOR- or LOOP-blocks, not all process instances perform exactly the same set of activities. The concept of *execution path* allows identifying which activities and control connectors are performed during an execution.

Given a process schema P, an **execution path** p (*exe-path*) denotes a connected maximal subgraph of the process schema containing its Start- and End-nodes, in which all XOR-split connectors have exactly one branch and each loop block has a fixed number of repetitions. In particular, each execution path represents one possible execution of the respective process schema. In turn, the set of all *exe-paths* of process schema P is denoted as $ExePaths_P$. An *exe-path* p can be also briefly described by a string containing the activity identifiers of the *exe-path* sorted w.r.t. their execution order and separated by a dash if the order is sequential or by a vertical bar if it is parallel [4]. Considering the schema from Fig. 2, the string A-((B-D)|(E-F-G))-H-I represents an example of an *exe-path*, where A is followed by a parallel execution of two sequential paths (B-D) (i.e., for the XOR-split the upper path is selected) and (E-F-G); then, H and I are sequentially executed. Note that set $ExePaths_P$ may have an exponential cardinality w.r.t. the number of XOR- or LOOP-blocks in the schema.

3.2 Time-Aware Process Schemas

Regarding the time patterns (TP) presented in Sect. 2, to set a focus, this work specifically considers the ones most relevant in practice [12]; i.e., *time lags between two activities* (TP1), *durations* (TP2) of activities, *fixed date elements* (TP4) of activities, and *cyclic elements* (TP9).

An **activity duration** (TP2) represents the time span allowed for executing an activity (or node, in general), i.e., the time span between start and completion of the activity [12]. We assume that each activity of a process schema has an assigned duration. Usually, activity durations are described in terms of minimum and maximum values. Even though these values are known for an activity at design time, the actual duration of a corresponding activity instance is only known at run time after its completion (i.e., it is contingent). Consequently, activity durations must not be restricted when checking controllability at design or run time to satisfy all temporal constraints specified on a process schema. However, in reality, in most cases activity durations are either based on the experience of a domain expert or extracted from process logs. Therefore, activity durations usually represent worst case estimates, i.e., respective maximum durations often cover cases with an exceptionally long duration. Further, execution times of most activities can be shortened if necessary. Accordingly, activity durations may be restricted to some extend during design time when verifying controllability or during run time. In particular, an activity has a *flexible* maximum duration $MaxD_F$. If necessary this may be restricted up to a *contingent* minimum and maximum duration range $[MinD_C, MaxD_C]$, which, in turn, *must* be at least available to the agent when executing the activity. Therefore, activity durations are expressed in terms of restrictable time intervals $[[MinD_C, MaxD_C]MaxD_F]G$ where $1 \leq MinD_C \leq MaxD_C \leq MaxD_F$[3] and G corresponds to the time unit used (i.e., temporal granularity like minutes, hours,...).[4] If a flexible maximum duration is not applicable for an activity, we write $[[MinD_C, MaxD_C]]G$ for short. If a process designer does not set a duration for an activity $[[1, 1]\infty]MinG$ is used as default value, where $MinG$ corresponds to the minimum time unit used by the system. Since control connectors are automatically executed by the PAIS and solely serve structuring purposes, we assume that they have a fixed duration defined by the PAIS (e.g., $[[1, 1]]MinG$) that cannot be modified by the process designer.

Time lags between two activities (TP1) restrict the time span allowed between the starting/ending instants of two activities [12]. Such a time lag may not only be defined between directly succeeding activities, but between any two activities that may be conjointly executed in the context of a particular process instance, i.e., the activities must not belong to exclusive branches. A time lag is visualized by a dashed edge with a clock between the source and target activity (cf. Fig. 3). The label of the edge specifies the constraint according to the following template: $\langle I_S \rangle\, [MinD, MaxD]G\, \langle I_T \rangle$; thereby, $\langle I_S \rangle \in \{S, E\}$ and $\langle I_T \rangle \in \{S, E\}$ mark the instant (i.e., starting/ending) of the source and target activity the time lag applies to; e.g., $\langle I_S \rangle = S$ marks the starting instant of the source activity and $\langle I_T \rangle = E$ the ending instant of the target activity. In turn, the interval $[MinD, MaxD]G$ represents the range allowed for the time span between instants $\langle I_S \rangle$ and $\langle I_T \rangle$ using time unit G. Further, we assume that $-\infty \leq MinD \leq MaxD \leq \infty$ holds. In particular, time lags may be used to specify minimum delays and maximum waiting times between succeeding activities. As example consider the time lag $E[5, 60]min\, S$ between E and F in Fig. 3. It expresses that there is an end-start time lag ($\langle I_S \rangle = E$, $\langle I_T \rangle = S$) of $[5, 60]min$ between the two activities; i.e., the delay between the end of C and the start of F must be at least 5 minutes, while the waiting time between the two must be at most 60 minutes. Finally, it is noteworthy that there exists an implicit

[3] 0 as minimum value for a duration is disallowed since it is not possible to execute an activity/control connector without consuming time.

[4] For the sake of clarity, we assume that all temporal values are expressed using the same granularity; if different granularities are used, it is required to convert them to a common one before executing the process [4].

$E[1, \infty]MinG\, S$ constraint between any couple of directly succeeding activities, i.e., the second activity may only be started after completing the first.

In extension to time lags between activities, **cyclic elements** (TP9) allow process designers to restrict the time span between activity instances belonging to different iterations of a loop structure [12]. This may either be instances of a specific activity or two different activities of the same loop structure. Like time lags, a cyclic element is visualized as dashed edge (with a clock) between the two activities. To differentiate between the two, the label of a cyclic element is extended by a "*" next to the allowed range: $\langle I_S \rangle\ [MinD, MaxD]G^*\ \langle I_T \rangle$. For the sake of simplicity, we only consider cyclic elements between two directly succeeding iterations. However, this is no restriction of the presented algorithms and may be easily extended if necessary.

Finally, **fixed-date elements** (TP4) for activities allow restricting the execution of an activity in relation to a particular date [12],[5] e.g., a fixed-date element may define that the activity must not be started before or must be completed by a particular date. Generally, the value of a fixed-date element is specific to a process instance, i.e., it is not known before creating the process instance or even becomes known only during run time. Therefore, the particular date of a fixed-date element is part of process-relevant data, i.e., it is stored in a data object during run time. When evaluating the fixed-date element, the respective data object is accessed and its current value is retrieved [11]. Graphically, a fixed-date element is visualized by a clock symbol attached to the respective activity (cf. Fig. 3). The label $\langle D \rangle \in \{E_S, L_S, E_E, L_E\}$ attached to this clock corresponds to the activity's earliest start date (E_S), latest start date (L_S), earliest completion date (E_E), or latest completion date (L_E), respectively.

As an example, Fig. 3 shows a process schema exhibiting several temporal constraints. Though some of the symbols used for visualizing the temporal constraints resemble timer events from BPMN, their semantics is quite different and should not be mixed up. Activities A, E, F, and H have an activity duration attached. The one of A, for example, expresses that A has a flexible maximum duration of $25\, min$. This may be further restricted to a contingent minimum duration of $5\, min$ and a maximum duration of $20\, min$ if necessary. In turn, the activity duration of H expresses that H has a contingent minimum duration of $60\, min$ and a maximum duration of $120\, min$, which must not be restricted any further. Between B and G there is a time lag described by $S[30, 120]min\, S$. Additionally, there is a time lag between E and F. Note that, in case a time lag restricts the time span between two directly succeeding activities, for the sake of readability, we attach the clock directly to the control edge and omit the dashed edge of the time lag. However, this is only a graphical simplification and does not change semantics. Next, there is a cyclic element $S[0, 120]min^*\, S$ between B and F. It describes that between the start of any instance of B and the start of an instance of F in the succeeding iteration, there is a time span of at most $120\, min$. Finally, G has a fixed-date element attached to it, whereby label L_E

[5] Fixed-date elements are often referred to as "deadlines". However, this does not completely meet the intended semantics.

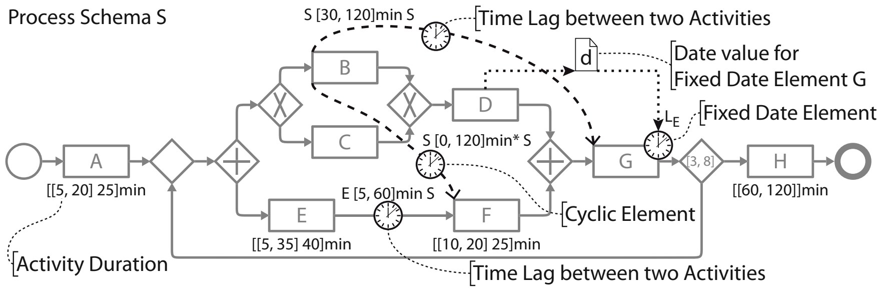


Fig. 3. Process with Temporal Constraints

indicates that the latest end date of the activity is restricted by the temporal constraint. In turn, the date of the fixed-date element is provided by activity D through data object *d*. Particularly, for each iteration of the loop, a new value for the fixed-date element of G is provided by D.

4 Executing Time-Aware Processes

This section introduces and discusses the concept of *controllability* of a time-aware process schema. Controllability guarantees that a process schema can be correctly executed considering all temporal constraints. More specifically, we first introduce the concept of controllability and the controllability check problem. Then, we show how to deal with the execution of a controllable time-aware process schema.

4.1 Controllability of Time-Aware Process Schemas

In general, controllability corresponds to the capability of a PAIS to execute a process schema for all possible contingent durations of all activities while still satisfying all temporal constraints; i.e., controllability ensures that it is possible to execute a process schema without ever having to restrict the contingent duration of an activity to satisfy one of the other temporal constraints.

In particular, an *exe-path* (cf. Sect. 3.1) is executed by performing activities and control connectors, thereby observing any structural and temporal constraints of the process schema. We denote a process schema as *controllable* if it is possible to perform any *exe-path* satisfying all temporal constraints without restricting contingent activity durations involved in the *exe-path*. If there are no time lags (TP1), or fixed date elements (TP4) the schema is controllable. Otherwise, it is necessary to verify and, possibly, adjust time lags in order to guarantee controllability of the process schema.

In [5], authors proposed *Conditional Simple Temporal Network with Uncertainty* (CSTNU) to represent and analyze a network of temporal constraints, where some constraints hold according to specific run-time-evaluated data conditions. Furthermore, they presented a sound algorithm that allows checking the controllability of a CSTNU (possibly adjusting non-contingent constraints) in exponential time w.r.t. the number of conditions in the worst (theoretical) case. Moreover, they provided an implementation of the algorithm showing that it is possible to manage the conditions in an appropriate way in order to avoid the worst case and obtain a practical fast convergence of the algorithm. In this paper, we propose to use the CSTNU checking algorithm to verify the controllability of a process schema extended with the temporal aspects (as discussed in Sect. 3.2). We propose to use CSTNU for two reasons: 1) it is preferable to exploit checking and execution algorithms for a well founded model of extended temporal constraint representation instead of developing new native algorithms, and 2) all other models for temporal constraint representation in literature (e.g., [15,18]) do not allow an effective representation and management of conditional executions with uncertainties.

We now show how to use CSTNU to check the controllability of the considered time-aware process schema (cf. Sect. 3) at design time as well as run time. In particular, a CSTNU is an STN extended with the following constructs:

– *observation nodes*: each observation node is associated with a specific proposition (cf. node $\times_E$ associated with proposition P in Fig. 5-b). The truth-value of the proposition is determined when the node is executed. Informally, an observation node represents the time point at which a relevant information (i.e., proposition) for the execution of the CSTNU is acquired, i.e., it represents the time point a decision is made.
– *labeled nodes and edges*: nodes and edges are characterized by a label consisting of propositions. Such nodes and labels are considered only when the corresponding propositions hold (cf. edge labels β, $P\beta$ and $\neg P\beta$ in Fig. 5-b). Informally, during an execution, the system maintains the truth values of propositions as the execution *scenario*. Then, it considers only nodes and edges having a label consistent with the scenario.
– *contingent links*: a contingent link represents an uncontrollable-but-bounded temporal interval. Each contingent link is described by the range $[x, y]$, $0 < x < y < \infty$, between two time-point variables (nodes), A and C, where C is the so called *contingent* time point. Once A is executed, C is guaranteed to execute such that $C - A \in [x, y]$. However, the particular time at which C executes is uncontrollable.

In the CSTNU model, each edge has a *labeled value* describing the meaning of the corresponding constraint. A *labeled value* is a triple $\langle PLabel,\ ALabel,\ Num\rangle$ where:

– *PLabel* is a propositional label representing a conjunction of propositions. Usually, $\alpha, \beta, \ldots$ are used for representing conjunctions of propositions. $\boxdot$ represents an empty label.
– *ALabel* is an alphabetic label, and either is:
  • an upper-case letter, C, specifying the upper bound of a contingent link;
  • a lower-case letter, c, specifying the lower bound of a contingent link; or
  • $\diamond$, representing no alphabetic label, representing an ordinary STN edge.
– *Num* is a real number, representing the value of the constraint.

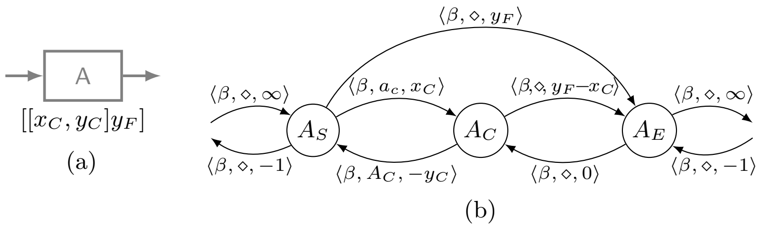

A
$[[x_C, y_C]y_F]$
(a)
$\langle\beta, \diamond, y_F\rangle$
$\langle\beta, \diamond, \infty\rangle$
$\langle\beta, a_c, x_C\rangle$
$\langle\beta, \diamond, y_F - x_C\rangle$
$\langle\beta, \diamond, \infty\rangle$
A_S
A_C
A_E
$\langle\beta, \diamond, -1\rangle$
$\langle\beta, A_C, -y_C\rangle$
$\langle\beta, \diamond, 0\rangle$
$\langle\beta, \diamond, -1\rangle$
(b)

Fig. 4. (a) An activity with a duration. (b) CSTNU translation.

A possible technique to translate a process schema into an equivalent CSTNU consists of mapping each process construct into an equivalent (with respect to the temporal constraints) CSTNU fragment.

More formally,

Theorem 1. *Given a time-aware process schema $\mathcal{P}_S$, there exists a corresponding CSTNU $\mathcal{N}$ such that all temporal features of $\mathcal{P}_S$ are represented in $\mathcal{N}$.*

Proof. The proof is given by construction. In the following, we will provide the mapping for the time-aware process constructs, discussing the most important mappings from the point of view of the temporal behaviour.

First the Start-/End-nodes of the process schema are mapped to two nodes, Z and W, respectively. In turn, an activity and its incoming/outgoing edges are mapped to CSTNU as shown in Fig. 4. In particular, each activity A with duration $[[x_C, y_C]y_F]$ corresponds to three nodes A_S, A_C, and A_E, which represent the *starting time-point*, *contingent ending time-point*, and *ending time-point*, respectively, linked by appropriate edges representing the give duration. The contingent ending time-point A_C is the uncontrollable ending point bounded by the contingent range $[x_C, y_C]$ with respect to the starting time-point A_S. The ending time-point A_E is the controllable ending point that allows the run-time algorithm to consider the flexible maximum duration y_F, represented by a upper-bound constraint between A_S and A_E with Plabel $\langle\beta, \diamond, y_F\rangle$. Edges between A_S and A_C represent contingent links in CSTNU; edges between A_C and A_E represent ordinary constraints; finally, any incoming (outgoing) edge of the activity is translated as a pair of edges representing the implicit temporal constraint $[1, \infty]$ between the ending (starting) node of the predecessor (successor) activity and the starting (ending) node of the considered activity.

The next construct to be considered is the XOR-split. Fig. 5 depicts the translation of an XOR-split evaluating a proposition P. The connector corresponds to two nodes, $\times_S$ and $\times_E$, representing its starting and ending instants, respectively. These nodes are connected by two edges representing the implicit duration range $[1, 1]$. $\times_E$ is the *observation* node for proposition P. All edges and nodes corresponding to activities, connectors and control edges in the XOR-block are suitably labeled with P or $\neg P$ depending on the branch they belong to. The corresponding XOR-Join is translated in a similar way, but the outgoing edge then corresponds to two edges in which propositions $P/\neg P$ are not present (cf. Fig. 6).

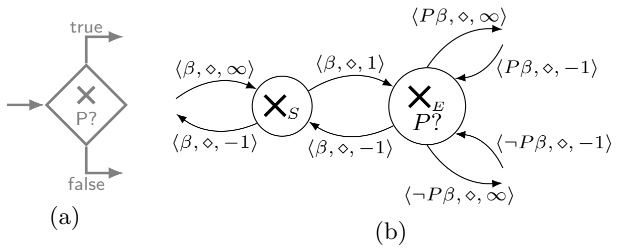

true
P?
false
(a)
$\langle \beta, \diamond, \infty \rangle$
$\langle \beta, \diamond, -1 \rangle$
$\langle \beta, \diamond, 1 \rangle$
$\langle \beta, \diamond, -1 \rangle$
$\langle P\beta, \diamond, \infty \rangle$
$\langle P\beta, \diamond, -1 \rangle$
$\langle \neg P\beta, \diamond, -1 \rangle$
$\langle \neg P\beta, \diamond, \infty \rangle$
$P?$
(b)

Fig. 5. (a) XOR-split with implicit duration [1, 1]. (b) CSTNU translation.

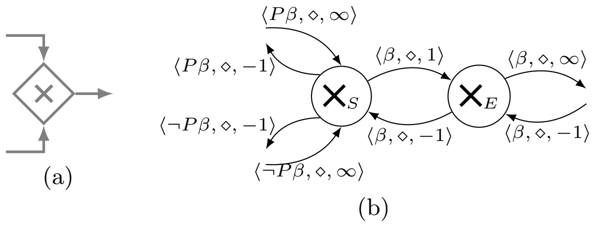

(a)
$\langle P\beta, \diamond, \infty \rangle$
$\langle P\beta, \diamond, -1 \rangle$
$\langle \neg P\beta, \diamond, -1 \rangle$
$\langle \neg P\beta, \diamond, \infty \rangle$
$\langle \beta, \diamond, 1 \rangle$
$\langle \beta, \diamond, -1 \rangle$
$\langle \beta, \diamond, \infty \rangle$
$\langle \beta, \diamond, -1 \rangle$
(b)

Fig. 6. (a) XOR-Join with implicit duration [1, 1]. (b) CSTNU translation.

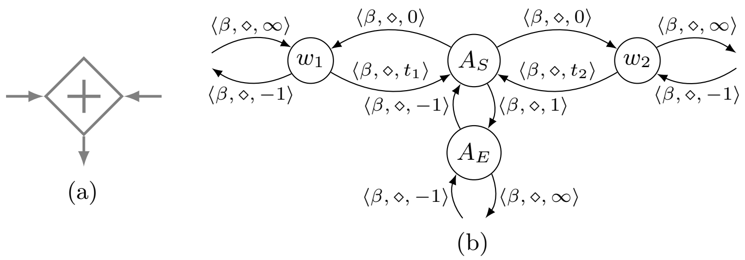

(a)
$\langle \beta, \diamond, \infty \rangle$
$\langle \beta, \diamond, -1 \rangle$
w_1
$\langle \beta, \diamond, 0 \rangle$
$\langle \beta, \diamond, t_1 \rangle$
A_S
$\langle \beta, \diamond, 0 \rangle$
$\langle \beta, \diamond, t_2 \rangle$
w_2
$\langle \beta, \diamond, \infty \rangle$
$\langle \beta, \diamond, -1 \rangle$
$\langle \beta, \diamond, -1 \rangle$
$\langle \beta, \diamond, 1 \rangle$
A_E
$\langle \beta, \diamond, -1 \rangle$
$\langle \beta, \diamond, \infty \rangle$
(b)

Fig. 7. (a) AND-Join connector. (b) CSTNU translation.

Another construct to be considered is the AND-Join connector (the mapping of the AND-Split connector is straightforward). Fig. 7-(a) depicts an example of an AND-Join connector with two incoming flows. The execution of this connector requires waiting for all incoming flows: after the last incoming flow has been triggered, the AND-Join is executed before triggering its outgoing edge. The key aspect of the AND-Join is that its incoming flows may arrive at different instants. Therefore, each incoming flow is connected to a "wait" node that, in turn, is connected to A_S by two edges, as depicted in Fig. 7-(b). For example, the constraint $\langle \beta, \diamond, 0 \rangle$ corresponding to edge (A_S, w_1) represents that A_S must be after w_1, while the constraint $\langle \beta, \diamond, t_1 \rangle$ on edge (w_1, A_S) represents the possible maximum delay due to the execution of w_2; the value t_1 is automatically determined by the controllability check at design time. One can easily show that if there are more incoming flows in the original AND-Join connector, it is possible to translate it using a sequence of pairs of "wait" nodes properly connected before A_S.

Since for each loop the maximum number of iterations is known, any process schema containing loops can be rewritten into a loop-free one. For this, the loop

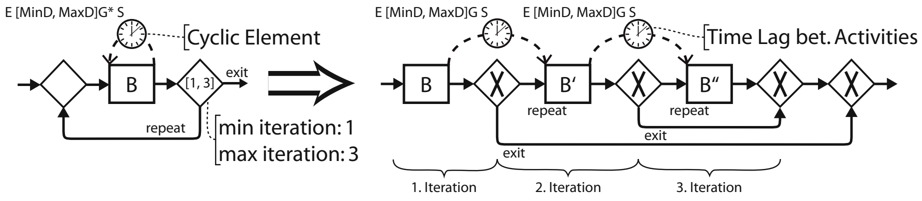


Fig. 8. Converting a loop to a set of consecutive XORs

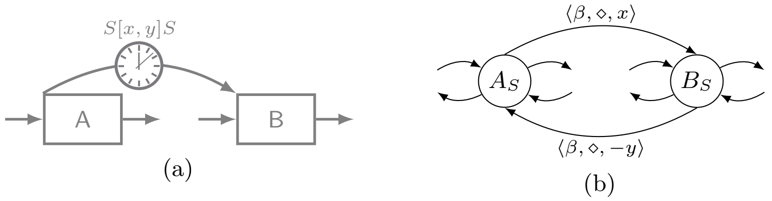

$S[x, y]S$
A
B
(a)
$\langle\beta, \diamond, x\rangle$
A_S
B_S
$\langle\beta, \diamond, -y\rangle$
(b)

Fig. 9. (a) Start-start time lag between activities A and B. (b) CSTNU translation.

block is replaced by a block containing clones of the original loop body, which are then linearly connected: all Loop-start connectors are removed and each Loop-end connector becomes an XOR-split connector with one edge connected to the first node of the following clone and the other connected to an XOR-join connector inserted after the last clone. The condition of these XOR-split connectors corresponds to the original condition of the Loop-end connector. Moreover, cyclic elements (TP9) are transformed to time lags (TP1) between the clones of respective activities. Fig. 8 shows an example of such an unfolding of a loop with at maximum three iterations. Consequently, a loop may be translated to CSTNU the same way as XOR-splits and -joins (cf. Figs. 5 and 6).

Next, a time lag between two activities (TP1) can be translated into a pair of CSTNU edges between between the starting/ending nodes of the two activities. In particular, for each time lag $\langle I_S\rangle\,[x, y]G\,\langle I_T\rangle$, depending on the value of $\langle I_S\rangle/\langle I_T\rangle$, a pair of ordinary constraint edges $\langle\beta, \diamond, x\rangle$ and $\langle\beta, \diamond, -y\rangle$ is added between the starting/ending node of the source and the starting/ending node of the target activity. Fig. 9 depicts this transformation exemplarily for a start-start time lag (i.e., $\langle I_S\rangle = S$ and $\langle I_T\rangle = S$).

The last major construct we consider for the translation is the fixed date constraint (TP4). It can be translated into a CSTNU edge between the start node Z of the process and the node representing the starting/ending node of the activity once the starting time d_Z of the process and the fixed date $d_{\langle D\rangle}$ value are known, i.e., the fixed date is represented as the time lag between the start of the process instance and the respective fixed date. Fig. 10 depicts the details of the translation of the different fixed date elements according to the constraint label $\langle D\rangle \in \{E_S, L_S, E_E, L_E\}$. This completes the proof. □

4.2 Run-Time Controllability Check

Controllability of a process schema must be checked both at design and run time. At design time, such a controllability check allows guaranteeing that the

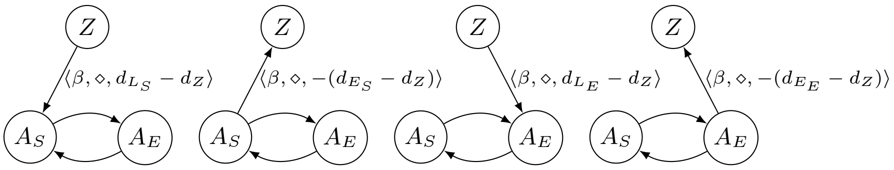

Z
$\langle\beta, \diamond, d_{L_S} - d_Z\rangle$
$\langle\beta, \diamond, -(d_{E_S} - d_Z)\rangle$
$\langle\beta, \diamond, d_{L_E} - d_Z\rangle$
$\langle\beta, \diamond, -(d_{E_E} - d_Z)\rangle$
A_S
A_E

Fig. 10. Translating possible fixed date constraints on an activity A when date d_Z of the process start and date $d_{E_S}|d_{L_S}|d_{E_E}|d_{L_E}$ of the constraint are known.

design phase is sound as any process instance may be executed meeting the given temporal constraints. At run time, the controllability check updates the temporal network according to the real durations of already executed activities, to the possible fixed date constraints, and to the current execution path. In particular, controllability has to be checked after the completion of each contingent activity.

When creating a process instance, a copy of the CSTNU created at design time is made. Next, any fixed date constraint known at process creation time is considered by adding the respective constraint(s) (cf. Sect. 4.1). This CSTNU is then updated according to the starting time of the process instance by executing a controllability check. Thus, the time frame for starting the first activity, determined by the previous check, is fixed and is used by the execution engine.

When completing an activity, its real duration and possible date value for fixed date constraints become known. Hence, in order to maintain the right time frames of unexecuted activities, it is necessary to update and check the CSTNU after the completion of each activity. In particular, the check may result in an update of the time frames of the remaining activities. Then, the engine determines the following activities to execute, taking into account the order given by the schema and the time frames provided by the updated CSTNU. Note that there are different possible execution strategies for choosing the exact instant to start an activity within its time frame. In the following we presume an execution strategy that allows executing an activity/connector as soon as it becomes *enabled*. An activity/connector is enabled when all its previous activities/connectors (w.r.t. the process schema) have been executed and all constraints involving the considered activity/connector are met. However, if—due to some delay—an activity is not started within its time frame or if it takes longer than permitted by the CSTNU, the process instance (potentially) becomes uncontrollable (i.e., it can no longer be guaranteed that the process may be completed without violating any time constraint). In this case, time-specific exception handling (i.e., escalations) should be triggered [9].

Let us label the CSTNU controllability checking algorithm as CSTNU-CC. For a process instance, the check of its controllability during run time, which we call *TimeAwareProcessControllabilityCheck*, works as follows:

1. Once the starting time of a process instance is set, all fixed date constraints whose date is also known at process creation time are translated into equivalent constraints w.r.t. to the starting date of the process instance (cf. Fig. 10) in the CSTNU instance. The controllability of the CSTNU instance is then

checked to ensure that any fixed date constraint is consistent with the execution time of the process instances.
2. Each time an activity is completed, the CSTNU instance is updated using the real duration of the completed activity and re-checked to propagate the modified constraints.
3. After the completion of any activity producing a date value for a fixed-date constraint, the CSTNU instance must be updated adding the equivalent constraint(s) as shown in the previous section (cf. Fig. 10) and then re-checked. For networks being controllable at design time it is noteworthy that, besides activity executions not being started within the time frames given by the CSTNU or not respecting the given activity duration constraint, only fixed date constraints could make the network uncontrollable at run time.
4. Each time an XOR-split is completed, the CSTNU instance must be updated by removing all nodes and edges belonging to skipped XOR branches. In particular, the execution of an XOR-split determines the one of the corresponding observation node. Such observation node determines the truth value of the associated proposition. Therefore, the execution *scenario* is updated and all nodes/edges not consistent with it are removed. Note that, due to the removal of the skipped XOR branches, the time frames of unexecuted activities may be potentially relaxed.

Fig. 11 depicts the pseudocode of algorithm *TimeAwareProcessControllability-Check*. It checks the controllability of the corresponding CSTNU network during run time according to the above approach.

Let us consider in a more detailed way how many times *TimeAwareProcess-ControllabilityCheck* is executed for a process instance. Let k be the number of XOR-split connectors and a the number of activities. *TimeAwareProcessControllabilityCheck* is then called $k + a$ times in the worst case (sequential XOR-splits containing activities in only one branch each). Each *TimeAwareProcessControllabilityCheck* execution corresponds to a single execution of the CSTNU-CC algorithm. The latter has an exponential-time complexity w.r.t. to the number k' of unexecuted XOR-splits, where $k' = k, k - 1, \ldots, 1$. Each unexecuted XOR-split determines at least 2 different outgoing execution paths and, thus, there exist at least $2^{k'}$ different possible execution paths in the process instance. Since k decreases linearly during the execution (worst case), the complexity of the following CSTNU-CC executions—after each XOR-split—decreases exponentially. As for the CSTNU-CC algorithm [5], the real time complexity of the controllability check algorithm is much lower than the theoretical worst case. First experiments we performed have confirmed this.

5 Discussion

Recently, we identified a set of time patterns for evaluating the support of the temporal perspective in PAIS [12,11]. Empirical evidence we gained in case studies has confirmed that these time patterns are common in practice and required for properly modeling the temporal perspective of processes in a variety of domains [12]. In particular, our case studies revealed the need for a comprehensive

```
Procedure TimeAwareProcessControllabilityCheck(event)
if (event == "end of activity" $A_i$) then
    $d_i$ = real duration of $A_i$;
    Update all constraints involving $A_i$ using $d_i$;
    foreach $fixedDate_{ij}$ = fixed date value known after the execution of $A_i$ do
        if ($fixedDate_{ij} \neq null$) then
            Update all constraints requiring $fixedDate_{ij}$;
    Execute CSTNU-CC on the updated network;
    if (network is not controllable) then
        Throw an exception;
        return Network not controllable
if (event == "end of XOR-split" $X_i$) then
    $d_i$ = real duration of $X_i$;
    Update all constraints involving $X_i$ using $d_i$;
    $b_i$ = selected branch;
    Remove all branches (edges and nodes) $\neq b_i$;
    Execute CSTNU-CC on the updated network;
return Network controllable;
```

Fig. 11. Pseudo code for controllability checking of time-aware processes during run time

design- and run-time support of time-aware process. This has been confirmed in a number of discussions, we had with process engineers when validating the formal semantics of our time patterns [11].

To ensure the soundness of a process schema and hence robust and correct execution of corresponding process instances, the controllability of their temporal constraints must be checked. In general, to solely verify time-aware process schemas at design time is neither sufficient nor completely possible. Recent work has shown that certain time patterns (i.e., temporal constraints) cannot be verified at design time, as they are specific for each process instance [12].

The time patterns considered in this paper were selected based on the empirical evaluation we conducted as part of [12]. In particular, they are the ones most commonly required in practice. Also, note that the particular patterns provide a reference time frame for any instance based on respective time-aware process schemas. To verify and test the practical usability of the proposed transformation and respective algorithms, we implemented a proof-of-concept prototype as part of CSTNUEditor [5]. It allows us to create a CSTNU instance based on a process schema and to check its controllability. First tests have shown that the algorithm finds the solution in an average number of iterations one order of magnitude smaller than the theoretical estimated upper bound. As an example, Fig. 12 depicts the CSTNUEditor screenshot of the controllability check of the process schema of Fig. 1: the left part of the screen shows the CSTNU corresponding to the process schema (green boxes contain nodes and constraints corresponding to original activities), while the right part depicts all the

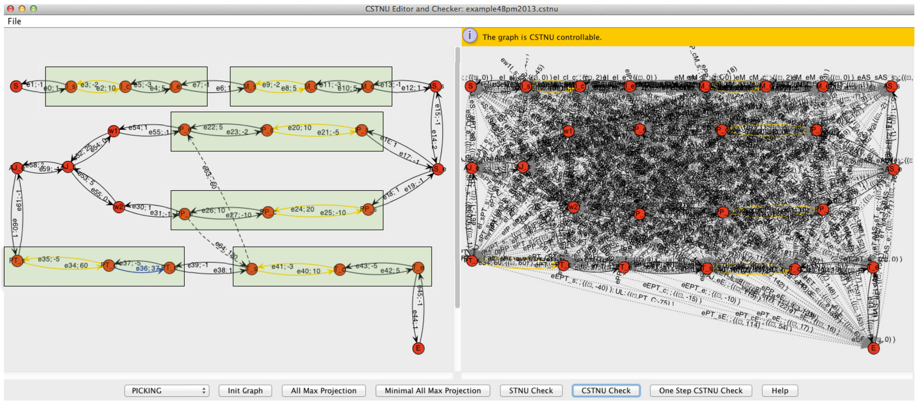

CSTNU Editor and Checker: example4Bpm2013.cstnu
File
The graph is CSTNU controllable.
PICKING
Init Graph
All Max Projection
Minimal All Max Projection
STNU Check
CSTNU Check
One Step CSTNU Check
Help

Fig. 12. Time-aware Process controllability check in CSTNUEditor

computed temporal constraints between nodes together with the overall analysis result showing that the process is controllable. Moreover, we have implemented most of the time patterns as part of a proof-of-concept prototype based on the AristaFlow BPM Suite [7]. In this context, we are working on integrating the presented algorithms for controllability checking at build time and during run time to obtain a *time- and process-aware information system.*

6 Summary and Outlook

Time is a fundamental concept regarding the support of business processes. In a real world environment, where even small delays may cause significant problems, it will be crucial for any enterprise to be aware of the temporal constraints of its business processes as well as to control and monitor them during process execution. Particularly, it must be ensured that no temporal constraint is violated during run time. This paper considered fundamental requirements for the run-time support of time-aware processes.

First, we defined a set of basic elements for modeling time-aware process schemas, which allow for a flexible execution of related processes instances. Specifically, we considered the need for dynamically adapting process instances to a specific context, e.g., we consider temporal constraints whose parameters only become known during process execution. The proposed set of temporal constraints is independent from a particular process modeling language.

Second, we presented a transformation of time-aware process schemas to *Conditional Simple Temporal Networks with Uncertainty* for checking controllability of respective process schemas at design time. We then demonstrated how this can be also applied for ensuring the controllability of corresponding time-aware process instances during run time. In particular, we presented an algorithm for controllability checking during run time and discussed its complexity.

In future work, we will investigate the complexity of the presented controllability checking algorithm in more detail. In this context, we will examine how process abstractions and process views as well as predictive knowledge about

XOR decisions may be applied to reduce the complexity of this algorithm. Furthermore, we will fully integrate the presented approach with the AristaFlow BPM Suite [7]. Finally, we will evaluate the impact, process changes have on time-aware processes and respective temporal constraints.

References

1. van der Aalst, W.M.P., ter Hofstede, A.H.M., Kiepuszewski, B., Barros, A.P.: Workflow patterns. Distributed and Parallel Databases 14(1), 5–51 (2003)
2. Bettini, C., Wang, X.S., Jajodia, S.: Temporal reasoning in workflow systems. Distributed and Parallel Databases 11(3), 269–306 (2002)
3. Combi, C., Gozzi, M., Juarez, J.M., Oliboni, B., Pozzi, G.: Conceptual modeling of temporal clinical workflows. In: Proc. TIME 2007, pp. 70–81. IEEE (2007)
4. Combi, C., Gozzi, M., Posenato, R., Pozzi, G.: Conceptual modeling of flexible temporal workflows. ACM Trans. Auton. Adapt. Syst. 7(2), 19:1–19:29 (2012)
5. Combi, C., Hunsberger, L., Posenato, R.: An algorithm for checking the dynamic controllability of a conditional simple temporal network with uncertainty. In: Proc. Int. Conf. Agents & Art. Int. (ICAART), vol. 2, pp. 144–156. SciTePress (2013)
6. Combi, C., Posenato, R.: Controllability in temporal conceptual workflow schemata. In: Dayal, U., Eder, J., Koehler, J., Reijers, H.A. (eds.) BPM 2009. LNCS, vol. 5701, pp. 64–79. Springer, Heidelberg (2009)
7. Dadam, P., Reichert, M.: The ADEPT project: A decade of research and development for robust and flexible process support - challenges and achievements. Computer Science - R&D 22(2), 81–97 (2009)
8. Dechter, R., Meiri, I., Pearl, J.: Temporal constraint networks. Artif. Intell. 49(1-3), 61–95 (1991)
9. Eder, J., Euthimios, P., Pozewaunig, H., Rabinovich, M.: Time management in workflow systems. In: Proc. BIS 1999, pp. 265–280. Springer (1999)
10. Eder, J., Gruber, W., Panagos, E.: Temporal modeling of workflows with conditional execution paths. In: Ibrahim, M., Küng, J., Revell, N. (eds.) DEXA 2000. LNCS, vol. 1873, pp. 243–253. Springer, Heidelberg (2000)
11. Lanz, A., Reichert, M., Weber, B.: A formal semantics of time patterns for process-aware information systems. Tech. Rep. UIB-2013-02, University of Ulm (2013)
12. Lanz, A., Weber, B., Reichert, M.: Time patterns for process-aware information systems. Requirements Engineering (2012)
13. Marjanovic, O., Orlowska, M.E.: On modeling and verification of temporal constraints in production workflows. Knowl. and Inf. Syst. 1(2), 157–192 (1999)
14. Mendling, J.: Metrics for process models: empirical foundations of verification, error prediction, and guidelines for correctness. Springer (2009)
15. Morris, P.H., Muscettola, N., Vidal, T.: Dynamic control of plans with temporal uncertainty. In: IJCAI, pp. 494–502 (2001)
16. Reichert, M., Rinderle, S., Kreher, U., Dadam, P.: Adaptive process management with ADEPT2. In: Proc. ICDE 2005, pp. 1113–1114. IEEE (2005)
17. Reichert, M., Weber, B.: Enabling Flexibility in Process-aware Information Systems: Challenges, Methods, Technologies. Springer (2012)
18. Tsamardinos, I., Vidal, T., Pollack, M.E.: CTP: A new constraint-based formalism for conditional, temporal planning. Constraints 8, 365–388 (2003)
19. Vanhatalo, J., Völzer, H., Leymann, F.: Faster and more focused control-flow analysis for business process models through SESE decomposition. In: Krämer, B.J., Lin, K.-J., Narasimhan, P. (eds.) ICSOC 2007. LNCS, vol. 4749, pp. 43–55. Springer, Heidelberg (2007)

Investigations on User Preferences of the Alignment of Process Activities, Objects and Roles

Agnes Koschmider[1], Simone Kriglstein[2,3], and Meike Ullrich[1]

[1] Institute of Applied Informatics and Formal Description Methods
Karlsruhe Institute of Technology, Germany
{agnes.koschmider,meike.ullrich}@kit.edu
[2] SBA Research[*], Vienna, Austria
[3] University of Vienna, Faculty of Computer Science
skriglstein@sba-research.at

Abstract. Numerous attempts have been made to research the variety of different influences on the understandability of process models. Common to all of these attempts is the limitation to the process model itself. Little empirical effort is spent on investigating the understandability of the alignment of process activities, objects, and roles. This paper tackles this issue and empirically studies preferences of how to visually align process activities with objects and roles. In particular, three visualization techniques are evaluated in order to support the combination of the object and organization units with their corresponding process model elements. The empirical study provides a strong support for the visualization of a process model that is disburdened from context information such as objects used and roles involved and thus is reduced to the sole visualization of process activities and its control-flow.

Keywords: process modeling, understandability, model visualization.

1 Introduction

In the context of Business Process Modeling, models have several intentions. For instance, they act as discussion foundation or are used as basis for the implementation of an information system. Thus, process models are usually not designed for personal use but rather need to be understood by a variety of peers. Consequently, apart from particular intentions, process models should be designed in a way that they are (easily) understandable by a heterogeneous set of users.

Effects on understandability of process models are subject of a plethora of academic discussions. In particular, it has been diagnosed that some syntax elements such as the number of routing symbols or the design of routing symbols [3,4,12] have implications on process model understandability. Thus, the design of process models has a significant effect on user's comprehension. To date, the debate on understandability mainly focuses on the process model itself. To allow for a more comprehensive

[*] The research was funded by COMET K1, FFG - Austrian Research Promotion Agency.

insight into impacts on understandability of process models, additional information about the process model context (beyond activities and control-flow) is required. A process model is jointly linked with further model types. In addition to process activities, a process model aligns e.g., data used and/or the roles involved. The latter are specified in separate models and are connected through assigning this information to process activities. To complement debates on process model understandability, an advanced view on the visual alignment between process models and its related model types is essential.

Such an investigation requires considering practical implementation of aligning process activities and its related information. One option is displayed in Fig. 1, which exemplarily shows the visual assignment of objects (data) and organization units (roles) to process activities in one window. The user views all corresponding information of each process activity within one window ("Process Window"). This kind of integrated visualization technique (further in this paper called *single view*) is used by several modeling languages (see section 2.) To view relationships between the objects and organization units themselves, the user has to open separate windows ("Object Window" and "Organization Window" respectively).

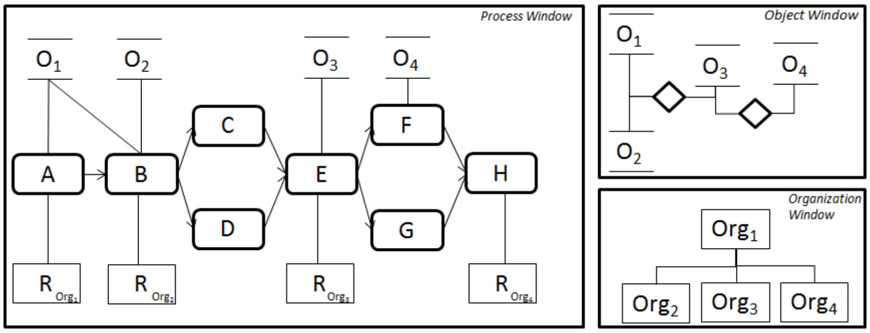


Fig. 1. Alignment of process activities, object and organizational units

The objective of this paper is to understand *visualization preferences* of the alignment of a process model and its corresponding objects and roles (what kind of visual alignment does the user prefer?). Exemplarily, we investigate preferences for the visualization of an alignment between process activities, objects and organization units (roles). The latter can be replaced by any other model type (e.g., risk model).

To achieve this objective, we proceed as follows. Next section summarizes visual alignment techniques implemented for common process modeling languages in practice. Section 3 discusses three visualization techniques, which mainly base upon these practical implementations. To provide evidence about the three visualization techniques, an empirical study has been conducted. The results of the study are presented and discussed in section 4. The topic addressed in this paper is highly related to discussions on process model understandability and visualization techniques for process models. These related approaches are tackled in section 5. The paper concludes with an outlook in section 6.

2 Alignment Visualization in Practice

To investigate visualization preferences, we studied the alignment visualization of common process modeling languages, here Business Process Modeling Notation (BPMN), Event-driven Process Chain (EPC), Unified Modeling Language (UML) and Petri nets.

In EPC (respectively eEPC) process objects and organizational units are linked to the process activities through arcs. Undirected arcs are used to link an organizational unit to a process activity for the definition of roles, whereas information objects are linked to activities with directed arcs to display the type of operation (read/write). Table 1 shows an example of an eEPC model created with ARIS Express. It displays a process activity A executed by a member of the role O1, which reads a document D1 and writes a document D2.

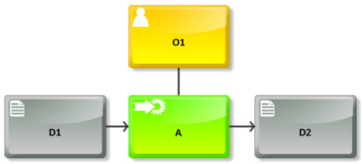
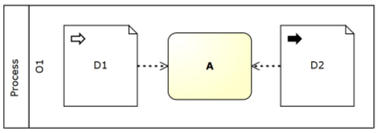
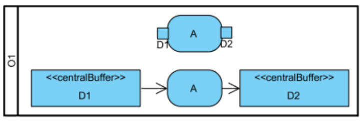
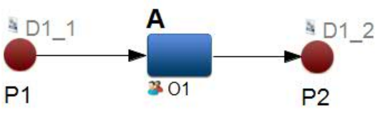
The BPMN 2.0 specification defined by the Object Management Group (OMG) provides several graphical elements to display process objects (data objects or data store). These objects can also be assigned in the control-flow model and linked to process activities, processes or global tasks using a data association arc. Organizational units however are embedded with the use of lanes, which is a fundamentally different solution than in eEPC. For each role in the process, a lane is created and process activities are distributed across the lanes according to the underlying role model for the task execution. Table 1 shows an example BPMN model for the visual alignment of a process activity A, which is executed by a member of role O1 and uses data object D1 as input and data object D2 as output. The model has been created with the Signavio Process Editor.

The current UML 2.4.1 specification provided by the OMG offers 14 different diagram types, which are designed mainly for modeling purposes from the field of system and software engineering. Most suitable for the modeling of business processes are UML activity diagrams. With UML 2.0 four different types of object nodes have been introduced for the representation of objects. So-called Pins can be used to represent the object flow between process activities (called actions in UML). They are visualized as small squares directly attached to action elements. It is also possible to use a CentralBufferNode (or the respective persistent equivalent DataStoreNode), which represents an object node storing objects independent from actions. Also, ActivityParameterNodes can be used to represent input or output objects to an activity (which in UML corresponds to a subprocess consisting of atomic actions). It is therefore necessary to understand the different semantics of the various object types in order to choose an appropriate element for the visual representation. Like BPMN, UML uses lanes (called partitions) to describe role distribution across the process activities. Table 1 contains two example UML models created with Visual Paradigm for UML. It demonstrates the two fundamental visualization styles UML offers. The examples show a process activity A, which is executed by role O1 and reads an object D1 and writes an object D2 as in the previous examples.

Finally, Petri nets are bipartite directed graphs consisting of places and transitions. In so-called high-level Petri nets places contain tokens, which are interpreted as distinguishable process objects travelling dynamically through the process model.

The token itself can be represented by different types of information. In Petri nets, the alignment between process activities, objects and roles is not represented with arcs or additional routing constructs (as it is done in the previously described notations). Instead, related context information can be accessed interactively by clicking through each process activity if supported by the modeling tool. The Petri net based modeling tool Horus Business Modeler allows visualizing the aligned information of process activities without browsing through the activities. The example Petri net model in Table 1 shows a process activity A being executed by role O1. Places P1 and P2 are containers for objects of type D1_1 and D1_2, which indicates that activity A reads object D1_1 and writes object D1_2.

Table 1. Visualization examples for the alignment of activities, objects and roles in different process modeling languages

Modeling Language	**Visualization example**	**Short description**
eEPC		Activities (A) are connected to objects (O) through directed arcs representing the access type (read/write), roles (D) are assigned through undirected arcs.
BPMN		Activities are connected to objects through directed arcs, the access type is represented by the icon style (white arrow: read; black arrow: write). Roles are displayed via lanes.
UML	 O1 A D1 D2 <<centralBuffer>> D1 A <<centralBuffer>> D2	Objects are either directly attached to activities or connected with directed arcs. The above example shows pins, the example below shows buffer nodes (CentralBufferNode). Roles are displayed via lanes.
Petri net		Objects can be stored in places, which are connected to activities through directed arcs. There is no explicit support for roles, however some tools realize the integrated display with an extended annotation of the model.

However, the sole inspection of a process model itself is not sufficient to give the user a complete picture about the business process. Independent from the modeling language used to design the process control-flow, additional but separate models must be studied (object or organizational perspective) in order to fully understand the internal structure. Fig. 2. shows such additional models for the Petri net from Table 1.

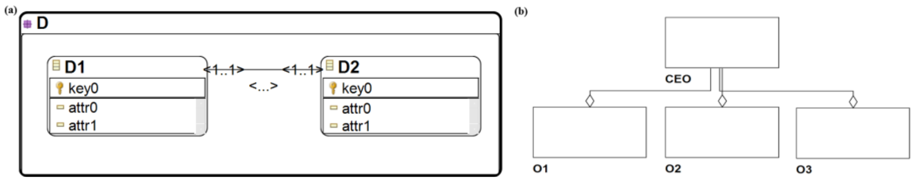


Fig. 2. Object model (a) and organization model (b) corresponding to the Petri net model from Table 1, both created with Horus Business Modeler

The object model (cf. Fig. 2a) visualizes the internal 1:1 relationship between objects D1_1 and D1_2 on the object/database level, whereas the organization model (cf. Fig. 2b) displays the organization structure and reveals the existence of further process roles.

The implementation of common process modeling tools requires that separated windows must be opened for a simultaneous view on all process model related information. To date there is no tool support for an integrated or parallel visualization of the different model types. Therefore, relationships of e.g., process activities and roles must be accessed by browsing different models and different windows.
In summary, the eEPC, BPMN and UML formalisms support the alignment of information from the object and role perspective with the control-flow model by inserting additional elements (as nodes) and linking them to the corresponding activities. Similarly, annotations can be used to tag the basic Petri net elements (places and transitions) with related information. Based on these findings, we aim to investigate both the efficiency of the present integrated visual implementation (single view) and the simultaneous display of different model types (multiple views) for the alignment of process activities with objects and roles.

3 Visualization Techniques

Visualizing the alignment between a process model, its assigned objects and organizational units at the same time with sufficient legibility and level of detail on the available screen space is often a big challenge and therefore effective visualization strategies are necessary. In this section we present three visualization techniques, which are a general representation of the visual implementations reported in section 2. Each of these three techniques – single view, multiple views, and multiple views in combination with linking and brushing – are discussed in this section. They build the foundation for the empirical validation of visualization preferences for process model alignment discussed in the subsequent section.

3.1 Single View Technique

The first visualization technique represents the alignment between a process model and its assigned information as node-link representation in a single view (see Fig. 3 for an example). Process activities, objects and roles are visualized as nodes. Different shapes for the nodes are used to differentiate between nodes that represent process activities (rectangle shape), roles (ellipse shape) or objects (trapezium shape).

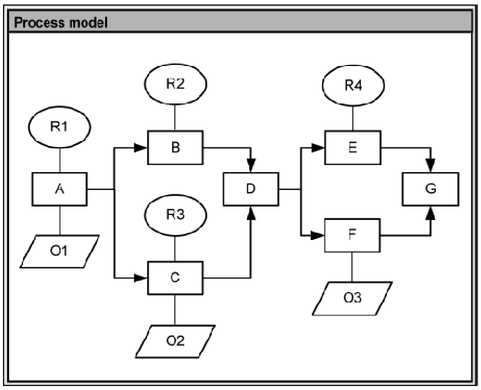


Fig. 3. Example for the single view visualization

The objects and roles are directly connected with the corresponding process activity in the process model with an arc. This kind of visualization allows users to get an overview about the structure of the process model and to see the connections between objects and roles for each process activity at the same time. If users want to view the structure of the organization model or of the object model, they can click on a role or object node in the process model to open the corresponding model in a separate window.

The single view technique provides an integrated overview about the structure of the process. However, users have to switch between three models to get a complete view. Especially for process models with plenty of objects and roles, the visualization of the process model at the same time in a single view/window is inefficient with regard to legibility. Clarity and consequently understandability of the single view decrease with the number of objects, roles and connection arcs used. Such a visualization technique is used in eEPCs, UML and also to some degree in BPMN.

3.2 Multiple Views Technique

Another technique is to use multiple views to present the alignment points between a process model and its organization model, and object model as node-link representation (see Fig. 4 for an example). Multiple views support different viewpoints that allow users to see process activities in combination with their objects and roles. The advantage of this technique is that the corresponding object and organization model are simultaneously displayed in the same window with the process model (they share one window). This allows for an immediate understanding of the connections between corresponding process elements. Scrolling and panning techniques are necessary for large process models, objects models and/or organizational models. In contrast to the single view approach (cf. example in Fig. 3), the connections between process activities to the corresponding objects and roles are not visualized as arcs respectively links in the process model view. This reduces the complexity and number of links that can have a negative effect on the general overview (understandability). The reduction of arcs also gives room to represent the object and organizational model together with the process model in one space. Especially for large process models it can happen that the links between the nodes are very long. Hence, following these links is very difficult and users can get lost in the node-link representation. The annotation of objects and roles directly below their process activity might be beneficial for large process models. Such a visualization technique is used for Petri net-based process modeling tools.

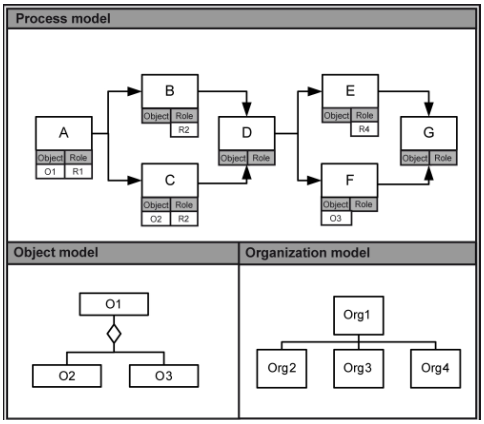


Fig. 4. Example for the multiple views visualization

3.3 Multiple Views Technique in Connection with Linking and Brushing

With the single and multiple views techniques, it is possible that the process model view is overloaded with too much information. A possible solution is to combine the visualization with interaction strategies, such as the linking and brushing technique [10], to simplify the representation of relationships between the process model, the object model, and the organization model in order to reduce overlaps between nodes and links. Linking and brushing technique is the most common interaction strategy for the representation of relationships between different views. If items are selected or highlighted in one view (called *brushing*), the corresponding connected items in the other views are also selected and highlighted (called *linking*). In our case, it supports users to trace the connections between the different models. For example, Fig. 5 shows four possibilities to present the connections between the process model view, the object model view, and the organization model view:

- No nodes are selected (cf. Fig. 5 (A)).
- One process activity is selected in the process model view. The corresponding objects in the object model view and the corresponding role in the organization model are highlighted (cf. Fig. 5 (B)).
- One object is selected in the object model view. The corresponding process activities in the process model view and the corresponding roles in the organization model are selected (cf. Fig. 5 (C)).
- One role is selected in the organization model view. The corresponding process activity in the process model view and the corresponding object in the organization model are highlighted (cf. Fig. 5 (D)).

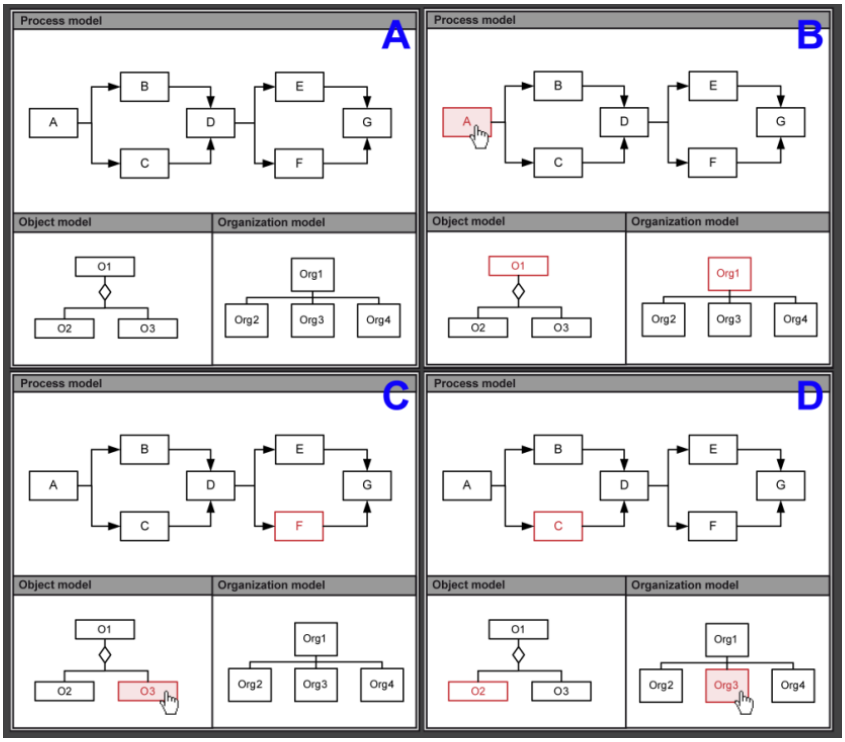


Fig. 5. Examples for multiple views approach in combination with linking and brushing technique: (A) no selection, (B) a process activity is selected, (C) an object is selected, and (D) a role is selected

3.4 Design Setting

To explore whether preferences for a particular visualization exist, we investigated the understandability of the alignment of process activities, objects and roles. This is validated with a web-based questionnaire that was set up. Participants were free to answer the questions and could withdraw the completion of the questionnaire at any time. The collection of data was anonymous. The questionnaire was designed in the following way.

Objects. The objects evaluated by each participant were five process models from the order management domain. One process model each was shown for the multiple views (see Fig. 7) and single view visualization (cf. Fig. 6) and three process models for the multiple views in combination with linking and brushing visualization (see, e.g., Fig. 8 and Fig. 9). The process models were not designed with a particular modeling language in order to avoid dogmatic discussions. To keep the process model itself simple, only sequence, split and join routing constructs were used. Additionally, we used a small process model (10 process activities). It has been identified that the number of process elements has implications on understandability if the preferred visualization is not used [7]. Therefore, we used a process model of moderate size in order to reduce cognitive load of the process model itself and direct the focus of the respondent to the alignment of process activities, objects and roles. While displaying a particular visualization technique, we asked some comprehension questions, which had to be answered with yes or no:

- The process activity *Process backorders* requires the objects *Order* and *Payment.*
- The object *Order* is used by six process activities.
- The process activity *Process order* is performed by the role *Buyer.*
- Two process activities require two different types of objects (data).

The last question was not asked for the visualization technique multiple views in combination with linking and brushing. This visualization technique is an interaction-based visualization. To answer the last question would require showing too many screenshots, which is beyond the scope of the questionnaire.

Factors and Factor Level. In our study, the alignment of process activities, objects and roles is the factor and the factor level is the visualization of the alignment.

Response Variable. The response variable in our study is the level of understanding that the respondents displayed with respect to the alignment. Understandability is measured as follows:

- the number of correct questions answered about the alignment visualization,
- the perceived ease of understanding (PEOU),
- the perceived usefulness (PU), and
- preferences between the three visualization techniques.

PEOU and PU are well-established measures that are widely used to investigate the understandability of models. They have been studied in [8]. PEOU, PU and visualization preferences had to be rated on a 5 point Likert scale (between 1=strongly disagree and 5=strongly agree).

Subjects. The survey was run from February to April 2013. To attract participants, we spread the link of the online questionnaire to modeling experts of different European universities and to research-driven institutes. In particular, we mainly personally asked post-graduate and assistant professors of institutes that work on business process management to complete the questionnaire[1]. The level of education of all participants was at least a completed master degree.

Instrumentation. We showed the participants a set of process activities, objects and roles visualized as multiple views, single view and multiple views in combination with linking and brushing. As the latter might not be intuitively understandable, we provided a short description of this technique (*"This visualization technique is an interaction technique (i.e., a click on an element shows its relationships to different units)."*).

Data Collection. Along with the questionnaire, we asked the participants about their gender, their professional situation, the years of modeling experience, their modeling

[1] After half time of the survey period we noticed an unbalance between responses concerning the principal modeling language. Therefore, we asked experts of European institutes that principally use a particular modeling language to also answer the questionnaire.

environment, number of created process models, number of analyzed process models, and their principal modeling language. Also, we received the answers for the comprehension questions, the PU and PEOU measures for each visualization technique and the preferences between the visualization techniques.

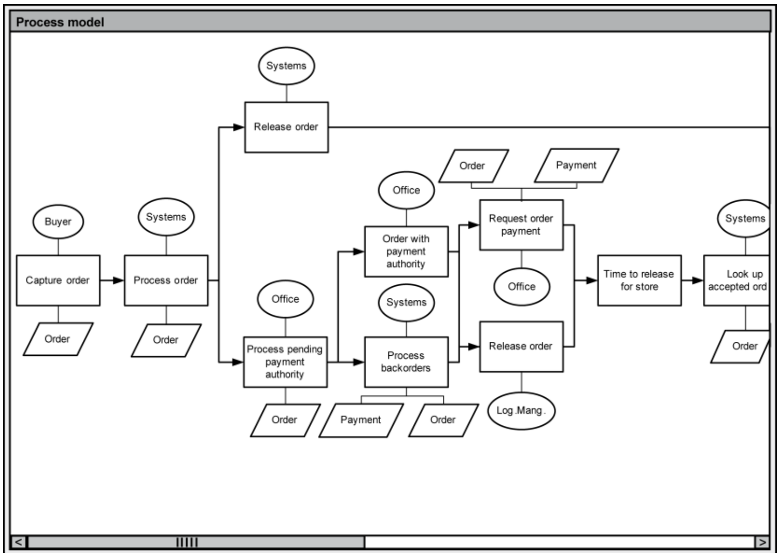


Fig. 6. Single View visualization from the evaluation (applied on the Order Management process)

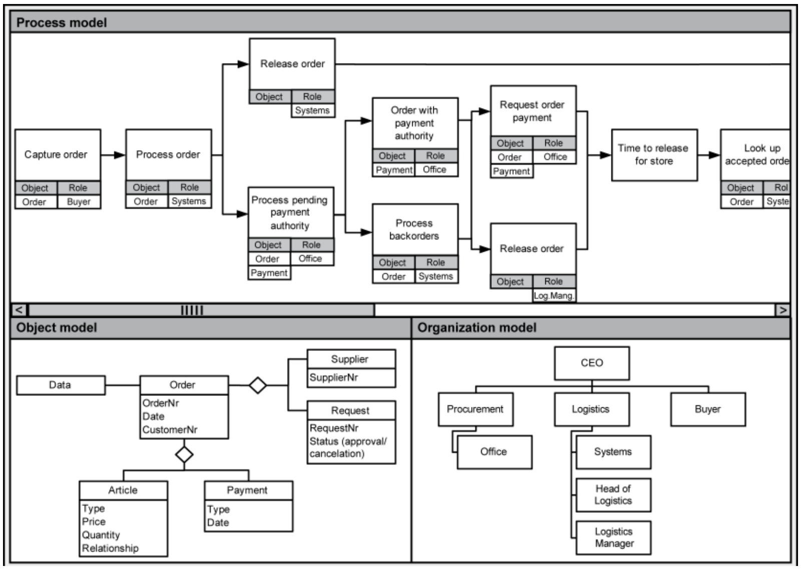


Fig. 7. Multiple views visualization from the evaluation (applied on the Order Management process)

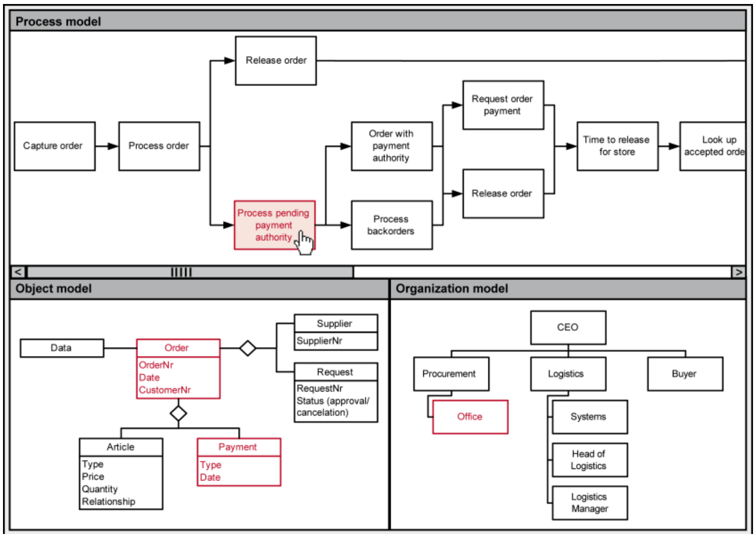


Fig. 8. Multiple views visualization in combination with linking and brushing from the evaluation (applied on the Order Management process)

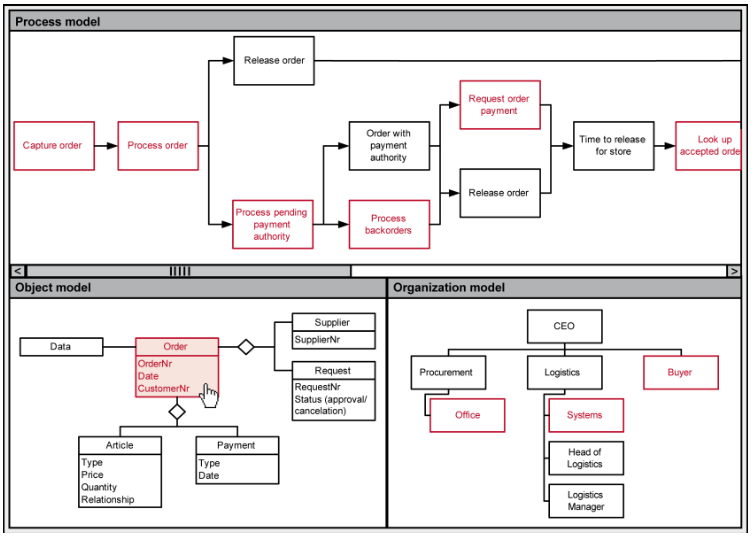


Fig. 9. Multiple views visualization in combination with linking and brushing from the evaluation (applied on the Order Management process)

3.5 Results

The questionnaire has been sent out to 81 persons. Overall, the questionnaire was answered by 52 persons, but only 33 completed questionnaires were obtained. The participants were 21.21% female (7 persons) and 78.78% male (26 persons).

Table 2. Statistical results for visualization preferences[2]

Preference	Options	Freq.	Freq. (%)	Cum.Freq(%)
Usefulness of *multiple views* over *single views*	s.agree	9	27.27	39.39
	agree	4	12.12	
	undecided	4	12.12	
	disagree	10	30.30	48.48
	s.disagree	6	18.18	
Improvement of performance between *multiple views* over *single views*	s.agree	6	18.18	39.39
	agree	7	21.21	
	undecided	7	21.21	
	disagree	7	21.21	39.39
	s.disagree	6	18.18	
Usefulness of *multiple views* over *multiple views in combination with linking and brushing*	agree	4	12.12	12.12
	undecided	6	18.18	
	disagree	8	24.24	72.72
	s.disagree	16	48.48	
Improvement of performance between *multiple views* over *multiple views in combination with linking and brushing*	agree	1	3.03	3.03
	undecided	9	27.27	
	disagree	8	24.24	69.69
	s.disagree	15	45.45	
Usefulness of *single views* over *multiple views*	s.agree	5	15.15	51.51
	agree	12	36.36	
	undecided	4	12.12	
	disagree	8	24.24	36.36
	s.disagree	4	12.12	
Improvement of performance between *single views* over *multiple views*	s.agree	6	18.18	42.42
	agree	8	24.24	
	undecided	6	18.18	
	disagree	9	27.27	39.39
	s.disagree	4	12.12	
Usefulness of *single views* over *multiple views in combination with linking and brushing*	s.agree	1	3.03	12.12
	agree	3	9.09	
	undecided	7	21.21	
	disagree	12	36.36	66.66
	s.disagree	10	30.30	
Improvement of performance between *single views* over *multiple views in combination with linking and brushing*	s.agree	1	3.03	12.12
	agree	3	9.09	
	undecided	7	21.21	
	disagree	12	36.36	66.66
	s.disagree	10	30.30	
Usefulness of *multiple views in combination with linking and brushing* over *multiple views*	s.agree	17	51.51	
	agree	9	27.27	78.78
	undecided	4	12.12	
	disagree	3	9.09	9.09
Improvement of performance between *multiple views in combination with linking and brushing* over *multiple views*	s.agree	15	45.45	78.78
	agree	11	33.33	
	undecided	4	12.12	
	disagree	3	9.09	9.09

[2] Please note that options, which were not selected are not considered in this table.

Most participants had only a modeling background in research, but 33% of the respondents additionally gained modeling experiences in industry projects. In average they had 4.81 years of modeling experience (St.Dev=3.8), modeled 59.5 and analyzed 59.68 business processes. The principal modeling language used by the participants was BPMN (13 persons, 39.39%), Petri nets (10 persons, 30.30%), EPC (5 persons, 15.15%), UML (3 persons, 9.09%), BPEL (1 person, 3.03%) and one person selected others (1 person, 3.03%). The results for visualization preferences were analyzed with respect to frequency distribution. Table 2 shows the statistical results for each preference, its answer options, the frequency in numbers per option, the frequency (%), and the cumulative frequency (%) for each question. Cumulative frequency is determined by aggregating agreement (strong agree, agree) and disagreement (disagree, strongly disagree) with the preference.

The multiple views in combination with linking and brushing is clearly preferred over multiple views and single view. The usefulness and the improvement of performance are significantly higher for multiple views in combination with linking and brushing visualization as for the other two visualization techniques. Comparing the frequency distribution between multiple views and single view, then a general indifference is observed. The single view visualization marginally wins with respect to usefulness. However, the agreement and disagreement for usefulness between both visualization techniques does not differ significantly.

We further investigated the PEOU and PU measures for each visualization technique in order to confirm the preference for the multiple views in combination with linking and brushing visualization. Table 3 summarizes the statistical results. The highest ease of use (PU) (agreement) is assigned to multiple views in combination with linking and brushing visualization (78.78) followed by single view (66.66) and multiple views (51.51). The same order is given for PEOU (agreement for ease of understanding and disagreement for frustration). Table 3 indicates an order between the three visualization techniques with respect to understandability, also multiple views in combination with linking and brushing visualization dominates marginally.

To detect any significant difference in the performance of participants, we checked the number of correct questions answered, which are showed in Table 4. Additionally, we determined the correlation between wrongly answered questions and each visualization approaches. The resulting values are shown in Table 4. The p-values exceed the threshold of 0.05 (using a confidence level of 95%) except for task 2 single view vs. multiple views in combination with linking and brushing (0.0066) and task 3 for multiple views vs. single view (0.0004). These results mean that mistakes for task 1 do not depend on a visualization technique studied in a previous question. Although a significance exists between mistakes for a particular task and visualization approaches, the correlation has no high validity. For instance, for task 3 the 0.0 mistake rate for the single view visualization is not given for the subsequent multiple views in combination with linking and brushing approach. This means that no differences can be observed (otherwise a significant p-value would be given for SingleView vs. MultipleViewsLB for task 3).

To investigate whether the understandability of a particular visual alignment depends on a principal modeling language, we run a correlation analysis. Table 5 shows the correlation coefficients calculated between the principal modeling language and PEOU, PU.

Table 3. Statistical results for PEOU and PU measures[2]

Statement	Options	Freq.	Freq. (%)	Cum.Freq (%)
Multiple Views				
It was easy for me to understand what the visualization was trying to model. (PEOU)	s.agree	11	33.33	66.66
	agree	11	33.33	
	undecided	6	18.18	
	disagree	3	9.09	15.15
	s.disagree	2	6.06	
Understanding the visualization was frustrating. (PEOU)	s.agree	1	3.03	12.12
	agree	3	9.09	
	undecided	5	15.15	
	disagree	12	36.36	72.72
	s.disagree	12	36.36	
Overall, the visualization was easy to use. (PU)	s.agree	6	18.18	51.51
	agree	11	33.33	
	undecided	10	30.30	
	disagree	3	9.09	18.18
	s.disagree	3	9.09	
Single View				
It was easy for me to understand what the visualization was trying to model. (PEOU)	s.agree	15	45.45	87.87
	agree	14	42.42	
	undecided	3	9.09	
	s.disagree	1	3.03	3.03
Understanding the visualization was frustrating. (PEOU)	s.agree	3	9.09	18.18
	agree	3	9.09	
	undecided	3	9.09	
	disagree	12	36.36	69.69
	s.disagree	11	33.33	
Overall, the visualization was easy to use. (PU)	s.agree	10	30.30	66.66
	agree	12	36.36	
	undecided	6	18.18	
	disagree	3	9.09	12.12
	s.disagree	1	3.03	
Multiple Views in Combination with Linking and Brushing				
It was easy for me to understand what the visualization was trying to model. (PEOU)	s.agree	15	45.45	81.81
	agree	12	36.36	
	undecided	3	9.09	
	disagree	2	6.06	9.09
	s.disagree	1	3.03	
Understanding the visualization was frustrating. (PEOU)	agree	2	6.06	6.06
	undecided	3	9.09	
	disagree	11	33.33	84.84
	s.disagree	17	51.51	
Overall, the visualization was easy to use. (PU)	s.agree	12	36.36	78.78
	agree	14	42.42	
	undecided	3	9.09	
	disagree	4	12.12	12.12

Table 4. Average Number of Mistakes and p-value for tasks

Avg.Mistake			
	Multiple Views	Single View	Multiple ViewsLB
task 1	0.2	0.000	0.029
task 2	0.114	0.571	0.000
task 3	0.086	0.000	0.143
task 4	0.314	0.227	

		p-value
task 1	MultipleViews vs. SingleView	0.1442
	SingleView vs. MultipleViewsLB	0.4049
	MultipleViews vs. MultipleViewsLB	0.3121
task 2	MultipleViews vs. SingleView	0.2905
	SingleView vs. MultipleViewsLB	0.0066
	MultipleViews vs. MultipleViewsLB	0.2609
task 3	MultipleViews *vs.* SingleView	0.0004
	SingleView *vs.* MultipleViewsLB	0.1694
	MultipleViews *vs.* MultipleViewsLB	0.1729

Table 5. Correlation coefficients for principal modeling language and PEOU and PU measures

	Corr.Coefficient(r)	*p-value*
Multiple Views	mod.lang,PEOU$_1$=0.284	0.0544
	mod.lang,PEOU$_2$=0.34	0.0264
	mod.lang,PU=0.347	0.0239
Single Views	mod.lang,PEOU$_1$=0.36	0.0198
	mod.lang,PEOU$_2$=0.34	0.0264
	mod.lang,PU=0.26	0.0719
Multiple Views in Combination with Linking and Brushing	mod.lang,PEOU$_1$=0.36	0.0198
	mod.lang,PEOU$_2$=0.417	0.0078
	mod.lang,PU= 0.32	0.0347

A significant indication for r is given for values from -0.6 or 0.6 respectively, which is not the case for any value in Table 5. However, significant p-values (<0.05) are given for several measures. It seems that understandability of visual alignments depends on the principal modeling language. Results in Table 5 reinforce the multiple views approach in combination with linking and brushing (since all corresponding p-values are <0.05). This visualization technique is easily understandable for all modeling language.

3.6 Discussion

Interpretation: The empirical study provides strong support for a process model visualization that is disburdened from context information such as objects used and roles involved. Instead an interactive alignment of process elements should be supported. The visualization of the process model should be reduced to the level of process activities and its control-flow when the model is shown for the first time.

This finding does not directly correspond to common practical implementations (see section 2) where plenty of graphical symbols and constructs are assigned to process activities. BPMN is an example for a "fussy layout"; i.e., the process model is overburdened with a variety of constructs.

Our statistical results also show that the visualization technique multiple views in combination with linking and brushing is easily understandable for users. Therefore, this visualization approach is a suitable alternative to current implementations.

Implications: The strong preference for a visual alignment in combination with linking and brushing is also in line with the postulation to decrease *symbol excess*, which increases diagrammatic and graphical complexity [10]. Therefore, the following implications are given for researchers and practitioners. Firstly, when proposing a new extension for a modeling language (e.g., consideration of security issues in BPMN) it is essential to empirically compare a visualization approach that add new symbols to process activities versus an interactive visualization technique (e.g., linking and brushing). Secondly, BPMS vendors should mainly implement tools that allow an interactive alignment between models.

Limitations: One might criticize the limited "complexity" of the used process models (e.g., the objects are abstracted from access controls, process model size). The intention of this study was to investigate the initial preferences for visualizing the alignment and therefore we decided to strip further disturbing factors from the process models. However, the visualization preference might change in case of large process models and in case that additional scrolling and panning techniques are necessary. This is one interesting issue, which provides room for further investigation.

Another open issue stems from the context of collaborative process models. In such a context, several organizations are involved in the modeling/management of the business process. Thus, inter-divisional questions must be addressed, which requires the consultation of several object and organization models. It remains open to investigate whether the multiple views in combination with linking and brushing visualization is still appropriate in such a context. Thirdly, it remains to investigate if process modelers in practice have same preferences as the academic respondents of this study. Lastly, the participants in our empirical study answered the questionnaire independent of a modeling purpose. However, [11] identified a correlation between the preferred visualization and the modeling purpose. For instance, the preference for a simplified BPMN (reduced to core constructs) depends on the modeling purpose. For persons with a managerial background, the core BPMN constructs are sufficient to finish the modeling task, while persons with an IT background argued for an extended set of BPMN constructs, which are more appropriate to model requirements in software implementation projects. Therefore, our findings might be suitable for users with no technical purpose.

4 Related Work

Various studies have investigated the factors which influence the understandability of business process models. Typically, two essential groups of factors are taken into account. Firstly, *model characteristics* such as control-flow complexity [2], the number of routing symbols [4] or activity labeling [12] give examples for metrics used to assess a process model itself. Secondly, *reader characteristics* describe e.g., the personal expertise of the person reading and interpreting a process model [13]. While previous work focuses solely on process model characteristics, this study addresses the integrated visualization of additional models: the object (data) model and organizational (role) model. We also assess reader characteristics to analyze possible correlations (see section 4).

Furthermore, several visualization concepts for information in and about business processes have been proposed. For example, the *proView* project aims at creating solutions for the flexible and personalized visualization of (large) business processes, e.g., with the help of parameterizable process views [5,6]. [13] developed an approach to cope with complexity of visual representations using a decomposition mechanism to split software system models "into manageable and understandable parts". In [1], a concept for the 3D representation of Petri nets is introduced, thus allowing the integration of additional information into a process model while still maintaining a compact and understandable visualization. Though, none of the mentioned examples investigates the simultaneous display of different models types (and provides an empirical evidence), which has been tackled in this paper.

5 Outlook

Cognitive load has an impact on understandability of process models. A heavy cognitive load is generated if novel material has to be understood [14]. A process model, which is viewed, e.g. for the first time ever, should be visualized in an easy and understandable way. Investigations of global and local understandability of models have been conducted in [3,4]. In this paper we present the first study that investigates the understandability of aligning process activity related information (in particular roles and objects). Three different types of visualization based on alignment techniques and practical implementations were proposed to participants.

The reported empirical study provides strong support for a visual alignment of models on demand, which is fully in line with the postulation to decrease *symbol excess* [10]. This means that users initially view the "pure" process model (with its activities and control-flow) and process element related alignments are displayed when users show interest for a particular context. This result requests researchers decreasing the number of symbols when suggesting a new extension for a process modeling languages. Instead alternative visualization techniques (e.g., in combination with linking and brushing) must be evaluated. Combining our results with findings of [11], we suggest fostering a purpose-oriented visualization starting with a process model that is reduced to the sole visualization of process activities and its control-flow and is continuously adjusted.

Aside from these implications, the results of our study provide extended insights into process model visualization and complement the body of experimental research on process model understandability.

References

1. Betz, S., Eichhorn, D., Hickl, S., Klink, S., Koschmider, A., Li, Y., Oberweis, A., Trunko, R.: 3D Representation of Business Process Models. In: MobIS 2008, pp. 73–87 (2008)
2. Cardoso, J.: Process control-flow complexity metric: An empirical validation. In: IEEE International Conference on Services Computing, pp. 167–173 (2006)
3. Figl, K., Laue, R.: Cognitive Complexity in Business Process Modeling. In: Mouratidis, H., Rolland, C. (eds.) CAiSE 2011. LNCS, vol. 6741, pp. 452–466. Springer, Heidelberg (2011)
4. Figl, K., Recker, J., Mendling, J.: A Study on the Effects of Routing Symbol Design on Process Model Comprehension. Decision Support Systems 54(2), 1104–1118 (2013)
5. Hipp, M., Michelberger, B., Mutschler, B., Reichert, M.: A Framework for the Intelligent Delivery and User-Adequate Visualization of Process Information. In: 28th Symposium on Applied Computing (SAC 2013), Coimbra, Portugal. ACM Press (March 2013)
6. Kolb, J., Reichert, M.: Data Flow Abstractions and Adaptations through Updatable Process Views. In: 28th Symposium on Applied Computing (SAC 2013), Coimbra, Portugal. ACM Press (March 2013)
7. Koschmider, A., Reijers, H.A., Dijman, R.: Empirical Support for the Usefulness of Personalized Process Model Views, Multikonferenz Wirtschaftsinformatik 2012, Braunschweig (March 2012)
8. Maes, A., Poels, G.: Evaluating quality of conceptual modelling scripts based on user perceptions. Data Knowl. Eng. 63(3), 701–724 (2007)
9. Moody, D.L.: The "Physics" of Notations: Toward a Scientific Basis for Constructing Visual Notations in Software Engineering. IEEE Transactions on Software Engineering 35(6), 756–779 (2009)
10. North, C.: Information Visualization. In: Salvendy, G. (ed.) Handbook of Human Factors and Ergonomics, 3rd edn., pp. 1222–1246. John Wiley & Sons (2005)
11. Recker, J., Indulska, M., Rosemann, M., Green, P.: How Good is BPMN Really? Insights from Theory and Practice. In: Proceedings 14th European Conference on Information Systems, Goeteborg, Sweden (2006)
12. Reijers, H.A., Mendling, J.: A Study Into the Factors That Influence the Understandability of Business Process Models. IEEE Transactions on Systems, Man and Cybernetics - Part A: Systems and Humans 41, 449–462 (2011)
13. Reinhard, T.: Complexity management in graphical models. Dissertation. University of Zurich (2011)
14. Sweller, J.: Evolution of human cognitive architecture. In: Ross, B. (ed.) The Psychology of Learning and Motivation, vol. 43, pp. 215–266. Academic Press (2003)

On-the-Fly Change Propagation for the Co-evolution of Business Processes

Shamila Mafazi, Georg Grossmann, Wolfgang Mayer, and Markus Stumptner

University of South Australia, Adelaide, SA, 5095, Australia
shamila.mafazi@mymail.unisa.edu.au,
firstname.lastname@unisa.edu.au

Abstract. In large organisations multiple stakeholders may modify the same business process. This paper addresses the problem when stakeholders perform changes on process views which become inconsistent with the business process and other views. Related work addressing this problem is based on execution trace analysis which is performed in a post-analysis phase and can be complex when dealing with large business process models. In this paper we propose a design-based approach that can efficiently check consistency criteria and propagate changes on-the-fly from a process view to its reference process and related process views. The technique is based on consistent specialisation of business processes and supports the data- and control flow perspective. This technique reduces the steps performed in the evolution of business processes by embedding the consistency checks and change propagation into the change enactment phase of the evolution.

1 Introduction

It is apparent that business processes evolve over time and have to be aligned with business rules, legislations, and law. Whereas, due to factors such as continuing change in regulations, policies, business compliances, and user demand the processes are subject to change. Since the economic success of any businesses relay on its ability to respond to the changes in its area swiftly, it is important for business to be able to update their processes in efficient and correct way [1].

Designing and updating process models in large organisations are usually performed by multiple stakeholders in parallel. Each stakeholder has a particular view on a process and performs changes independently from other stakeholders leading to the challenge that views may become out of sync and inconsistent with its reference process and other views. In this paper we propose a framework that offers generic methods and techniques for the efficient change propagation from a stakeholder to reference process and its process views on the process model level.

Gerth et al. [2] identified three steps for change management in process models: detection of differences, identification of change conflicts, and resolution of conflicts. We propose to propagate changes on the fly so stakeholders become immediately aware of changes made by other stakeholders which affect their view. By providing these capabilities, a separate step for conflict identification and resolution can be kept to a minimum or in optimal cases be eliminated. In this paper we focus on the mechanism on how to propagate changes on-the-fly and deal with changes made on the same process fragment at the same time in future work.

Related work often assumes that two corresponding process models are in strict refinement relation, whereas such a refinement relation is hardly noticed in practice [3]. We are applying existing techniques based on those strict process hierarchies but use this technique to propagate changes efficiently, i.e., we allow users to perform changes that violate the refinement relations and then propagate changes so consistency is established again. In particular, we are applying consistent specialisation of business processes to ensure consistency between views and reference process. By checking simple rules on the structure of process models and keeping relationships between elements in different processes, we are able to achieve co-evolution more efficiently.

The main contribution is twofold: In Section 3 we present our framework for on-the-fly change propagation in process co-evolution on the model level, and in Section 4 we provide a literature review and comparison criteria of state-of-the-art work in process (co-)evolution and change management.

The paper is structure as follows: the next section includes a motivating example which is used throughout the paper. Afterwards we explain our approach followed by a comparison of related work. The last section provides an outlook on future work and conclusion.

2 Motivating Example

Process models are widely used within organisations while they are mostly large and complicated. On the other hand dealing with a large and complicated process model is not always necessary, as preforming different business tasks need different parts of a process models to be highlighted. To fulfil this requirement, several methods to provide process views regarding different aspects of the process model have been proposed such as [4–6]. In previous work [7], we proposed a knowledge based framework to provide different views on a process model considering the user specified constraints. We have applied our approach on a real-world business process repository from the *product and system engineering* domain which contained software development processes with up to 260 activities in a single process. In some use cases, a view could be generated that reduced the complexity by up to 90%.

Different users may change their views over a reference model. To preserve the consistency between them change in one view needs to be propagated to the remaining views through the reference model. The co-evolution of process models refers to the change propagation between models either at the same or different abstraction hierarchy [8]. These changes may cause inconsistency in the process model views or the reference model. In this work we deal with inconsistent models in a model co-evolution scenario.

Fig. 1 indicates our motivating scenario, where every user has their own view over the reference model. As indicated in the figure users make changes in their process views. To preserve the consistency between these process models, change in one view needs to be propagated to the remaining views through the reference process model.

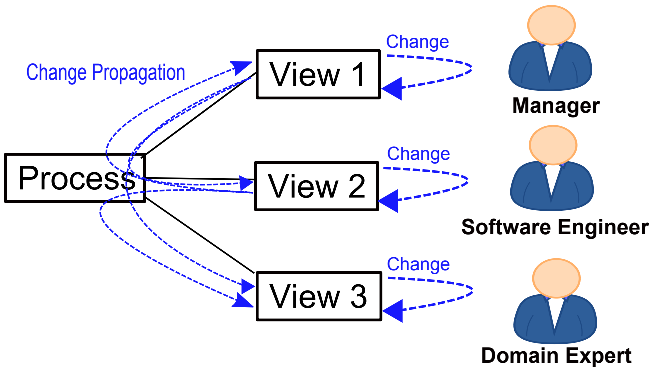


Fig. 1. Motivating Scenario

Fig. 2 shows an example for co-evolution of a reference process model and its views in BPMN. The model in the lower left depicts the reference model, the models in the top show two different views over the reference model. The grey and pink dashed lines indicate the correspondence relations for the tasks in process view 1 and view 2 respectively. In this figure the correspondence relations for gateways, control flows, and identical activities are left implicit. As illustrated in the figure, the mappings between views and the reference process model are different. For example tasks *Cancel Late* and *Send Cancellation Confirmation* from the reference model have been mapped to the task *Cancel* in process view 1 whereas, in process view 2, task *Send Cancellation Confirmation* has been removed by mapping it to a control flow in the view.

In the example, a new activity *Inform* is introduced by a stakeholder in View 2 which is highlighted by a bubble within View 2 in Fig. 2. This change is then propagated to the reference process by inserting *Inform* between *Cancel Late* and *Send Cancellation Confirmation*. In a last step, the changes are propagated to View 1 which consists only of an update of the properties in *Cancel* because the region including *Cancel Late* and *Send Cancellation Confirmation* is abstracted in View 1.

Related work mostly deals with schema evolution and migration of process instances, preserving the data flow consistency. To the best of our knowledge on-the-fly propagation of changes among different views at different abstraction levels has not been investigated yet. The following section presents our framework.

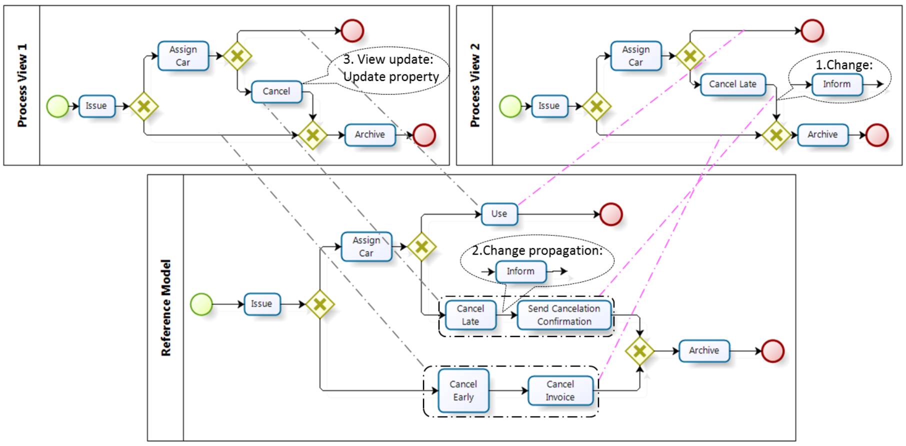


Fig. 2. Change Propagation Example

3 Consistent Co-evolution of Business Process Modelling

A typical software development life-cycle consists of several phases. Köhler et al. [9] introduced a cycle for the alignment of business and IT which consists of model-, develop-, deploy-, monitor- and analyse & adapt phases. We propose a framework for the consistent co-evolution of business processes in the modelling phase which consists of three steps: (1) in the *process modelling* step, a business process is created that represents the reference process, (2) in the *process view generation* step, views are generated from the reference process for stakeholders, and (3) in the *change and change propagation* step, different stakeholders perform changes on their views and their changes are propagated on the fly to reference processes and other views. The first step needs to be performed only once whereas steps 2 and 3 are usually repeated for each development cycle. This framework makes use of the *model driven engineering, MDE* paradigm. The advantages of MDE include increasing the productivity and compatibility by simplifying the process of design, and maximising reusability by lifting platform specific code to a platform independent model level. Our approach is set on the level of models which are the central artifact in MDE. The problem of consistent software evolution is also addressed by other communities such as semantic web comunity. However, these communities rather target a lower level of implementation and specific language whereas our approach is on the platform independent level.

We explain now each step in more detail and provide definitions for the formal notation and consistency constraints.

3.1 Process Modelling

There exist different languages for modelling business processes. BPMN has become the de-facto standard in recent years and we use this notation to represent the models. However, a more formal representation is required for specifying the relationship between reference process and process views, and for the verification of consistency.

Therefore we use a Petri net representation of the BPMN models, in particular labelled Petri nets which are helpful to express the consistent specialisation of processes [10]. Different approaches for mapping between the BPMN and Petri net notation exist. Since the discussion of such mappings are out of the scope of this paper we refer the interested users to [11].

A labelled Petri net is a Petri net that has labels attached to its arcs. As discussed in [10] the idea of labelling arcs corresponds to the mechanism by which different copies of a form are handled by each activity in business processes guided by paperwork, where different copies of a form have different colours. Each activity deals with different copies of a form where it can collect the copies from different input boxes and deliver them to different output boxes. The labels determine which copies have been used by which activities.

Definition 1 (Labelled Petri Net)
A tuple: $(N, F, D, D_{in}, D_{out}, \delta_{in}, \delta_{out}, L, l)$ *is a labelled Petri Net model, where* N *is a finite set of nodes partitioned into disjoint sets of activities* N_t *and states* N_s*,* $F \subseteq (N_s \times N_t) \cup (N_t \times N_s)$ *is the flow relation such that* (N, F) *is a connected graph,* D *is a finite set of data variables,* $D_{in} \subseteq D$ *and* $D_{out} \subseteq D$ *are the sets of input and output variables of the process, and* $\delta_{in} : N_t \mapsto 2^D$ *and* $\delta_{out} : N_t \mapsto 2^D$ *are functions mapping each activity node to its set of input and output variables, respectively. We write* $n_t.D_{in}$ *and* $n_t.D_{out}$ *for* $\delta_{in}(n_t)$ *and* $\delta_{out}(n_t)$*, respectively.* L *is a finite set of labels, and* $l : F \mapsto 2^L \setminus \emptyset$ *is a function that assigns a set of labels to each control flow. Moreover, we write* F^* *to denote the transitive closure of the control flow relation* F*.*

Different operations, such as initializing, updating, and verifying, performed by each activity in a process model, can be divided to read and write operations from an implementation point of view. In our framework, when an activity x reads or writes a data object, that data object is considered to be the input or the output of that activity denoted as $x.D_{in}$ and $x.D_{out}$ respectively.

The execution semantics of a labelled Petri Net is the same as a Petri Net where an activity produces a token to its immediate post states and consumes a token from its immediate pre states. For the reference process model and all its views, we assume a subset of labelled Petri Net which satisfy the certain properties:

– Safe: a labelled Petri Net is safe iff there is no execution trace in the model such that some activities on that trace can be completed while there exist some post states of those activities on that trace. This is a necessary property as unsafe Petri Net contradicts the completeness and intention of a behaviour diagram.

- Activity reduced: this property is satisfied iff for every activity there is an execution trace that contains it. This property is necessary as the process model must not contain an activity which does not have any impacts on the process model.
- Deadlock: this property ensures that the execution must continue unless it reaches a final state. This is an important property as it prevents from blocking execution.
- Label preservation: requires that for every activity the incoming and outgoing arcs contain the same set of labels. In the real world scenario, this property ensures that once a specific form for a process instance has been developed, that form is neither overwritten nor are new copies of the form produced by other activities in the process model.
- Unique label: this property ensures that the label set of the incoming arc(s) of every joint or split activities be different from its outgoing label set. This property ensures that every activity can have some copies of a form only one input box and must deliver them to only one output box.
- Common label distribution: To preserve this property all the immediate arc(s) of a state must contain the same set of labels. This property ensures that each box contains the same copies for a form.

Figure 3 shows the Petri net representation of the BPMN process model shown in Fig. 2. Data input and -output are indicated above each activity as label in form of $d*$ where $*$ is an ID indicating which activity consumes data from another activity. For example, the activity *Assign Car* in Process View 1 of Figure 3 has inputs $d1$–$d3$ which are produced by activity *Issue*.

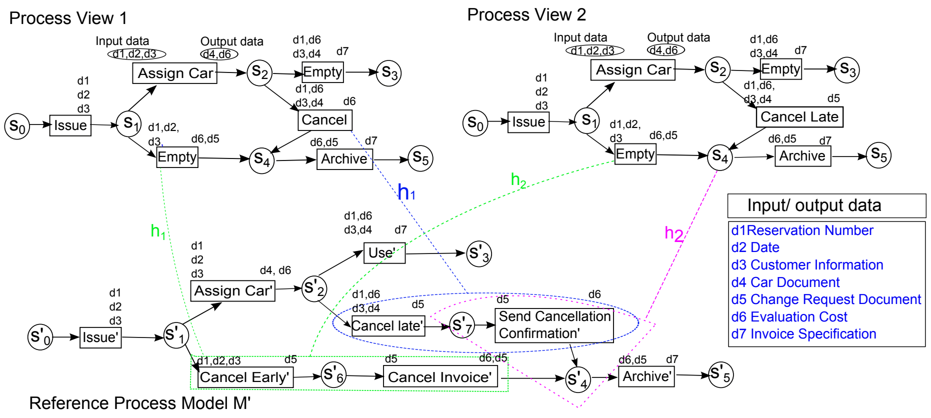


Fig. 3. Petri Net model for process models m in Fig. 2, for clarity's sake the control flow labels are omitted

To capture the relation of two activities regarding the execution sequence we base our formalization on the concept of *dominance* by Cytron et al. [12]. The binary *dominance* defines a relation between process model elements, considering all possible execution traces:

Definition 2 (Dominance Relation). *Let P be a labelled Petri Net model with nodes N and flow relation F. A non-empty set $D = \{n_1, ..., n_k\} \subseteq N$* dominates *a node $n \in N$, denoted as $D \lhd n$, if and only if in all execution traces from the start node to node n, a node in D precedes every execution of n.*

Example 1. Node *Issue* in Fig. 3 View 1 dominates all subsequent nodes; and node set {*Empty*, *AssignCar*} dominates node *Archive*.

Definition 3 (Immediate Pre-Node and Post-Node)

The set of pre-nodes of node $n \in N$ is defined as $\bullet n = \{n_1 \in N | (n_1, n) \in F\}$. Likewise the set of the post-nodes of node $n \in N$ is defined as $n\bullet = \{n_1 \in N | (n, n_1) \in F\}$.

Example 2. $\bullet$CancelLate=$\{s_2\}$, and CancelInvoice$'\bullet$=$\{s'_4\}$ in Figure 3.

After a reference process is modelled in BPMN and its equivalent representation as a labelled Petri net is generated, process views can be created from it.

3.2 Creating Process Views

In our previous work, we describe a knowledge-based framework for the purposeful abstraction of process models from constraints [7]. Constraints can be specified on activity properties and process views are generated automatically which fulfil the constraints. A constraint is specified according to the interests of a particular stakeholder, for example, a performance manager may only be interested in observing activities which take longer than a specified threshold. The framework is based on configuration techniques and can provide multiple views on different abstraction levels. The correspondences between a process model and its views are captured by a function while constructing the views. This allows checking for consistency and change propagation later in the modelling phase.

In our framework the main functions for abstracting a process model are *AggregateActivities()* and *RemoveActivity()*. By aggregating a group of activities from the reference model and mapping them to a composite activity in the view, a n:1 relationship is established.

Example 3. In Figure 3, the activities *Cancel late* and *Send Cancellation Confirmation* of Reference Process M' are aggregated to *Cancel* in Process View 1.

For removing irrelevant activities from a view, the *RemoveActivity()* function is applied. There are two possible outcomes: nodes are either removed or replaced by a *silent activity*. In the first case, a relationship between the deleted nodes in the reference model and an arc representing a control flow in the view is established. The second case occurs if removing a node would violate the consistency between the reference process and its view. In this case, a relationship between the deleted nodes and a *silent activity* is established.

Definition 4 (Silent activity)
An empty activity $N_e \subseteq N_t$ is a silent activity [11] which has no effect in the process model but it is required to preserve the syntax of Petri net notation as well as the model consistency.

Example 4. In Figure 3, activity *Use* is mapped to a silent activity labelled "Empty" in both views. This is necessary because removing it would make states s_2 in both views a final state which is inconsistent with the reference process.

We capture the relationship between elements in the reference process and its views by a total mapping function $h()$ such that:

1. It maps each state, activity, data and control flow of the reference process model to the elements of the view.
2. Labels of the reference process are mapped to labels in the view. Different labels in the reference process model can be mapped to a single label in the view.
3. The initial state of the original model is mapped to the initial state of the changed model.

Definition 5 (Process View). *A labelled Petri Net diagram (for brevity we excluded the functions from the definition of the labelled Petri Net) $M = (N, F, D, D_{in}, D_{out}, L)$ is defined as a view of another labelled Petri Net diagram $M' = (N', F', D', D'_{in}, D'_{out}, L)$ by the function $h_{M' \mapsto M}$.*
$h : N' \cup F' \cup D'_{in} \cup D'_{out} \cup L' \mapsto N \cup F \cup D_{in} \cup D_{out} \cup L$

1. $e \in N' \cup F' \Rightarrow h(e) \in N \cup F$,
2. $e \in D'_{in} \cup D'_{out} \Rightarrow h(e) \in D_{in} \cup D_{out}$,
3. $l' \in L' \Rightarrow h(l') \in L$,
4. $a' \in N'_s, a \in N_s, \bullet a = \emptyset, \bullet a' = \emptyset \Rightarrow h(a') = a$

Example 5. Figure 3 shows two process views that have been generated from the reference process M' where, for example, the sequence $\{s'_7$, *Send Cancellation Confirmation*$'$, $s'_4\}$ in the reference model is mapped to state s_4 in View 2. The relationship between the reference process and the view for this mapping is specified as $h(s'_7) = h(\textit{Send Cancellation Confirmation}') = h(s'_4) = s_4$.

The *process view generation* step ensures that a generated view is consistent with the reference process. In the following section we describe the consistency criteria between references process and views, and how they can be checked using simply design rules.

3.3 Consistency Criteria

The problem of inconsistency between multiple views of a reference model is well-studied [13]. Consistency criteria help to identify contradictions between views and their reference process model once different stakeholders change the views. In our framework we apply *consistent specialization of business processes* to ensure the consistency between process views and their respective reference process model.

Specialisation consists of *refinement* and *extension* where refinement refers to refining an activity into more detail and extension refers to adding new elements to a process model. A process model P' is an extension of process model P if it possesses new features in comparison to P. Likewise, process model P' is a refinement of a process model P if the inherited features are considered in more detail.

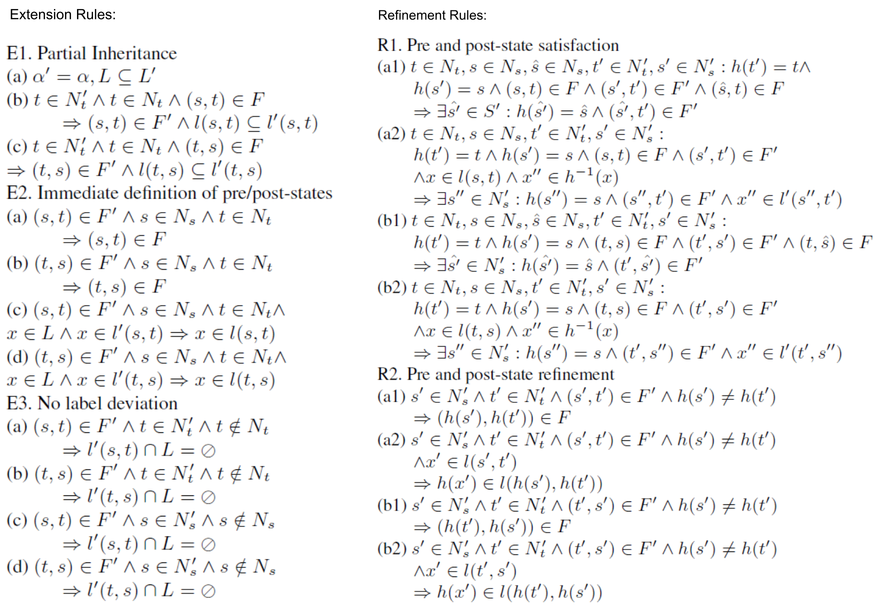
Schrefl et al. [10] investigated three criteria for the consistent refinement and extension of business processes: *observation consistency*, *weak-* and *strong invocation consistency.* Informally, observation consistent specialization guarantees that if features added at a process are ignored and features refined at a changed process model are considered unrefined, any instance of that changed process can be observed as correct from the point of view of the original process.

Weak invocation consistency captures the idea that instances of a changed process can be used the same way as instances of the original process. An extended version of this property, strong invocation consistency, guarantees that one can continue to use instances of a changed process the same way as instances of a original model, even after activities that have been added in the changed process model have been executed.

These criteria relate to other methods that investigated consistency between business processes. Protocol- and projection inheritance introduced by Basten et al. [14] corresponds to weak invocation consistency and observation consistency respectively. The advantage of the work by Schrefl et al. is a set of rules that allow to efficiently check for consistency without the necessity to analyse all possible traces.

Fig. 4 shows the consistency rules by which the consistency of a process model is assessed. According to Schrefl et al. [10] preserving the extension rules E_1 to E_3, on the left side of Fig. 4 guarantees the observation consistency between the two models, whereas refinement rules R_1 and R_2 from Fig. 4 assure observation and invocation consistency. A process model A' is the observation consistent with another model A if, by ignoring the new features of A', the instances of A' can be observed seamlessly by A. Two process models are invocation consistent if all the instances of process model A' can be invoked by the process model A.

The rules shown in Fig. 4 are used to check for consistency of control flows between reference process and views. As mentioned above, there are two types of consistency rules, rules for extension and rules for refinement, depicted on the left and right hand side in Fig. 4. Which consistency rule needs to be checked depends on the applied change. For example if an activity is inserted into or



Extension Rules:

E1. Partial Inheritance

(a) $\alpha' = \alpha, L \subseteq L'$

(b) $t \in N'_t \wedge t \in N_t \wedge (s,t) \in F$
$\Rightarrow (s,t) \in F' \wedge l(s,t) \subseteq l'(s,t)$

(c) $t \in N'_t \wedge t \in N_t \wedge (t,s) \in F$
$\Rightarrow (t,s) \in F' \wedge l(t,s) \subseteq l'(t,s)$

E2. Immediate definition of pre/post-states

(a) $(s,t) \in F' \wedge s \in N_s \wedge t \in N_t$
$\Rightarrow (s,t) \in F$

(b) $(t,s) \in F' \wedge s \in N_s \wedge t \in N_t$
$\Rightarrow (t,s) \in F$

(c) $(s,t) \in F' \wedge s \in N_s \wedge t \in N_t \wedge$
$x \in L \wedge x \in l'(s,t) \Rightarrow x \in l(s,t)$

(d) $(t,s) \in F' \wedge s \in N_s \wedge t \in N_t \wedge$
$x \in L \wedge x \in l'(t,s) \Rightarrow x \in l(t,s)$

E3. No label deviation

(a) $(s,t) \in F' \wedge t \in N'_t \wedge t \notin N_t$
$\Rightarrow l'(s,t) \cap L = \emptyset$

(b) $(t,s) \in F' \wedge t \in N'_t \wedge t \notin N_t$
$\Rightarrow l'(t,s) \cap L = \emptyset$

(c) $(s,t) \in F' \wedge s \in N'_s \wedge s \notin N_s$
$\Rightarrow l'(s,t) \cap L = \emptyset$

(d) $(t,s) \in F' \wedge s \in N'_s \wedge s \notin N_s$
$\Rightarrow l'(t,s) \cap L = \emptyset$

Refinement Rules:

R1. Pre and post-state satisfaction

(a1) $t \in N_t, s \in N_s, \hat{s} \in N_s, t' \in N'_t, s' \in N'_s : h(t') = t \wedge$
$h(s') = s \wedge (s,t) \in F \wedge (s',t') \in F' \wedge (\hat{s},t) \in F$
$\Rightarrow \exists \hat{s'} \in S' : h(\hat{s'}) = \hat{s} \wedge (\hat{s'},t') \in F'$

(a2) $t \in N_t, s \in N_s, t' \in N'_t, s' \in N'_s :$
$h(t') = t \wedge h(s') = s \wedge (s,t) \in F \wedge (s',t') \in F'$
$\wedge x \in l(s,t) \wedge x'' \in h^{-1}(x)$
$\Rightarrow \exists s'' \in N'_s : h(s'') = s \wedge (s'',t') \in F' \wedge x'' \in l'(s'',t')$

(b1) $t \in N_t, s \in N_s, \hat{s} \in N_s, t' \in N'_t, s' \in N'_s :$
$h(t') = t \wedge h(s') = s \wedge (t,s) \in F \wedge (t',s') \in F' \wedge (t,\hat{s}) \in F$
$\Rightarrow \exists \hat{s'} \in N'_s : h(\hat{s'}) = \hat{s} \wedge (t',\hat{s'}) \in F'$

(b2) $t \in N_t, s \in N_s, t' \in N'_t, s' \in N'_s :$
$h(t') = t \wedge h(s') = s \wedge (t,s) \in F \wedge (t',s') \in F'$
$\wedge x \in l(t,s) \wedge x'' \in h^{-1}(x)$
$\Rightarrow \exists s'' \in N'_s : h(s'') = s \wedge (t',s'') \in F' \wedge x'' \in l'(t',s'')$

R2. Pre and post-state refinement

(a1) $s' \in N'_s \wedge t' \in N'_t \wedge (s',t') \in F' \wedge h(s') \neq h(t')$
$\Rightarrow (h(s'),h(t')) \in F$

(a2) $s' \in N'_s \wedge t' \in N'_t \wedge (s',t') \in F' \wedge h(s') \neq h(t')$
$\wedge x' \in l(s',t')$
$\Rightarrow h(x') \in l(h(s'),h(t'))$

(b1) $s' \in N'_s \wedge t' \in N'_t \wedge (t',s') \in F' \wedge h(s') \neq h(t')$
$\Rightarrow (h(t'),h(s')) \in F$

(b2) $s' \in N'_s \wedge t' \in N'_t \wedge (t',s') \in F' \wedge h(s') \neq h(t')$
$\wedge x' \in l(t',s')$
$\Rightarrow h(x') \in l(h(t'),h(s'))$

Fig. 4. Rules for checking behaviour consistency of extension (on the left) and refinement (on the right) [10]

deleted from a view, the extension rules are relevant. In the following we discuss data flow consistency which is checked locally in a view once changes are applied.

Data Flow Consistency Criteria: We discuss three data-flows anomalies investigated by Reichert et al. [1] which are *missing data*, *unnecessary data*, and *lost data*.

In the following firstly the rules for identifying the violation of data flow consistency after propagating all the required changes in a process model are presented. Secondly, the repair option(s) for each violation are presented.

Missing data refers to the situation where a data object o is read by an activity x before it has been written. In our framework, the rule to ensure the absence of missing data in a model, can be expressed formally as:
$\forall x \in N_t : o \in x.D_{in} \Rightarrow \exists Y \subset N_t : Y \lhd x \wedge o \in \bigcap_{y \in Y} y.D_{out}$
This states that for each reader x of o, there must be a dominating set Y where each member writes o.

$\forall x \in N_t : o \in x.D_{in} \rightarrow \exists y \in N_t : (y,x) \in F^* \wedge o \in y.D_{out}$
If a process model contains missing data, i.e. the above rule gets violated, to repair the inconsistency, the process modeller has two options:

1. The missing data object is added to a preceding activity, i.e. $AddOutputDataFlow(y,o)$
2. The activity causing the missing data inconsistency situation is removed from the process model, i.e. $RemoveActivity(x)$

An *unnecessary data* situation occurs where a data object o is written by an activity x, but is not read by any activity (such as y) afterwards in the process model. Formally the rule is expressed as:
$\forall x \in N_t : o \in x.D_{out} \Rightarrow \exists y \in N_t : (x, y) \in F^* \wedge o \in y.D_{in}$

In case of violation of the rule, the data object o which causes the inconsistency needs to be removed from the output data of the activity x, that is $RemoveOutputDataFlow(x, o)$. The other option is to map the data object o to the input data set of a successor activity. However considering this option, requires change in the internal functionality of that successor activity which cannot be considered at the model level.

Lost data happens when a data object o is written twice by two consecutive activities x and y without being read in between. Formally:

$$\forall x, y \in N_t : o \in x.D_{out}, o \in y.D_{in}, (x, y) \in F^* \Rightarrow \nexists Z \subset N_t \setminus \{x\} : \\ Z \lhd y \bigwedge_{z \in Z} [o \in z.D_{out} \wedge (x, z) \in F^*]$$

A data object o written by x is lost data if any potential reader y of o reachable from x is dominated by a set of writers Z whose members are reachable from x. Hence, o written by x is not lost if no such set Z exists. To repair the lost data flow inconsistency, in case of violation of the mentioned rule, Sadiq et al. [15] provided two options for the process modeller in order to resolve the inconsistency as follows:

1. The modeller selects one activity to write data object o and that data object is removed from the output data set of the other consecutive activity(ies) writing that object.
2. The other option is to change the data object o in one of the consecutive activities.

The consistency of control- and data flow is checked after changes are made to a process view. In the following we discuss applying changes to views in general and then how the consistency criteria are used in the change propagation step.

3.4 Performing Changes

Our library of change operators currently contains two types of operators: structural and entity change operators. For change propagation we address two types of change operations, adding and deleting activities. As indicated by related work, a majority of change patterns can be composed out of those operators [16]. An example for entity change operators are also *add* and *remove* which are applicable on the data inputs and outputs of activities. Table 5 indicates a list of operators currently defined in our library.

3.5 Change Propagation

If one of the views changes, the changes need to be propagated to restore consistency between the views and the reference model. However in some cases there

Operator	Type
Remove Node/Flow	Structural
Add Node/Flow	Structural
Remove Data flow	Entity
Add Data flow	Entity

Examples for Logic Representation of Operators
RemoveActivity(x)**:**
$\forall x \in N_t, e \in N_e \rightarrow$
$h(x) = e \wedge e.D_{in} \leftarrow \oslash \wedge e.D_{out} \leftarrow \oslash$
AddOutputDataFlow(x, i)**:**
$x.D_{out} \leftarrow i$
RemoveOutputDataFlow(x, o)**:**
$x.D_{out} \leftarrow o$

Fig. 5. Change Operators

is an issue, regarding preserving the consistency and privacy of a process view. On the one hand one can imagine that some changes applied by a stakeholder are private or sensitive which cannot be shared with other views or added to a reference model. On the other hand to preserve the consistency, every change in a process view needs to be propagated to the other views as well as the reference model regardless of the sensitivity or privacy of those changes. One possible compromise is that the private changes in a view can be propagated anonymously i.e. as an empty task, such that the changes are not visible to other stakeholder yet the changes have been propagated hence the consistency is preserved. Fig. 6 illustrates a change propagation scenario where activity *Inform* needs to be inserted in Process View 2 after activity *CancelLate*. The scenario includes the following steps:

Apply Changes: A stakeholder applies changes on a view using available operators.

Example 6. As illustrated in Fig. 3, activity *Inform* is inserted in View 2.

Check Process Properties: A first correctness check includes checking for local properties: safe, activity-reduced, deadlock-free, and label properties as explained in Section 3.1 and data inconsistencies. The process properties can be checked efficiently using techniques such as SESE fragmentation [17]. If properties or data inconsistencies are violated then the process model must be changed in a way so they are satisfied and no data flow violation occurs.

Example 7. After activity *Inform* is inserted, the process view remains safe, activity-reduced and deadlock-free, and no data flow inconsistencies were introduced.

Check Consistency Rules: In a following step the consistency rules shown in Fig. 4 are checked for observation, weak- and strong invocation consistency. If the criteria are violated then the stakeholder is informed that the applied changes will have an impact on reference process and other views. This provides an option to undo certain changes in case the impacts of the changes were unintentional.

Example 8. In the example Views 2 in Fig. 3 and 6 are not consistent after activity *Inform* is inserted because it violates the observation consistency. As the post-states of *Cancel Late* in the two views are different, the post-state satisfaction, rule b_1 in Fig. 4 is violated. Hence View 2 in Figure 6 is not a behaviour consistent specialisation of View 2 in Fig. 3. The stakeholder receives a warning about this and proceeds because the change was intentional and has to be propagated.

Propagate Changes to Reference Model: The same change operators that have been applied on the view are applied on the reference process with the difference that the arguments of the change operators are mapped according to the relationship between the elements of the view and the reference process. For this the *h()* function is used which relates each element from one process to the other. Further the relationship between the reference process and the view needs to be updated in the implementation of the *h()* function.

Example 9. As illustrated in reference model from Fig. 6 activity *Inform* is added to the reference process. The relationship between the reference process and View 2 is updated with $h(s_6') = \{s_6\}$ and $h(\mathit{Inform}') = \{Inform\}$.

Propagate Changes to Views: First, for each view, its consistency with the reference model is checked. If the process and the view are consistent and the changes in the reference process affect a region that is abstracted in the view then no changes need to be propagated. Otherwise the same change operator as on the reference process is applied on the view where the arguments of the operator are translated by using the mapping function *h()* that maps elements in the reference process to the view.

Example 10. In Fig. 6 the reference model is not a behaviour consistent refinement of View 1. Checking the changed region in the reference model from Fig. 6 and $h()$ function, it is obvious that no change needs to be applied in the structure of the process View 1. To keep View 1 consistent with the reference model, only the activity *Inform′* and its pre-state s_6' from the reference model need to be mapped to the activity *Cancel* from process View 1, that is: $h(s_6') = h(Inform') = \{Cancel\}$ between reference process and View 1.

4 Related Work

The following subsections discuss the related work directly addressing change propagation between process models as well as the approaches which are potentially useful in the co-evolution of process models such as finding corresponding process model elements. All the mentioned approaches are compliment to one aspect presented in this paper. The following sub sections discuss the three main aspects in the change management as, *process model alignment*, *co-evolution* and *variability*. The results of comparision of related work have been captured in tables 1 and 2 where the first table compares the related work in the first area and

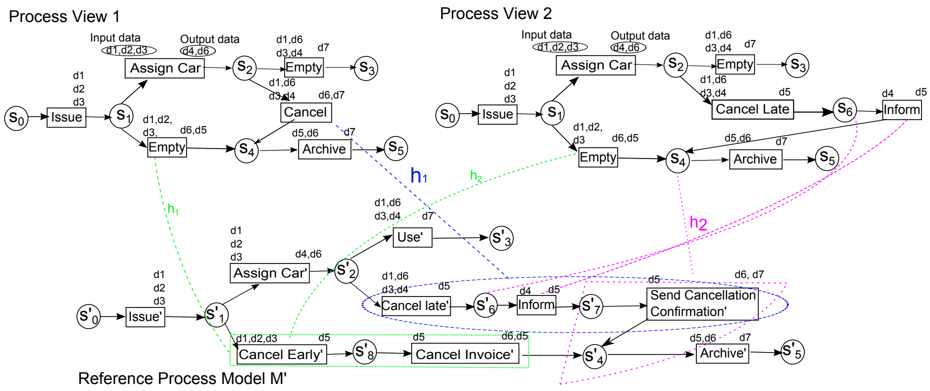


Fig. 6. Change propagation steps following changes made to Fig. 3, for clarity sake the control flow labels are omitted

the second table the two former areas. In these tables rows specify the papers and indicate their support towards the following criteria:

Goal: This column classifies the related work based on their goal.

Data Flow (Consistency): The data flow column indicates whether the related work considers data flow consistency in a change propagation scenario. This criteria is important as the robustness of a process model is fundamentally depended on the correctness of the data flow [1]. The possible values include supported, not supported, the criterion is not in the scope of the paper, and the criterion is in the scope of the paper but it has not been discussed.

Bidirectional Change Propagation: This column indicates the support of the approach for a top down and bottom up change propagation where different levels of abstraction hierarchy has been considered by the approach. This is an important aspect since the change cannot be restricted to any particular abstraction levels.

Support for Process Views: This column indicates the support of the approach for different process views. This column holds the same possible values as the previous column.

Consistency Preservation: The column discusses the consistency criteria proposed by the related work in the change propagation scenario. This is an important criterion as in practice there is no single process model but a group of process models. In order to preserve the uniformity between such models, the consistency between them must be evaluated against formal criteria.

Change Operators: This column indicates whether the change propagation technique is based on change operators or not. Using an operator based approach has different advantages such as identifying and enabling the traceability of the changed region.

Abstraction Hierarchy: This column discusses whether a process model at different abstraction levels are considered or not in the related work, as in practice process models are usually at different levels.

4.1 Process Model Alignment

One important aspect in propagating changes from one process model to another, is finding corresponding elements in the two process models. Finding corresponding elements in two structures means that for each entity in one structure, a corresponding entity in the other structure, with the same intended meaning can be found. To find the correspondences between models at different abstraction levels, Dijkman et al. [18] used two methods as lexical matching of element pairs and graph matching.

Brockmans et al. [19] proposed a technique for semantic alignment of Petri Net models at different levels of abstraction by translating the process model to OWL. For this purpose they consider the syntactic and semantic similarity between model elements.

Branco et al. [20] deals with management of consistency between models at different levels of abstraction. They proposed an algorithm which establishes corresponding nodes between the models belong to different abstraction levels, based on the type and name of the nodes.

Weidlich et al. [21] represent a framework for matching process model elements. The focus is mainly on addressing the problem of computational complexity of an element from one model which corresponds to multiple elements from another model. Although the approach deals with behavioural consistency between process models, the focus is on inheritance of single process model. We extend their approach to a group of related process models.

Table 1. Comparison of Process Model Change Propagation Approaches in Process Model Alignment

Related Work	*Goal*	*Data Flow*	*Bidirect. Propag.*	*Process View*	*Consist. Pres.*	*Oper.*	*Abstr. Hier.*
Dijkman [18]	Process Alignment	N/A	N/A	+	N/A	N/A	+
Brockmans [19]	Inter-operability	-	N/A	N/A	N/A	N/A	+
Branco [20]	Matching	-	N/A	+	n.d.	-	+
Weidlich [21]	Compatibility	-	N/A	N/A	+	N/A	-
Liu [22]	Create View	-	N/A	+	+	N/A	+
Eshuis [5]	Create View	-	N/A	+	+	N/A	+
Bobrik [23]	Create View	-	N/A	+	+	N/A	+
Zhao [24]	Create View	-	N/A	+	+	N/A	+

Legend: + *supported*, − *not supported*, N/A *not applicable*, n.d. *not discussed.*

Finally, several approaches deal with producing different process views considering different requirements such as, [5, 22–24] while there is no focus on change propagation between these views. Table 1 shows the comparison of the above mentioned approaches.

4.2 Process Co-evolution

Weidlich et al. [3] used *behavioural profile* for identifying the changed regions in a model to propagate the changes between other models at different levels of abstraction. Since any changes in the structure of a process model impacts the relation of nodes, which is behavioural profile, comparing the behavioural profile of the source model and the changed model, the changed region can be identified. In this approach data and properties are not considered. Also the approach depends on the existence of the execution traces of a process model while the computation of the behavioural profile is a costly task.

Fdhila et al. [25] proposed a generic approach to propagate changes in collaborative process scenario. In this scenario different business partners have their own private processes by which a public view is provided for other partners. The proposed approach deals with propagating the changes from a changed view to others preserving the invocation consistency. In compare to our work, in this approach if the consistency of a changed view is not preserved with other views, the change is abandoned. There is no repair plan by which the identified inconsistency can be resolved.

Weidmann et al. [26] has proposed a synchronization approach for process models at different levels of abstraction assuming the model element correspondences exist. The approach focuses on alignment of tasks only and do not consider gateways or tasks properties including data flows.

Dam et al. [27] proposed an agent oriented framework to fix the inconsistencies resulted from change propagation. For this purpose they have created a library including the plans for fixing the constraints violation during the run time. Although the completeness and correctness of generated plans are discussed and guaranteed, since all the possible repair plan choices are considered, the approach is not scalable.

Ekanayake et al. [28] presented a semi-automated technique for change propagation across process model versions. To keep the versions synchronised, for changing a fragment, that fragment is locked until the change is propagated to all the other versions containing that changed fragment.

Gerth et al. [29] proposed a language independent framework for change management. The framework uses workflow graphs as an intermediate representation for the process model, by which the list of differences, dependencies, and conflicts are computed. This framework is applicable in the versioning scenarios where different versions must be refinement of the source model. By merging different versions to get the source model, the changes in the views are identified and propagated to the source model. The possibility that change in one version may impact other versions is not considered and also the changes are always propagated from the views to the source model not vice versa.

The process co-evolution approach mentioned above are compared in Table 2.

4.3 Process Model Variability

Configuration is a kind of design activity which has been widely used to manage process model variability in literature. The aim is to design a process model from a set of predefined components which can be connected together in a certain way.

Hallerbach et al. [30] introduces *Provop* framework to tackle with the challenges in the management of process model variability, such as the challenge relates to designing a reference model and its predefined adjustments, such that different variants can be derived by configuring the reference model. The configuration is applied on a collection of change operations for control- and dataflow. The framework can ensure the soundness of process variants belong to a process family from data flow perspective but not the control flow. Becker et al. [31] produces different variants by projecting the model elements from a reference process model.

A comparison of the two approaches mentioned above can be found on the bottom of Table 2.

Table 2. Comparison of Process Model Change Propagation Approaches

Related Work	*Goal*	*Data Flow*	*Bidirect. Propag.*	*Process View*	*Consist. Pres.*	*Oper.*	*Abstr. Hier.*
Process Co-Evolution							
Kurniawan [32]	Refinement Preservation	-	+	-	+	+	+
Küster [33]	Conflict resolution	-	+	N/A	+	+	+
Weidlich [3]	Change Propagation	-	+	N/A	+	+	+
Fdhila [25]	Compatibility	-	-	+	+	+	+
Weidmann [26]	Change Propagation	-	+	-	+	+	+
Dam [27]	Change Propagation	-	+	N/A	+	+	-
Gerth [29]	Change Management	-	-	-	n.d.	+	+
Our Approach	Change Propagation	+	+	+	+	+	+
Process Model Variability							
Hallerbach [30]	Process Configuration	+	+	N/A	+	+	+
Ekanayake [28]	Change Propagation	-	+	N/A	n.d.	n.d	+

Legend: + *supported*, − *not supported*, N/A *not applicable*, n.d. *not discussed.*

5 Conclusion

We presented a framework for process co-evolution that applies the notion of consistent specialisation for on-the-fly change propagation. We showed on an example how rules developed for checking consistent refinement and extension of business processes can be used to propagate changes efficiently and re-establish consistency among a reference process and its views. So far the framework supports change propagation on the model level and the rules have been implemented in a process abstraction framework. For the next step in our research agenda we plan to implement those rules in a collaborative process design environment and investigate conflicts and resolution of changes that have been performed on the same process fragment and at the same time.

References

1. Reichert, M., Weber, B.: Enabling Flexibility in Process-Aware Information Systems. Springer (2012)
2. Gerth, C., Küster, J., Engels, G.: Language-independent change management of process models. In: Schürr, A., Selic, B. (eds.) MODELS 2009. LNCS, vol. 5795, pp. 152–166. Springer, Heidelberg (2009)
3. Weidlich, M., Mendling, J., Weske, M.: Propagating changes between aligned process models. Journal of Systems and Software 85(8), 1885–1898 (2012)
4. Smirnov, S., Reijers, H., Weske, M., Nugteren, T.: Business process model abstraction: a definition, catalog, and survey. Distributed and Parallel Databases 30, 63–99 (2012)
5. Eshuis, R., Grefen, P.: Constructing customized process views. Data & Knowledge Engineering 64, 419–438 (2008)
6. Reichert, M., Kolb, J., Bobrik, R., Bauer, T.: Enabling personalized visualization of large business processes through parameterizable views. In: Proc. ACM SAC, pp. 1653–1660. ACM Press (2012)
7. Mafazi, S., Mayer, W., Grossmann, G., Stumptner, M.: A knowledge-based approach to the configuration of business process model abstractions. In: Int'l Workshop on Knowledge-intensive Business Processes (2012)
8. Weidlich, M., Weske, M., Mendling, J.: Change propagation in process models using behavioural profiles. In: Proc. IEEE SCC, pp. 33–40 (2009)
9. Küster, J.M., Koehler, J., Ryndina, K.: Improving Business Process Models with Reference Models in Business-Driven Development. In: Eder, J., Dustdar, S. (eds.) BPM 2006 Workshops. LNCS, vol. 4103, pp. 35–44. Springer, Heidelberg (2006)
10. Schrefl, M., Stumptner, M.: Behavior-consistent specialization of object life cycles. ACM Trans. Softw. Eng. Methodol. 11(1), 92–148 (2002)
11. Dijkman, R.M., Dumas, M., Ouyang, C.: Semantics and analysis of business process models in BPMN. Information & Software Technology 50(12), 1281–1294 (2008)
12. Cytron, R., Ferrante, J., Rosen, B.K., Wegman, M.N., Zadeck, F.K.: Efficiently computing static single assignment form and the control dependence graph. ACM Trans. Program. Lang. Syst. 13, 451–490 (1991)
13. Weidlich, M., Mendling, J.: Perceived consistency between process models. Information Systems 37(2), 80–98 (2012)
14. Basten, T., van der Aalst, W.M.: Inheritance of behavior. The Journal of Logic and Algebraic Programming 47(2), 47–145 (2001)
15. Sadiq, S., Orlowska, M., Sadiq, W., Foulger, C.: Data flow and validation in workflow modelling. In: Proc. Australasian DB Conf., pp. 207–214 (2004)

16. Weber, B., Rinderle, S., Reichert, M.: Change patterns and change support features in process-aware information systems. In: Krogstie, J., Opdahl, A.L., Sindre, G. (eds.) CAiSE 2007. LNCS, vol. 4495, pp. 574–588. Springer, Heidelberg (2007)
17. Fahland, D., Favre, C., Jobstmann, B., Koehler, J., Lohmann, N., Völzer, H., Wolf, K.: Instantaneous Soundness Checking of Industrial Business Process Models. In: Dayal, U., Eder, J., Koehler, J., Reijers, H.A. (eds.) BPM 2009. LNCS, vol. 5701, pp. 278–293. Springer, Heidelberg (2009)
18. Dijkman, R., Dumas, M., Garcia-Banuelos, L., Kaarik, R.: Aligning business process models. In: Proc. of EDOC, pp. 45–53 (2009)
19. Brockmans, S., Ehrig, M., Koschmider, A., Oberweis, A., Studer, R.: Semantic alignment of business processes. In: Proc. of ICEIS, pp. 191–196 (2006)
20. Castelo Branco, M., Troya, J., Czarnecki, K., Küster, J., Völzer, H.: Matching business process workflows across abstraction levels. In: France, R.B., Kazmeier, J., Breu, R., Atkinson, C. (eds.) MODELS 2012. LNCS, vol. 7590, pp. 626–641. Springer, Heidelberg (2012)
21. Weidlich, M., Dijkman, R., Weske, M.: Deciding behaviour compatibility of complex correspondences between process models. In: Hull, R., Mendling, J., Tai, S. (eds.) BPM 2010. LNCS, vol. 6336, pp. 78–94. Springer, Heidelberg (2010)
22. Liu, D.R., Shen, M.: Workflow modeling for virtual processes: an order-preserving process-view approach. Information Systems 28(6), 505–532 (2003)
23. Bobrik, R., Reichert, M., Bauer, T.: View-based process visualization. In: Alonso, G., Dadam, P., Rosemann, M. (eds.) BPM 2007. LNCS, vol. 4714, pp. 88–95. Springer, Heidelberg (2007)
24. Zhao, X., Liu, C., Sadiq, W., Kowalkiewicz, M.: Process view derivation and composition in a dynamic collaboration environment. In: Meersman, R., Tari, Z. (eds.) OTM 2008, Part I. LNCS, vol. 5331, pp. 82–99. Springer, Heidelberg (2008)
25. Fdhila, W., Rinderle-Ma, S., Reichert, M.: Change propagation in collaborative processes scenarios. In: Proc. CollaborateCom 2012, pp. 452–461 (2012)
26. Weidmann, M., Alvi, M., Koetter, F., Leymann, F., Renner, T., Schumm, D.: Business process change management based on process model synchronization of multiple abstraction levels. In: Proc. SOCA. IEEE Computer Society (2011)
27. Dam, K.H., Winikoff, M.: Generation of repair plans for change propagation. In: Luck, M., Padgham, L. (eds.) AOSE 2007. LNCS, vol. 4951, pp. 132–146. Springer, Heidelberg (2008)
28. Ekanayake, C.C., La Rosa, M., ter Hofstede, A.H.M., Fauvet, M.-C.: Fragment-based version management for repositories of business process models. In: Meersman, R., Dillon, T., Herrero, P. (eds.) OTM 2011, Part I. LNCS, vol. 7044, pp. 20–37. Springer, Heidelberg (2011)
29. Gerth, C.: Business Process Models. LNCS, vol. 7849. Springer, Heidelberg (2013)
30. Hallerbach, A., Bauer, T., Reichert, M.: Capturing variability in business process models: the Provop approach. Journal of Software Maintenance and Evolution: Research and Practice 22(6-7), 519–546 (2010)
31. Becker, J., Delfmann, P., Knackstedt, R.: Adaptive reference modeling: Integrating configurative and generic adaptation techniques for information models. In: Reference Modeling, pp. 27–58. Physica-Verlag HD (2007)
32. Kurniawan, T.A., Ghose, A.K., Dam, H.K., Lê, L.-S.: Relationship-preserving change propagation in process ecosystems. In: Liu, C., Ludwig, H., Toumani, F., Yu, Q. (eds.) ICSOC 2012. LNCS, vol. 7636, pp. 63–78. Springer, Heidelberg (2012)
33. Küster, J., Gerth, C., Engels, G.: Dependent and conflicting change operations of process models. In: Paige, R.F., Hartman, A., Rensink, A. (eds.) ECMDA-FA 2009. LNCS, vol. 5562, pp. 158–173. Springer, Heidelberg (2009)

Online Process Discovery to Detect Concept Drifts in LTL-Based Declarative Process Models

Fabrizio Maria Maggi[1], Andrea Burattin[2], Marta Cimitile[3], and Alessandro Sperduti[2]

[1] University of Tartu, Estonia
f.m.maggi@ut.ee
[2] University of Padova, Italy
{burattin,sperduti}@math.unipd.it
[3] Unitelma Sapienza University, Italy
marta.cimitile@unitelma.it

Abstract. Today's business processes are often controlled and supported by information systems. These systems record real-time information about business processes during their executions. This enables the analysis at runtime of the process behavior. However, many modern systems produce "big data", i.e., collections of data sets so large and complex that it becomes impossible to store and process all of them. Moreover, few processes are in steady-state and due to changing circumstances processes evolve and systems need to adapt continuously. In this paper, we present a novel framework for the discovery of LTL-based declarative process models from streaming event data in settings where it is impossible to store all events over an extended period or where processes evolve while being analyzed. The framework continuously updates a set of valid business constraints based on the events occurred in the event stream. In addition, our approach is able to provide meaningful information about the most significant concept drifts, i.e., changes occurring in a process during its execution. We report about experimental results obtained using logs pertaining the health insurance claims handling in a travel agency.

Keywords: Process Discovery, Event Stream Analysis, Operational Support, Concept Drift, Linear Temporal Logic, Business Constraints, Declare.

1 Introduction

Together with conformance checking and process model enhancement, process discovery is one of the three basic forms of process mining [25]. In particular, process discovery aims at producing a process model based on example executions in an event log and without using any a-priori information. When using event logs recorded by information systems supporting changeable and highly dynamic processes (e.g., healthcare processes), traditional process discovery techniques (mainly based on procedural process modeling languages) often produce the so called spaghetti-like models. In these models, too many execution paths are explicitly represented so that they become completely unreadable.

In the last years, several works have been focused on the discovery of declarative process models, i.e., on the discovery of a set of business constraints that hold during the process execution [12,8,10,17,15,16]. These techniques are very suitable for processes characterized by high complexity and variability due to the turbulence and the changeability of their execution environments. One of the challenges in process mining listed in the Process Mining Manifesto[1] is *to find a suitable representational bias to visualize the resulting models but also to be used internally when searching for a model.* In this sense, the dichotomy procedural versus declarative can be seen as a guideline when choosing the most suitable language to represent models resulting from process discovery algorithms: process mining techniques based on procedural languages can be used for predictable processes working in stable environments (e.g., a process for handling travel requests), whereas techniques based on declarative languages can be used for unpredictable, variable processes working in turbulent environments (see also [19]).

In this paper, business constraints are represented using the Declare notation [26]. Declare is a declarative language that combines a formal semantics grounded in Linear Temporal Logic (LTL) on finite traces,[2] with a graphical representation. In essence, a Declare model is a collection of LTL rules, each capturing a control flow dependency between two activities.

The Process Mining Manifesto also states that *process mining should not be restricted to off-line analysis and can also be used for online operational support.* In particular, process mining techniques should be able to mine an event stream, i.e., a *real-time, continuous, ordered sequence of items* [13]. Algorithms that are supposed to interact with event streams must respect some requirements, such as: a) it is impossible to store the complete stream; b) backtracking over an event stream is not feasible, so algorithms are required to make only one pass over data; c) it is important to quickly adapt the model to cope with unusual data values; d) the approach must deal with variable system conditions, such as fluctuating stream rates. Currently, only few algorithms are able to mine an event stream [7,6]. In addition, this is the first paper that presents algorithms for discovering declarative process models based on streaming event data. In particular, our technique is able to show a snapshot of the business constraints valid at a certain point in time during the process execution, thus providing the user with a continuously updated picture of the process behavior.

Another challenge mentioned in the Process Mining Manifesto is related to the fact that *process mining must deal with concept drifts*, i.e., with the situation in which the process is changing while being analyzed. Processes may change due to periodic/seasonal changes (e.g., "in December there is more demand" or "on Friday afternoon there are fewer employees available") or due to changing

[1] The Process Mining Manifesto is authored by the IEEE Task Force on Process Mining (`http://www.win.tue.nl/ieeetfpm/`).

[2] For compactness, we will use the LTL acronym to denote LTL on finite traces.

conditions (e.g., “the market is getting more competitive”). Such changes impact processes and it is vital to detect and analyze them. In this paper, we identify concept drifts in a Declare model in terms of constraint activations. Activations for Declare constraints are defined in [5]. For example, for a business constraint like “every request is eventually acknowledged” each request is an activation. An activation becomes a fulfillment or a violation depending on whether the request is followed by an acknowledgement or not. Concept drifts are discovered by analyzing how the percentages of activations, fulfillments and violations for a set of Declare constraints vary over the time.

To assess the applicability of our approach, we conducted an experimentation with a set of synthetic logs. In particular, we used two variants of a process pertaining the health insurance claims handling in a travel agency.

The paper is structured as follows. Section 2 introduces the characteristics of event stream mining as well as some basic notation about Declare. Next, Section 3 introduces the proposed algorithms for the online discovery of Declare models. Section 4 illustrates the two approaches for stream mining we use in this paper based on sliding window and lossy counting respectively. Section 5 presents and discusses the experimental results. Finally, Section 6 concludes and spells out directions for future work.

2 Preliminaries

In this section, we introduce some preliminary notions. In particular, in Section 2.1, we summarize some basic information about event stream mining needed to understand the paper. In Section 2.2, we give an overview of the Declare language.

2.1 Event Stream Mining

In the data mining literature, there are few definitions of event stream. In this work, we consider an event stream as an unbounded, sequence of data items, observed at a high speed [2,3]. Stream mining approaches can be divided into two main categories: *data-based* and *task-based* [11]. Data-based mining algorithms reduce the stream into finite datasets, which are supposed to be representatives of the complete stream; task-based algorithms are modified (or new) approaches, specifically designed for streams, in order to minimize time and space complexity. The framework presented here falls into this latter category.

As typically reported in the literature, we assume that: *i*) the data that constitutes the stream (i.e., the *events*) have a small and fixed amount of attributes; *ii*) a mining approach should be able to analyze an infinite amount of data; *iii*) a mining approach is allowed to use only a finite amount of memory and, typically, such amount is considerably smaller with respect to the data observed in a reasonable span of time; *iv*) there is an upper bound on the time allowed to analyze an event, typically the mining approach is required to linearly scale with the number of processed items (e.g., the algorithm works with one pass of

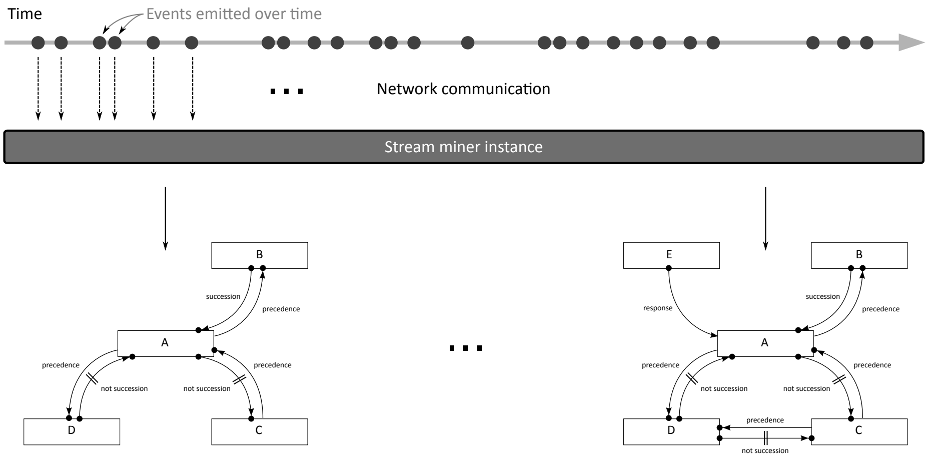


Fig. 1. General idea of event stream mining: the stream miner continuously receives events and, using the latest observations, updates the process model

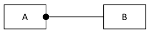
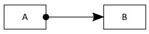
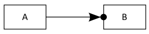
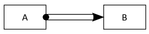
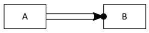
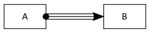
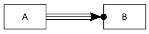
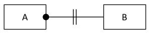
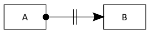
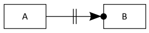
the data [21]); *v*) the "concepts" generating the event stream may be *stationary* or *evolving* [28,30]. We assume that each event in an event stream is associated with an activity (i.e., a well-defined step in the process), is related to a particular case (i.e., a process instance) and is executed at a certain point in time specified through a timestamp.

In this work, our aim is to reconstruct a declarative model of the process generating the stream, while storing a minimal amount of information. Figure 1 proposes a simple representation of our approach: one or more sources emit events (represented, in the picture, as solid dots), which are collected by our miner instance. The miner elaborates these events and keeps the process model updated. In addition, we want to characterize the portions of the event stream in which the process behavior changes in terms of business constraints. Therefore, it is important to evaluate whether our approach is able to correctly detect concept drifts.

2.2 Declare: Some Basic Notions

Declare is a declarative process modeling language introduced by Pesic and van der Aalst in [26]. A Declare model consists of a set of constraints which, in turn, are based on templates. Templates are abstract entities that define parameterized classes of properties and constraints are their concrete instantiations. Here, we indicate template parameters with capital letters (see Table 1) and real activities in their instantiations with lower case letters (e.g., constraint $\Box(a \rightarrow \Diamond b)$). Templates have a user-friendly graphical representation understandable to the user and their semantics are specified through LTL formulas. Each constraint inherits the graphical representation and semantics from its template. The most frequently used Declare templates are shown in Table 1. However, the language

Table 1. Graphical notation and textual description of some Declare constraints

Template	Meaning	LTL semantics	Graphical notation
responded existence(A,B)	if A occurs then B occurs before or after A	$\Diamond A \rightarrow \Diamond B$	
response(A,B)	if A occurs then eventually B occurs after A	$\Box(A \rightarrow \Diamond B)$	
precedence(A,B)	if B occurs then A occurs before B	$(\neg B \sqcup A) \vee \Box(\neg B)$	
alternate response(A,B)	if A occurs then eventually B occurs after A without other occurrences of A in between	$\Box(A \rightarrow \bigcirc(\neg A \sqcup B))$	
alternate precedence(A,B)	if B occurs then A occurs before B without other occurrences of B in between	$((\neg B \sqcup A) \vee \Box(\neg B)) \wedge$ $\Box(B \rightarrow \bigcirc((\neg B \sqcup A) \vee \Box(\neg B)))$	
chain response(A,B)	if A occurs then B occurs in the next position after A	$\Box(A \rightarrow \bigcirc B)$	
chain precedence(A,B)	if B occurs then A occurs in the next position before B	$\Box(\bigcirc B \rightarrow A)$	
not responded existence(A,B)	if A occurs then B cannot occur before or after A	$\Diamond A \rightarrow \neg(\Diamond B)$	
not response(A,B)	if A occurs then B cannot eventually occur after A	$\Box(A \rightarrow \neg(\Diamond B))$	
not precedence(A,B)	if B occurs then A cannot occur before B	$\Box(\Diamond B \rightarrow \neg A)$	

is extensible and new templates can be defined with their own graphical representations and LTL semantics.

Consider, for example, the *response* constraint $\Box(a \rightarrow \Diamond b)$. This constraint indicates that if *a occurs*, b must eventually *follow*. Therefore, this constraint is satisfied for traces such as $\mathbf{t}_1 = \langle a, a, b, c\rangle$, $\mathbf{t}_2 = \langle b, b, c, d\rangle$, and $\mathbf{t}_3 = \langle a, b, c, b\rangle$, but not for $\mathbf{t}_4 = \langle a, b, a, c\rangle$ because, in this case, the second instance of a is not followed by a b. Note that, in $\mathbf{t}_2$, the considered response constraint is satisfied in a trivial way because a never occurs. In this case, we say that the constraint is *vacuously satisfied* [14]. In [5], the authors introduce the notion of *behavioral vacuity detection* according to which a constraint is non-vacuously satisfied in a trace when it is activated in that trace. An *activation* of a constraint in a trace is an event whose occurrence imposes, because of that constraint, some obligations on other events in the same trace. For example, a is an activation for the *response* constraint $\Box(a \rightarrow \Diamond b)$, because the execution of a forces b to be executed eventually.

An activation of a constraint can be a *fulfillment* or a *violation* for that constraint. When a trace is perfectly compliant with respect to a constraint, every activation of the constraint in the trace leads to a fulfillment. Consider, again, the response constraint $\Box(a \rightarrow \Diamond b)$. In trace $\mathbf{t}_1$, the constraint is activated and fulfilled twice, whereas, in trace $\mathbf{t}_3$, the same constraint is activated and fulfilled only once. On the other hand, when a trace is not compliant with respect to a constraint, an activation of the constraint in the trace can lead to a fulfillment but also to a violation (at least one activation leads to a violation). In trace $\mathbf{t}_4$, for example, the response constraint $\Box(a \rightarrow \Diamond b)$ is activated twice, but the first activation leads to a fulfillment (eventually b occurs) and the second activation leads to a violation (b does not occur subsequently). An algorithm to discriminate between fulfillments and violations for a constraint in a trace is presented in [5].

In [5], the authors define two metrics to measure the conformance of an event log with respect to a constraint in terms of violations and fulfillments, called *violation ratio* and *fulfillment ratio* of the constraint in the log. These metrics are valued 0 if the log contains no activations of the considered constraint. Otherwise, they are evaluated as the percentage of violations and fulfillments of the constraint over the total number of activations.

3 Algorithms for the Online Discovery of Declare Models

In this section, we describe the proposed algorithms for the online discovery of Declare constraints. In particular, we will illustrate three different basic algorithms. Each of them can be used to discover constraints referring to different Declare templates. Every algorithm takes as input an event stream and builds a set of candidate constraints obtained by instantiating a Declare template with all the possible combinations of activity names detected so far in the event stream. The algorithms keep up-to-date the fulfillment ratio and the violation ratio of each candidate constraint at every occurrence of an event in the event stream. Then, at each point in time, it is possible to discover a Declare model by selecting, among the candidates, the ones with the highest fulfillment ratio.

In all the algorithms, we use the notion of *map*. Here, a *map* refers to the well known *hash table* data structure [9]. Given a set of keys K and a set of values V, a map is a set $M \subseteq K \times V$. We use the following operators: *(i)* $M.\text{put}(k, v)$, to add value v with key k to M; *(ii)* $M.\text{get}(x) \in V$, with $k \in K$, to retrieve from M the value associated with key k; *(iii)* $M.\text{keys} \subseteq K$ to obtain the set of keys in M.

3.1 Response and Not Response

The algorithm presented in this section can be used for the online discovery of *response* and *not response* constraints. We illustrate the algorithm (Algorithm 1) for the response template. The algorithm is identical for the not response template, since the fulfillment ratio for a response constraint corresponds to the violation ratio of the not response constraint over the same activities (and vice versa).[3] A similar algorithm (with small modifications) is able to discover *alternate response* and *chain response* constraints.

In this algorithm, L is the set of all the activity names observed in the event stream (in all the cases). $activationsCounter_c$ is a map defined for each case c and containing, for each activity name, the number of its occurrences in c. This map can be used to count the number of activations for each constraint (the number of activations can be obtained by counting how many times the corresponding activity name has occurred in each case of the event stream).

[3] A similar observation also applies to the algorithms described in the following sections.

Algorithm 1. Online algorithm for response and not response

```
Input: $e = (c, a, t)$ the event to be processed ($c$ is the case id of the event, $a$ is the activity
       name, $t$ is the timestamp)
1  if $L$ is not defined then define the empty set $L$
2  if $activationsCounter_c$ is not defined then define the map $activationsCounter_c$
3  if $pendingActivations_c$ is not defined then define the map $pendingActivations_c$
4  if $a \notin activationsCounter_c$.keys then
5      $activationsCounter_c$.put$(a, 1)$
6      foreach $l \in L$ do
7          $pendingActivations_c$.put$((l, a), 0)$
8          $pendingActivations_c$.put$((a, l), 1)$
9  else
10     $activationsCounter_c$.put$(a, activationsCounter_c$.get$(a) + 1)$
11     foreach $(k_1, k_2) \in pendingActivations_c$.keys do
12         if $k_2 = a$ then                    /* $a$ equals to the second activity name */
13             $pendingActivations_c$.put$((k_1, a), 0)$
14         else if $k_1 = a$ then               /* $a$ equals to the first activity name */
15             $acts \leftarrow pendingActivations_c$.get$((a, k_2))$
16             $pendingActivations_c$.put$((a, k_2), acts + 1)$
17 if $a \notin L$ then $L \leftarrow L \cup \{a\}$
```

pendingActivations$_c$ is a map defined for each case c and containing the number of pending activations in c for each response constraint activated in c (the keys in the map are pairs of activity names representing the constraint parameters).

The algorithm receives as input an event $e = (c, a, t)$, where c is the case id, a is the activity name and t is the timestamp. This event is processed by updating maps *activationsCounter*$_c$ and *pendingActivations*$_c$. If c is a new case, these maps are first defined (lines 2-3).

Then, if a has never occurred in c (i.e., if a is not in *activationsCounter*$_c$), a is added as a key in *activationsCounter*$_c$ with number of occurrences equal to 1 (a has occurred only once in c). Lines 6-8 of the algorithm are used to update map *pendingActivations*$_c$ when a occurs in c for the first time. In particular, all the response constraints having a as second parameter cannot have any pending activation when a occurs (all of them are fulfilled when a occurs). Therefore, line 7 of the algorithm sets to 0 the number of pending activations in c for the response constraints having a as second parameter. On the other hand, all the response constraints having a as first parameter are activated when a occurs and waiting for some other event to occur. Then, a is a pending activation for these constraints (and the only one). Therefore, line 8 of the algorithm sets to 1 the number of pending activations in c for the response constraints having a as first parameter.

If a has already occurred in c (i.e., if a is already in *activationsCounter*$_c$), the number of occurrences of a is incremented by 1 in *activationsCounter*$_c$. Line 13 of the algorithm sets to 0 the pending activations in c for the response constraints having a as second parameter. On the other hand, all the response constraints having a as first parameter are activated when a occurs and, therefore, the number of pending activations for these constraints in c is incremented by 1.

Algorithm 2. Online algorithm for precedence and not precedence

```
Input: $e = (c, a, t)$ the event to be processed ($c$ is the case id of the event, $a$ is the activity
       name, $t$ is the timestamp)
if $L$ is not defined then define the empty set $L$
if $activationsCounter_c$ is not defined then define the map $activationsCounter_c$
if $fulfilledActivations_c$ is not defined then define the map $fulfilledActivations_c$
if $a \notin activationsCounter_c$ then
    $activationsCounter_c$.put($a$, 1)
    foreach $l \in L$ do
        if $l \in activationsCounter_c$ then
            $fulfilledActivations_c$.put(($l, a$), 1)
            $fulfilledActivations_c$.put(($a, l$), 0)
        else
            $fulfilledActivations_c$.put(($l, a$), 0)
            $fulfilledActivations_c$.put(($a, l$), 0)
else
    $activationsCounter_c$.put($a$, $activationsCounter_c$.get($a$) + 1)
    foreach $l \in L$ do
        if $l \in activationsCounter_c$ then
            $acts \leftarrow fulfilledActivations_c$.get(($l, a$))
            $fulfilledActivations_c$.put(($l, a$), $acts + 1$)
if $a \notin L$ then $L \leftarrow L \cup \{a\}$
```

Finally (line 17), if activity name a is observed for the first time in the event stream, a is added to L.

Using the algorithm just described it is possible to count the number of activations for every candidate response constraint (using maps $activationsCounter_c$), and the number of violations (counting the number of activations still pending in maps $pendingActivations_c$). These metrics are enough for deriving, at any point in time, *fulfillment ratio* and *violation ratio* for (not) response constraints.

3.2 Precedence and Not Precedence

The algorithm presented in this section can be used for the online discovery of *precedence* and *not precedence* constraints. We illustrate the algorithm (Algorithm 2) for the precedence template. A variant of this algorithm can be easily derived for the online discovery of *alternate precedence* and *chain precedence* constraints.

L is again the set of all the activity names occurred in the event stream (in all the cases). $activationsCounter_c$ is a map containing, for each activity name, the number of its occurrences in c. $fulfilledActivations_c$ is a map defined for each case c and containing the number of fulfilled activations in c, for each precedence constraint activated in c (the keys in the map are pairs of activity names representing the constraint parameters).

The algorithm receives as input an event $e = (c, a, t)$ belonging to a case c. This event is processed by updating maps $activationsCounter_c$ and $fulfilledActivations_c$. If c is a new case, these maps are first defined (lines 2-3).

If a has never occurred in c, a is added as a key in $activationsCounter_c$ with number of occurrences equal to 1. Lines 6-12 are used to update map $fulfilledActivations_c$ when a occurs in c for the first time. In particular, for each activity l in L, the precedence constraint having a as first parameter and l as second parameter has no fulfillments. Indeed, this constraint is activated when l occurs and is fulfilled if l is preceded by a. This cannot be the case because a has occurred now for the first time. Therefore, lines 9 and 12 of the algorithm sets to 0 the number of fulfilled activations in c for the precedence constraints having l as second parameter. For each activity l in L, the precedence constraints having l as first parameter and a as second parameter are activated when a occurs. In particular, a is a fulfillment if and only if l has already occurred in c. Only in this case, the algorithm sets to 1 the number of fulfilled activations (line 8). Otherwise, the number of fulfilled activations is 0 (line 11).

If a has already occurred in c, the number of occurrences of a is incremented by 1 in $activationsCounter_c$. For each event l in L, the precedence constraints having l as first parameter and a as second parameter are activated when a occurs. Therefore, if l has already occurred in c (and only in this case), the algorithm increments by 1 the number of fulfilled activations in c for the precedence constraints having a as second parameter (line 18). If activity name a has occurred for the first time in the event stream, a is added to L.

Using the algorithm just described it is possible to count the number of activations for every candidate precedence constraint (using maps $activationsCounter_c$), and the number of fulfillments (counting the number of fulfilled activations in maps $fulfilledActivations_c$). With this information it is possible to derive, at any point in time, *fulfillment ratio* and *violation ratio* for (not) precedence constraints.

3.3 Responded Existence and Not Responded Existence

We use Algorithm 3 to discover *responded existence* and *not responded existence* constraints. We illustrate the algorithm for the responded existence template.

As in the algorithms already discussed, also for this algorithm, L is the set of all the activity names occurred in the event stream. $activationsCounter_c$ contains, for each activity name, the number of its occurrences in c. $pendingActivations_c$ contains the number of pending activations in c for each responded existence constraint activated in c.

The algorithm receives as input an event $e = (c, a, t)$ belonging to a case c. This event is processed by updating maps $activationsCounter_c$ and $pendingActivations_c$. If c is a new case, these maps are first defined (lines 2-3).

Then, if activity a has never occurred in c, a is added as a key in $activationsCounter_c$ with number of occurrences equal to 1. Lines 6-11 of the algorithm are used to update map $pendingActivations_c$ when activity a occurs in c for the first time. In particular, all the responded existence constraints having a as second parameter cannot have pending activations (all of them are fulfilled when a occurs). Therefore, line 7 of the algorithm sets to 0 the pending activations in c for the responded existence constraints having a as second parameter. On the

Algorithm 3. Algorithm for responded existence and not resp. existence

```
Input: $e = (c, a, t)$ the event to be processed ($c$ is the case id of the event, $a$ is the activity
       name, $t$ is the timestamp)
1  if $L$ is not defined then define the empty set $L$
2  if $activationsCounter_c$ is not defined then define the map $activationsCounter_c$
3  if $pendingActivations_c$ is not defined then define the map $pendingActivations_c$
4  if $a \notin activationsCounter_c$ then
5      $activationsCounter_c$.put($a$, 1)
6      foreach $l \in L$ do
7          $pendingActivations_c$.put(($l$, $a$), 0)
8          if $l \in activationsCounter_c$ then
9              $pendingActivations_c$.put(($a$, $l$), 0)
10         else
11             $pendingActivations_c$.put(($a$, $l$), 1)
12 else
13     $activationsCounter_c$.put($a$, $activationsCounter_c$.get($a$) + 1)
14     foreach $(k_1, k_2) \in pendingActivations_c$.keys do
15         if $k_2 = a$ then                                /* $a$ equals to the second event */
16             $pendingActivations_c$.put(($k_1$, $a$), 0)
17         else if $k_1 = a$ then                           /* $a$ equals to the first event */
18             if $k_2 \in activationsCounter_c$ then
19                 $pendingActivations_c$.get(($a$, $k_2$))
20             else
21                 $acts \leftarrow pendingActivations_c$.get(($a$, $k_2$))
22                 $pendingActivations_c$.put(($a$, $k_2$), $acts + 1$)
23 if $a \notin L$ then $L \leftarrow L \cup \{a\}$
```

other hand, for each event l in L, the responded existence constraints having a as first parameter and l as second parameter are activated when a occurs. In particular, a is a fulfillment if and only if l has already occurred in c. In this case, the algorithm sets to 0 the number of pending activations in c (line 9). Otherwise, if l has not occurred in c yet, a is a pending activation and the algorithm sets to 1 the number of pending activations (line 11).

If activity name a has already occurred in c, the number of occurrences of a is incremented by 1 in $activationsCounter_c$. Line 16 of the algorithm sets to 0 the pending activations in c for the responded existence constraints having a as second parameter (there are no longer pending activations for these constraints when a occurs). All the responded existence constraints having a as first parameter are activated when a occurs and, therefore, the number of pending activations for these constraints in c is set to 0 if the second parameter has already occurred and is incremented by 1 if the second parameter has not occurred yet. Finally (line 23), if activity name a has occurred for the first time in the event stream, a is added to L.

4 Stream Mining Algorithms

As mentioned earlier in this paper, an event stream is an infinite sequence of events. Due to the infinite amount of cases we need to cope with and due to the finite memory size available, we need approaches to remove less significant cases

Algorithm 4. Stream Mining with Sliding Window

```
Input: $S$: event stream; $max_M$: maximum size of the memory
1 Let $M$ be the available memory
2 forever do
3     $e \leftarrow observe(S)$                      /* Observe a new event $e = (c, a, t)$ */
      /* Check if event $e$ has to be used                                  */
4     if $analyze(e)$ then
          /* Memory update                                                  */
5         if $size(M) = max_M$ then
6             $shift(M)$
7         $insert(M, e)$
          /* Mining update                                                  */
8         if $perform\ mining$ then
9             $DeclareMiner(M)$
```

and keep only the most representative ones. In this work, we use two approaches to deal with infinite streams: sliding window and an adaptation of lossy counting [18]. The first approach is very simple and effective but, as shown in our experimentation, not very efficient; the second approach is more complex but guarantees better performance and, at the same time, good quality of the results. In general, both these approaches give us strategies that guide the decision about which information is not useful anymore and, therefore, can be forgotten.

4.1 Sliding Window

One of the simplest way to tackle the stream mining problem is to collect events for a certain observation period and apply the “off-line version” of a discovery algorithm on the collected data. This idea is presented in Algorithm 4 and the approach is called *sliding window*. An event $e = (c, a, t)$ observed in a stream S (line 3) is analyzed (line 4) to decide whether it will be considered for mining.[4] If the *analyze* function returns true, the algorithm inserts the new event into the memory M (line 7). When M reaches its maximum size (line 5), it is necessary to delete the oldest event (line 6). Periodically (e.g., after the observation of a certain number of events), it is possible to perform the discovery. In our case, we can run the Declare Miner [15], but any other discovery algorithm can be applied.

The main drawback of the sliding window approach is that the time required for processing an event is completely unbalanced and strongly depends on the event processed. In particular, when a new event is observed, most of the times, only inexpensive operations are performed (i.e., $insert(M, e)$). However, when the model must be updated, the log retained in memory is mined from scratch. This implies that, in this case, the event is handled at least twice: the first time to store it in the memory M and the second time by the discovery algorithm.

[4] In general, all the incoming events are analyzed. Only in extreme cases in which it is not possible to store further events in memory, the new events are discarded.

Algorithm 5. Stream Mining with Lossy Counting

```
   Input: $S$ event stream; $\epsilon$ maximum allowed error; $\mathcal{T}$ template.
1  $N \leftarrow 1$
2  $w \leftarrow \lceil \frac{1}{\epsilon} \rceil$                              /* Define the bucket width */
3  $\mathcal{D}_\mathcal{T} = \emptyset$                    /* Main data structure, its type is
   $\mathcal{D}$ : case id $\times$ replayer for template $\mathcal{T}$ $\times$ frequency $\times$ maximum error */
4  forever do
5      $b_{current} = \lceil \frac{N}{w} \rceil$                     /* Define the current bucket id */
6      $e \leftarrow observe(S)$          /* Observe a new event, where $e = (c, a, t)$ */
       /* Update data structure $\mathcal{D}_\mathcal{T}$                                     */
7      if $\exists$ $(c_{candidate}, r, f, \Delta) \in \mathcal{D}_\mathcal{T}$ such that $c_{candidate} = c$ then
8          Process event $e$ on replayer $r$
9          Update the $(c, r, f, \Delta)$ tuple of $\mathcal{D}_\mathcal{T}$, by incrementing $f$ by 1
10     else
11         $r_\mathcal{T} \leftarrow$ new replayer for template $\mathcal{T}$
12         $\mathcal{D}_\mathcal{T} \leftarrow \mathcal{D}_\mathcal{T} \cup \{(c, r_\mathcal{T}, 1, b_{current} - 1)\}$
       /* Periodic cleanup                                                 */
13     if $N = 0 \mod w$ then
14         foreach $(c_{iter}, r, f, \Delta) \in \mathcal{D}_\mathcal{T}$ such that $f + \Delta \leq b_{current}$ do
15             Remove $(c_{iter}, r, f, \Delta)$ from $\mathcal{D}_\mathcal{T}$
       /* Generate model                                                   */
16     if model then
17         Extract constraints from $\mathcal{D}_\mathcal{T}$
18     $N \leftarrow N + 1$                       /* Increment the buckets counter */
```

Generally speaking, however, an off-line algorithm may iterate several times on a log. This issue is critical since, in online settings, for performance reasons, it is desirable a procedure that analyze each event no more than once.

4.2 Lossy Counting

One of the strongest approach that can be used to solve the approximate frequency counting problem is called *lossy counting*. In this section, we present a modified version of lossy counting that can be used for the discovery of Declare models. The entire procedure is presented in Algorithm 5.

The basic idea of lossy counting is to divide the stream into ideal buckets, each of them with size $w = \lceil \frac{1}{\epsilon} \rceil$, where $\epsilon \in (0, 1)$ is a parameter indicating the maximum allowed error (0 means that no data is discarded, leading to large space usage; 1 indicates that almost all historical data is discarded, leading to less reliable results). The *current* bucket (i.e., the bucket containing the last event observed) is $b_{current} = \lceil \frac{N}{w} \rceil$, where N is the number of events observed so far.

The most important data structure used by this approach is a set of entries of the form (c, r, f, Δ), where c is the identifier of a case (a case id), r is the replayer associated with a Declare template $\mathcal{T}$ and implementing one of the algorithms illustrated in Section 3, f is the estimated frequency of case c and Δ is the current maximum error. Note that there is a replayer for each case and for each template and each replayer keeps track of the activations, the fulfillments and the violations in that case for that template.

Every time a new element $e = (c, a, t)$ is observed (line 6), the algorithm checks whether the data structure contains an entry for case c (line 7). If such entry exists, then its frequency value f is incremented by one and the replayer processes the new event (lines 8-9). Otherwise a new tuple is added, $(c, r_{\mathcal{T}}, 1, b_{current} - 1)$ (where $r_{\mathcal{T}}$ is a new replayer, for case c) at line 12. Every time $N \equiv 0 \mod w$, the algorithm cleans the data structure by removing the entries that satisfy the inequality $f + \Delta \leq b_{current}$ (line 15). Such condition ensures that, every time the cleanup procedure is executed, $b_{current} \leq \epsilon N$. Periodically (e.g., after the observation of a certain number of events), it is possible to extract the set of valid constraints as described in Section 3 (line 17).

5 Case Study

We implemented the algorithms illustrated in the previous sections as a plug-in of the process mining tool ProM (`http://www.processmining.org`). To carry out our experiments,[5] we have generated two synthetic logs ($\mathcal{L}_1$ and $\mathcal{L}_2$) by modeling two variants of the insurance claim process described in [4] in CPN Tools (`http://cpntools.org`) and by simulating the models. $\mathcal{L}_1$ contains 14,840 events and $\mathcal{L}_2$ contains 16,438 events. We merged the logs (four alternations of $\mathcal{L}_1$ and $\mathcal{L}_2$) using the *Stream Package*, publicly available in the ProM repositories. The same package has been used to transform the resulting log into an event stream.

Starting from the generated stream, we have compared the effectiveness and the efficiency of the online process discovery using the lossy counting approach with respect to the approach based on sliding window. In our experimentation, we discovered a Declare model every 1000 events processed, i.e., we fixed an *evaluation point* every 1000 events.[6]

For evaluating the effectiveness of the two approaches, we have used metrics *precision* and *recall* [20]. To compute recall and precision we assumed that the discovered Declare constraints could be classified into one of four categories, i.e., *i*) true-positive (T_P: correctly discovered); *ii*) false-positive (F_P: incorrectly discovered); *iii*) true-negative (T_N: correctly missed); *iv*) false-negative (F_N: incorrectly missed). Precision and recall are defined as

$$Precision = \frac{T_P}{T_P + F_P} \qquad Recall = \frac{T_P}{T_P + F_N}. \tag{1}$$

The *gold standard* used as reference is the set of all true positive instances. In our experiments, we have used as gold standards two Declare models ($\mathcal{M}_1$ and

[5] All the experiments have been conducted on a machine with an Intel i7 processor (limiting the execution to just one core), 8 GB of RAM and the Oracle Java virtual machine installed on a GNU/Linux Ubuntu operating system.

[6] In all the experiments, we discover (not) response, (not) precedence and (not) responded existence constraints.

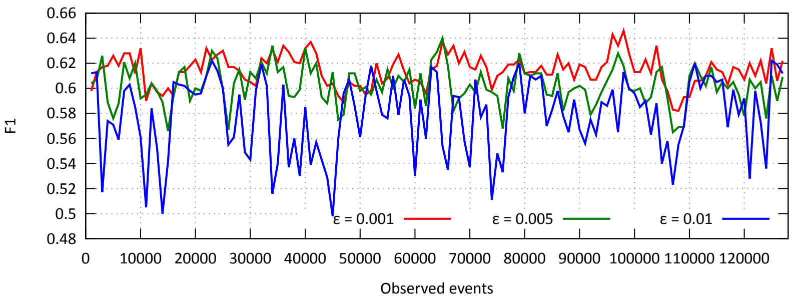


Fig. 2. F_1 trend for the lossy counting approach for different values of the maximum allowed error ϵ

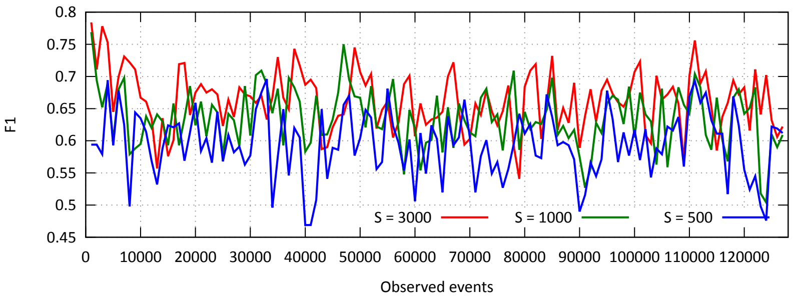


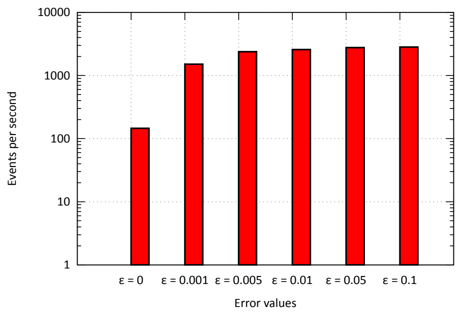
Fig. 3. F_1 trend for the sliding window approach for different values of the window size

$\mathcal{M}_2$) discovered from $\mathcal{L}_1$ and $\mathcal{L}_2$ using the Declare Miner (available in ProM) and containing constraints satisfied in all the cases. Precision and recall have been evaluated in every evaluation point. To compute them, we have used either $\mathcal{M}_1$ or $\mathcal{M}_2$ as gold standard based on whether the evaluation point corresponds to an event belonging to $\mathcal{L}_1$ or $\mathcal{L}_2$ respectively. In every evaluation point, we selected the constraints with fulfillment ratio equal to 1 among the candidate constraints generated by the lossy counting approach and discovered (with the Declare Miner) the constraints with support 100% in the sliding window approach. Then, we compared these sets of constraints with the gold standards. A discovered constraint is classified as true-positive or false-positive depending on whether it belongs to the gold standard or not. A constraint that belongs to the gold standard but that has not been discovered is a false-negative.

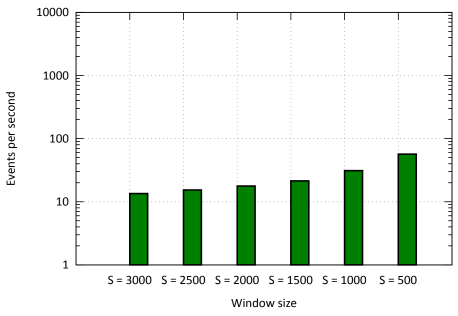
In Fig. 2, we show the trend of the harmonic mean of precision and recall

$$F_1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{2}$$

for the lossy counting approach. The plot shows that the quality of the discovered models in every evaluation point is sufficiently high with respect to the gold standards.



(a) Time efficiency for lossy counting



(b) Time efficiency for sliding window

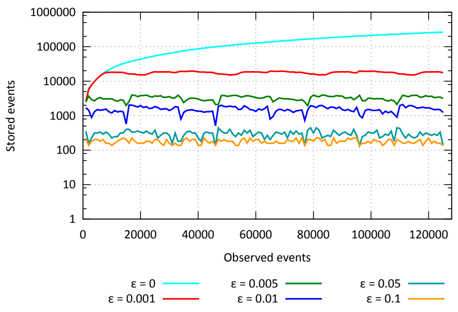
Fig. 4. Time efficiency for lossy counting and sliding window approaches. Both graphs use logarithmic scales

In Fig. 3, we show the trend of F_1 for the sliding window approach. Concerning the configuration parameters chosen for this experiment, the quality of the models generated by the sliding window approach is, on average, higher with respect to the models generated by the lossy counting approach, even if it has a higher variance. However, as shown later in this section, for the same configuration parameters, the approach based on sliding window is extremely more expensive in terms of time efficiency. Note that the values for F_1 are, in both cases, not very high. This is due to the fact that incomplete traces might not be able to provide insight into violations (see [29]). Both approaches are able to detect the concept drifts correctly, since the comparisons with both gold standards lead to good values for F_1.

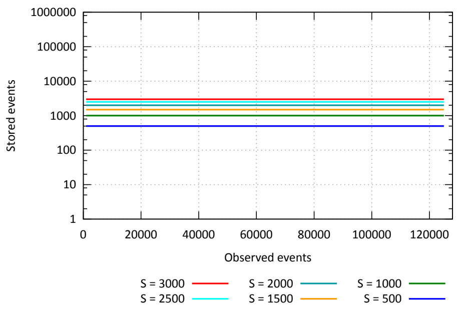
The efficiency of the two approaches has been evaluated through *i*) the number of events processed per second (for evaluating the response times of the two approaches) and *ii*) the number of events stored at each evaluation point (for evaluating the memory size needed). For the lossy counting-based approach, we have evaluated these two metrics for different values of the maximum allowed error. For the sliding window-based approach, we have evaluated he metrics for different sizes of the window.

The results are shown in Fig. 4-5. Plots (a) and (b), in Fig. 4, report the number of events that each approach is able to analyze per second. As expected, the time efficiency in both cases depends on the configuration parameters. For the lossy counting approach, the lower is the allowed error, the greater is the number of replayers to be updated (and this results in a lower efficiency). For the sliding window approach, the larger is the window size, the larger is the log to be mined at every evaluation point (again, this has a negative influence over the time efficiency). In general, however, it is evident that, when trying to obtain models with good quality, the sliding approach becomes much more inefficient than the approach based on lossy counting.

Plots (a) and (b) of Fig. 5 report the space usage for the two approaches to store events. For the lossy counting approach, the larger is the allowed error, the lower is the space required (since less events must be retained). When $\epsilon = 0$, no



(a) Space usage for lossy counting



(b) Space usage for sliding window

Fig. 5. Space usage for lossy counting and sliding window approaches. Both graphs use logarithmic scales

data is discarded and, therefore, the space usage grows linearly (the plot uses a logarithmic scale). In all the other cases, given an error value, the space required is rather constant. Some variations may be due to concept drifts. Indeed, when we pass from $\mathcal{L}_1$ to $\mathcal{L}_2$ in the event stream, cases belonging to the first log are discarded because become out of date. For the sliding window case, the space usage is constant and proportional to the window size.

Table 2. Applicability to the BPI Challenges 2011, 2012 and 2013

Event stream	**No. of activity names**	**Total events processed**	**Total processing time** (secs)	**Milliseconds per event**	**Events per second**
2011 BPI Challenge	623	150 291	2977.238	19.81	50.48
2012 BPI Challenge	36	262 200	191.406	0.73	1369.86
2013 BPI Challenge	13	65 533	37.350	0.57	1754.56

Finally, we checked the applicability of the lossy counting approach to assess its scalability when the number of activities in the process is high and, then, the number of candidate constraints grows. In Table 2, we specify the execution time for the logs provided in the BPI Challenges 2011 [1], 2012 [27] and 2013 [24] (maximum allowed error 0.01). This table shows that the approach is applicable even for streams containing more than 600 activity names. Of course, this solution is more expensive in terms of time and memory. However, to improve this aspect and deal also with these extreme cases, the approach can be adapted by using approaches that try to keep track only of the most interesting candidates. This can be easily done by integrating this approach with approaches to discover frequent item sets like the one proposed in [15]. Another way to save memory in this sense is to remove redundancies as explained in [16]. For example, if a constraint is stronger than another, it is possible to monitor only the strongest one and consider the weakest only if the strongest is violated. Another approach to avoid redundancies and contradictions in the discovered constraints is explained in [22,23].

6 Conclusion

In this paper, we presented a novel framework for the online discovery of declarative process models from streaming event data. The framework is able to produce at runtime an updated picture of the process behavior in terms of LTL-based business constraints. Moreover, it gives to the user meaningful information about the most significant concept drifts occurring during the process execution. The proposed approach combines algorithms for the online discovery of Declare models and algorithms for stream mining. The framework has been implemented in ProM. Our experimentation shows the higher effectiveness of the approach based on sliding window with respect to the approach based on lossy counting at the cost of very poor performances.

As future work, we will conduct a wider experimentation of the proposed framework on several case studies also in real-life scenarios. In this way, it will be possible to understand what can be improved and how. It may be the case, for example, that some templates are more difficult to discover than others. As discussed in Section 5, optimizations are also possible in terms of time efficiency and memory usage.

Acknowledgement. Andrea Burattin and Alessandro Sperduti are supported by the Eurostars-Eureka project PROMPT (E!6696). The authors would like to thank Francesca Rossi and Paolo Baldan for their advice.

References

1. 3TU Data Center. BPI Challenge 2011 Event Log (2011), doi:10.4121/uuid:d9769f3d-0ab0-4fb8-803b-0d1120ffcf54
2. Aggarwal, C.: Data Streams: Models and Algorithms. Advances in Database Systems, vol. 31. Springer, US (2007)
3. Bifet, A., Holmes, G., Kirkby, R., Pfahringer, B.: MOA: Massive Online Analysis Learning Examples. Journal of Machine Learning Research 11, 1601–1604 (2010)
4. Jagadeesh Chandra Bose, R.P.: Process Mining in the Large: Preprocessing, Discovery, and Diagnostics. PhD thesis, Eindhoven University of Technology (2012)
5. Burattin, A., Maggi, F.M., van der Aalst, W.M.P., Sperduti, A.: Techniques for a Posteriori Analysis of Declarative Processes. In: EDOC, pp. 41–50 (2012)
6. Burattin, A.: Applicability of Process Mining Techniques in Business Environments. PhD Thesis, University of Bologna (2013)
7. Burattin, A., Sperduti, A., van der Aalst, W.M.P.: Heuristics Miners for Streaming Event Data. ArXiv CoRR (December 2012)
8. Chesani, F., Lamma, E., Mello, P., Montali, M., Riguzzi, F., Storari, S.: Exploiting Inductive Logic Programming Techniques for Declarative Process Mining. In: Jensen, K., van der Aalst, W.M.P. (eds.) TOPNOC II. LNCS, vol. 5460, pp. 278–295. Springer, Heidelberg (2009)
9. Cormen, T.H., Stein, C., Rivest, R.L., Leiserson, C.E.: Introduction to Algorithms, 2nd edn. The MIT Press (September 2001)
10. Di Ciccio, C., Mecella, M.: Mining constraints for artful processes. In: Abramowicz, W., Kriksciuniene, D., Sakalauskas, V. (eds.) BIS 2012. LNBIP, vol. 117, pp. 11–23. Springer, Heidelberg (2012)

11. Gaber, M.M., Zaslavsky, A., Krishnaswamy, S.: Mining Data Streams: a Review. ACM Sigmod Record 34(2), 18–26 (2005)
12. Goedertier, S., Martens, D., Vanthienen, J., Baesens, B.: Robust process discovery with artificial negative events. JMLR 10, 1305–1340 (2009)
13. Golab, L., Tamer Özsu, M.: Issues in Data Stream Management. ACM SIGMOD Record 32(2), 5–14 (2003)
14. Kupferman, O., Vardi, M.Y.: Vacuity Detection in Temporal Model Checking. Int. Journal on Software Tools for Technology Transfer, 224–233 (2003)
15. Maggi, F.M., Jagadeesh Chandra Bose, R.P., van der Aalst, W.M.P.: Efficient discovery of understandable declarative process models from event logs. In: Ralyté, J., Franch, X., Brinkkemper, S., Wrycza, S. (eds.) CAiSE 2012. LNCS, vol. 7328, pp. 270–285. Springer, Heidelberg (2012)
16. Maggi, F.M., Jagadeesh Chandra Bose, R.P., van der Aalst, W.M.P.: A knowledge-based integrated approach for discovering and repairing declare maps. In: Salinesi, C., Norrie, M.C., Pastor, Ó. (eds.) CAiSE 2013. LNCS, vol. 7908, pp. 433–448. Springer, Heidelberg (2013)
17. Maggi, F.M., Mooij, A.J., van der Aalst, W.M.P.: User-guided discovery of declarative process models. In: Proc. of CIDM, pp. 192–199. IEEE (2011)
18. Manku, G.S., Motwani, R.: Approximate Frequency Counts over Data Streams. In: VLDB, pp. 346–357 (2002)
19. Pichler, P., Weber, B., Zugal, S., Pinggera, J., Mendling, J., Reijers, H.A.: Imperative versus declarative process modeling languages: An empirical investigation. In: Daniel, F., Barkaoui, K., Dustdar, S. (eds.) BPM Workshops 2011, Part I. LNBIP, vol. 99, pp. 383–394. Springer, Heidelberg (2012)
20. Rozinat, A., Alves de Medeiros, A.K., Günther, C.W., Weijters, A.J.M.M., van der Aalst, W.M.P.: The need for a process mining evaluation framework in research and practice: position paper. In: ter Hofstede, A.H.M., Benatallah, B., Paik, H.-Y. (eds.) BPM 2007 Workshops. LNCS, vol. 4928, pp. 84–89. Springer, Heidelberg (2008)
21. Schweikardt, N.: Short-Entry on One-Pass Algorithms. In: Encyclopedia of Database Systems, pp. 1948–1949 (2009)
22. Smirnov, S., Weidlich, M., Mendling, J., Weske, M.: Action patterns in business process models. In: Baresi, L., Chi, C.-H., Suzuki, J. (eds.) ICSOC-ServiceWave 2009. LNCS, vol. 5900, pp. 115–129. Springer, Heidelberg (2009)
23. Smirnov, S., Weidlich, M., Mendling, J., Weske, M.: Action patterns in business process model repositories. Computers in Industry 63(2), 98–111 (2012)
24. Steeman, W.: Bpi challenge 2013, incidents (2013)
25. van der Aalst, W.M.P.: Process Mining: Discovery, Conformance and Enhancement of Business Processes. Springer (2011)
26. van der Aalst, W.M.P., Pesic, M., Schonenberg, H.: Declarative Workflows: Balancing Between Flexibility and Support. Computer Science - R&D, 99–113 (2009)
27. van Dongen, B.F.: Bpi challenge 2012 (2012)
28. Daelemans, W., Goethals, B., Morik, K. (eds.): ECML PKDD 2008, Part I. LNAI (LNAI), vol. 5211, pp. 672–687. Springer, Heidelberg (2008)
29. Weidlich, M., Ziekow, H., Mendling, J., Günther, O., Weske, M., Desai, N.: Event-based monitoring of process execution violations. In: Rinderle-Ma, S., Toumani, F., Wolf, K. (eds.) BPM 2011. LNCS, vol. 6896, pp. 182–198. Springer, Heidelberg (2011)
30. Widmer, G., Kubat, M.: Learning in the Presence of Concept Drift and Hidden Contexts. Machine Learning 23(1), 69–101 (1996)

Determining the Link and Rate Popularity of Enterprise Process Information*

Bernd Michelberger[1], Bela Mutschler[1], Markus Hipp[2], and Manfred Reichert[3]

[1] University of Applied Sciences Ravensburg-Weingarten, Germany
{bernd.michelberger,bela.mutschler}@hs-weingarten.de
[2] Group Research & Advanced Engineering, Daimler AG, Germany
markus.hipp@daimler.com
[3] Institute of Databases and Information Systems, University of Ulm, Germany
manfred.reichert@uni-ulm.de

Abstract. Today's knowledge workers are confronted with a high load of heterogeneous information making it difficult for them to identify the information relevant for performing their tasks. Particularly challenging is thereby the alignment of process-related information (*process information* for short), such as e-mails, office files, forms, checklists, guidelines, and best practices, with business processes. In previous work, we introduced the concept of *process-oriented information logistics* (POIL) to bridge this gap. POIL allows for the process-oriented and context-aware delivery of relevant process information to knowledge workers. So far, we have introduced concepts to integrate business processes with process information. A remaining challenge is to identify the process information relevant for a given process context. This paper tackles this challenge and extends our POIL approach with techniques and algorithms for identifying relevant process information. More specifically, we introduce two algorithms for determining the relevance of process information based on their link and rate popularity. We use a scenario from the automotive domain to demonstrate and validate the applicability of our approach.

Keywords: process-oriented information logistics, process information relevance, link popularity algorithm, rate popularity algorithm.

1 Introduction

Today's knowledge workers are confronted with a continuously increasing amount of heterogeneous information in their day-to-day operations [1]. Examples include e-mails, office files, process descriptions, forms, checklists, guidelines, working instructions, and best practices. This information may be accessed, for example, through shared drives, databases, portals, or enterprise information systems. Particularly, knowledge workers are not only interested in quickly

* This paper was done in the niPRO research project. The project is funded by the German Federal Ministry of Education and Research (BMBF) under grant number 17102X10. More information can be found at `http://www.nipro-project.org`.

accessing this information, but also require comprehensive and aggregated information when performing a certain task [2,3]. Identifying information required in this context, however, is much more time-consuming and complex than just managing information [4]. Problems frequently encountered include, for example, incomplete, incorrect, unpunctual, or outdated information [5].

A particular challenge is to align process-related information (*process information* for short) with business processes. In practice, process information is not only stored in large, distributed, and heterogeneous data sources, but usually managed separately from business processes. Shared drives, databases, portals, and enterprise information systems are used to manage process information. In turn, business processes are managed using process management technology. Hence, in practice, process information and business processes are often manually linked, i.e., process information is hard-wired to business processes, e.g., in Intranet portals linking specific process information with process tasks. However, this approach often fails due to high maintenance efforts and missing support for the specific requirements of individual process participants.

To tackle this challenge, in previous work, we introduced the concept of *process-oriented information logistics* (POIL) as new paradigm for delivering the right process information, in the right format and quality, at the right place, at the right point in time, and to the right people [6,7]. Specifically, POIL shall enable a process-oriented and context-aware (i.e., personalized) delivery of relevant process information to knowledge workers. Goal is to no longer manually hard-wire process information to business processes, but to identify and deliver relevant process information to knowledge workers automatically.

This paper extends our POIL approach and introduces techniques for determining the relevance of process information based on two algorithms. The first one determines the *link popularity* of process information based on their relationships. The second one determines the *rate popularity* of process information based on user ratings. Section 2 sketches POIL. Section 3 provides formal definitions required for describing the algorithms. Section 4 introduces the algorithms in detail. Section 5 presents a scenario and a survey verifying the applicability of our algorithms. Section 6 discusses related work and Section 7 concludes the paper with a summary and outlook.

2 Process-Oriented Information Logistics

Traditional *information logistics* (IL) approaches deal with the question of how to deliver information to knowledge workers as effectively and efficiently as possible [8]. For this purpose, basic principles from the fields of material logistics and lean management are applied. Examples include just-in-time delivery [9] and satisfaction of customer needs [10]. Particularly, IL aims at delivering that information to knowledge workers fitting their demands best. Thus, *information awareness* (e.g., awareness of information quality and flows) and, to a smaller extent, *context awareness* (e.g., awareness of the user context for the delivery of personalized information) adopt a key role in IL [11] (cf. Fig. 1).

Although IL is independent from the use of information and communication technology (ICT), the latter has been intensively used as IL enabler for several years. Consider ICT solutions in areas like business intelligence, management information systems, and enterprise content management. However, these solutions also suffer from shortcomings like limited applicability (e.g., only applicable within enterprises and not between them) [12], missing operational functionality (e.g., only the management level is addressed) [13], and lack of *process awareness* (e.g., delivering information without considering the current process context).

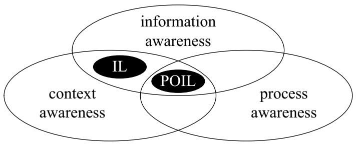


Fig. 1. Problem dimensions of IL and POIL

Missing process awareness in contemporary IL solutions has guided our development of *process-oriented information logistics* (POIL) [6]. POIL aligns process information with business processes, both at the process schema and process instance level [14]. It enables a process-oriented and context-aware delivery of process information to knowledge workers. Thereby, POIL not only combines *information* and *context awareness*, but takes *process awareness* into account as well (cf. Fig. 1), i.e., awareness of process schemas and corresponding instances. Note that POIL focuses on knowledge-intensive business processes involving large amounts of process information, expertise, user interaction, creativity, and decision-making [15] such as the engineering of cars or the medical treatment of patients in hospitals.

The core component of any POIL is a *semantic information network* (SIN), which comprises unified *information objects* (e.g., e-mails, guidelines, best practices), *process objects* (e.g., tasks, pools, lanes, data objects, events, task instances), and the *relationships* (e.g., "is similar to", "has same author as") between them. In particular, a SIN allows identifying objects linked to each other in the one or other way, e.g., information objects addressing the same topic or needed when performing a particular process task. In order to create a SIN, business processes and process information are transformed into unified process and information objects (cf. Fig. 2a-b). In the second step, these objects are semantically analyzed to detect their relationships (cf. Fig. 2c) [7,16].

More precisely, the SIN is created in six consecutive phases (see [6] for details). Our main idea is to split up business processes into their constituent *process objects* and to integrate the latter with *information objects* in the SIN. For creating and maintaining a SIN, we apply algorithms provided by a semantic middleware we use to implement the SIN (see [6] for details). These algorithms, however, do not allow identifying relevant, i.e., currently needed, information objects within a SIN. What we additionally need are further algorithms. This is

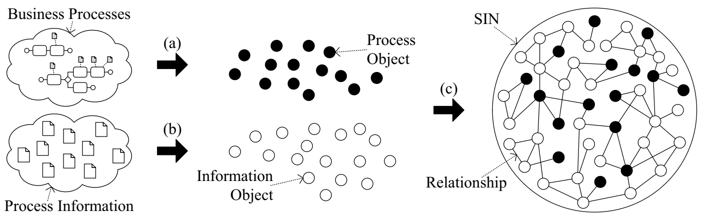


Fig. 2. Schematic and simplified creation of a SIN

indispensable in order to reach the aforementioned goals of POIL, i.e., to provide knowledge workers with the right process information.

Generally, the SIN's relationships may exist between information objects (e.g., a guideline similar to another one), between process objects (e.g., an event triggering a subprocess), and between information and process objects (e.g., an instruction required for executing a task) (cf. Fig. 3a-c). Further, a relationship can be either *explicit* (i.e., hard-wired) or *implicit* (i.e., not hard-wired). Explicit relationships are, for example, modeled data flows in a process schema. Implicit relationships, in turn, are automatically identified by a variety of algorithms and link, for example, objects addressing the same topic or objects used in the same working context [6]. Moreover, relationships are labeled (e.g., "is a template") and weighted. A weight is expressed in terms of a number ranging from 0 to 1 (with 1 indicating the strongest possible relationship) [16]. This allows determining why objects are interlinked and how strong their relationship is.

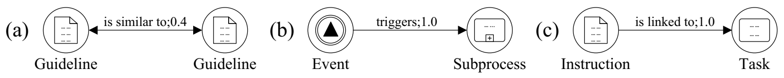


Fig. 3. Relationships between objects

3 Preliminaries

Generally, a SIN is a labeled and weighted *directed graph*. Each directed *edge* $e = (u, v)$ represents a relationship and is associated with an ordered pair of *vertices* (u, v), which represents information and process objects; u is the source and v is the destination of e. Based on this, a SIN is formally defined as follows:

Definition 1 (SIN). *A labeled and weighted digraph is called semantic information network $SIN = (V, E, L, W, f_l, f_w)$, iff:*

- *V is a set of vertices representing information and process objects*
- *E is a set of edges representing relationships between objects*

- *L is a set of labels indicating relationship reasons*
- *W is a set of weights representing the relevance of relationships*
- *f_l is a labeling function with $f_l : E \rightarrow L$ assigning to each edge $e \in E(SIN)$ a label $f_l(e) \in L$*
- *f_w is a weighting function with $f_w : E \rightarrow W$ assigning to each edge $e \in E(SIN)$ a weight $f_w(e) \in W = [0, 1]$*

The membership of V and E in SIN is denoted as $V(SIN)$ and $E(SIN)$. A SIN constitutes a *finite graph*, i.e., V and E are finite sets [17]. A SIN may contain *slings* (i.e., $\exists\ e = (v, v)$, cf. Fig. 4a), *parallelism* (i.e., $\exists\ e = (u, v) \land f = (u, v)$, cf. Fig. 4b), and *anti-parallelism* (i.e., $\exists\ e = (u, v) \land f = (v, u)$, cf. Fig. 4c).

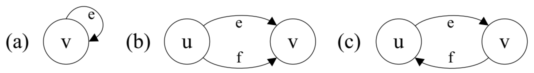


Fig. 4. Slings, parallelism and anti-parallelism of SINs

In general, each vertex v may have several incoming and outgoing edges. The number of incoming edges of a vertex constitutes its *incoming degree*, whereas the number of outgoing edges is denoted as *outgoing degree*. The *total degree* of a vertex corresponds to the sum of its incoming and outgoing degrees. Vertices having no incoming edges are denoted as *unreferenced*. In turn, vertices without outgoing edges are called *non-referencing*. Finally, vertices being unreferenced as well as non-referencing are *isolated* [18].

Definition 2 (Degree). *The number of incoming and outgoing edges of a vertex $v \in V(SIN)$ is denoted as degree of v, where:*

- *$deg^-(v)$ is the incoming degree of a vertex $v \in V(SIN)$ which is denoted as $deg^-(v) = |E^-(v)| = |\{e = (x, y) \in E \mid y = v\}|$*
- *$deg^+(v)$ is the outgoing degree of a vertex $v \in V(SIN)$ which is denoted as $deg^+(v) = |E^+(v)| = |\{e = (x, y) \in E \mid x = v\}|$*
- *$deg(v)$ is the total degree of a vertex $v \in V(SIN)$ which is denoted as $deg(v) = deg^-(v) + deg^+(v) = |E(v)| = |E^-(v)| + |E^+(v)|$*

Vertices directly relating to a neighbored vertex are called *internal neighborhood*, whereas vertices referenced by another vertex are called *external neighborhood*. Then, the *total neighborhood* corresponds to the union of both internal and external neighborhood.

Definition 3 (Neighborhood). *Referencing and referenced vertices of a vertex $v \in V(SIN)$ are denoted as neighborhood of v, where:*

- *$\Gamma^-(v)$ is the internal neighborhood of a vertex $v \in V(SIN)$ which is denoted as $\Gamma^-(v) = V^-(v) = \{u \in V^-(v)\}$*

- *$\Gamma^+(v)$ is the external neighborhood of a vertex $v \in V(SIN)$ which is denoted as $\Gamma^+(v) = V^+(v) = \{u \in V^+(v)\}$*
- *$\Gamma(v)$ is the total neighborhood of a vertex $v \in V(SIN)$ which is denoted as $\Gamma(v) = \Gamma^-(v) \cup \Gamma^+(v) = V(v) = \{u \in V(v)\}$*

As set out in Definition 1, the function f_w assigns a weight to each edge e. This weight indicates the relevance of an edge and therewith the strength of the relationship between two vertices. However, in a SIN, there may be multiple edges between vertices with different weights. In order to determine the overall strength between two vertices, we calculate the average weight of all edges between them. The *average weight$_\varnothing$* of a set of edges F can be calculated as follows:

$$avg_\varnothing(F) = \sum_{f \in F} \frac{f_w(f)}{|F|} \tag{1}$$

In practice, however, certain edges have to be weighted higher. As an example consider a "is similar to" relationship, which is usually more important than a "has same file extension as" relationship. Therefore, we additionally use a significance function f_s with $f_s : E \to \mathbb{N}_1$ assigning to each edge $e \in E$ a significance value $f_s(e) \in \mathbb{N}_1$. The higher a significance value is, the more important is an edge. The *average weight$_\Delta$* of a set of edges F can be calculated as follows:

$$avg_\Delta(F) = \sum_{f \in F} \frac{f_s(f) * f_w(f)}{\sum_{g \in F} f_s(g)} \tag{2}$$

4 Determining the Relevance of Process Information

In two case studies as well as an online survey [19,20], we already showed that knowledge workers spend considerable efforts to handle process information. One challenging task in this context is to identify relevant process information. In POIL, the SIN constitutes the basis for this task. However, additional techniques are needed to determine relevant process information, i.e., currently needed information objects in a SIN dependent on the process context (cf. Fig. 5).

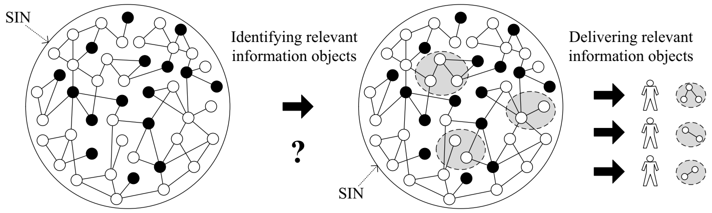

SIN
Identifying relevant information objects
?
SIN
Delivering relevant information objects

Fig. 5. Delivering relevant information objects

In the following, we introduce two algorithms for identifying relevant information objects in a SIN. The first one determines the *link popularity* of information objects based on the SIN's relationship structure. The second one determines the *rate popularity* of information objects based on user ratings. Note that the algorithms can be used independently, but can be combined as well.

4.1 Determining Link Popularity

In enterprises, process information is usually not explicitly linked to other process information or business processes. Therefore, it is not possible to take advantage of a rich relationship structure within an enterprise environment. Instead, process information is implicitly linked to other process information and business processes, e.g., dealing with the same topic or used in the same process context. A SIN makes such implicit relationships explicit by means of its edges. The SIN's relationship structure enables us to apply algorithms to identify strongly linked and therefore popular objects. The problem, however, is that existing link popularity algorithms are not sufficient in our context (as shown in the following). Thus, we extend them and introduce the *SIN LP algorithm*, which allows us determining the link popularity of information objects in a SIN (cf. Fig. 6).

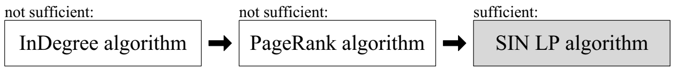


Fig. 6. Link popularity algorithms

Basic to any link popularity algorithm is an *InDegree algorithm* [21] measuring the *link popularity* $LP(v)$ of an information object v by taking its number of incoming edges into account (cf. Formula (3)). The higher the number of incoming edges is, the greater the popularity of an information object becomes:

$$LP(v) = deg^{-}(v) \tag{3}$$

In a SIN, the InDegree is not really helpful since certain relationships might be more valuable than others. This issue, in turn, is picked up by the *PageRank algorithm* [22]: Relationships originating from information objects of high quality are considered being more valuable than relationships originating from information objects of low quality (cf. Formula (4)). Thus, the link popularity $LP(v)$ of an information object v is calculated as follows (with d corresponding to a damping factor ranging from 0 to 1):

$$LP(v) = (1-d) + d \sum_{w \in \Gamma^{-}(v)} \tfrac{LP(w)}{deg^{+}(w)} \tag{4}$$

However, like the InDegree, the conventional PageRank (originally designed for the web) is not applicable to a SIN since it only considers single relationships. In a SIN, there are multiple, weighted, and labeled relationships. Hence, we must extend the PageRank. First, we have to support multiple relationships:

$$LP(v) = (1-d) + d \sum_{w \in \Gamma^-(v)} |\{e = (w,v) \in E\}| * \tfrac{LP(w)}{deg^+(w)} \tag{5}$$

To also support weighted relationships, we further extend Formula (5) and include an average weighting function $avg_{\emptyset}$ (cf. Section 3):

$$LP(v) = (1-d) + d \sum_{w \in \Gamma^-(v)} avg_{\emptyset}(\{e = (w,v) \in E\}) * |\{e = (w,v) \in E\}| * \tfrac{LP(w)}{deg^+(w)} \tag{6}$$

Note that Formula (6) only deals with equally weighted relationships. To finally support differently weighted relationships, we have to extend it by the average weighting function avg_{Δ} (cf. Section 3):

$$LP(v) = (1-d) + d \sum_{w \in \Gamma^-(v)} avg_{\Delta}(\{e = (w,v) \in E\}) * |\{e = (w,v) \in E\}| * \tfrac{LP(w)}{deg^+(w)} \tag{7}$$

Based on Formula (7) it becomes possible to determine the link popularity of information objects in a SIN. Note that this corresponds to the solution of a system of equations. In our approach we use an approximate, iterative calculation of the link popularity, i.e., we assign an initial $LP(v) = init$ to each information object v. The link popularity $LP(v)$ is then iteratively determined for each information object v as follows (let i be the number of iterations)[1]:

```
Input: $SIN = (V, E, L, W, f_l, f_w)$; $d$; $i$; $init$;
Result: $LP(v)$ for each $v \in V(SIN)$;
foreach $v \in V(SIN)$ do $LP(v) = init$ foreach $e \in E(SIN)$ do $f_s(e)$ for $j = 1$
to $i$ do
    foreach $v \in V(SIN)$ do
        $pop = 0$;
        foreach $w \in \Gamma^-(v)$ do
            $pop \overset{+}{=} avg_{\Delta}(\{e = (w,v) \in E\}) *$
                $|\{e = (w,v) \in E\}| * LP(w) \;/\; deg^+(w)$;
        end
        $LP(v) = (1-d) + d * pop$;
    end
    $j = j + 1$;
end
```

Algorithm 1. SIN Link Popularity Algorithm

In summary, algorithm 1 allows determining the link popularity of information objects based on the SIN's relationship structure in an iterative way.

[1] Our implementation can be found at `http://sourceforge.net/projects/linkinganalyzer/`

4.2 Determining Rate Popularity

This section introduces another algorithm that allows determining the rate popularity of process information based on user ratings. In enterprises, existing IL solutions often allow users to rate the quality of process information, e.g., by means of "like buttons" or "five stars ratings". The set of ratings R can then be used to determine the *rate popularity* $RP(v)$ of an information object v. However, ranking information objects based on user ratings is a non-trivial task. Like before, we first show that existing algorithms are not sufficient in POIL and then introduce our *SIN RP algorithm*, which allows us determining the rate popularity of information objects in a SIN (cf. Fig. 7).

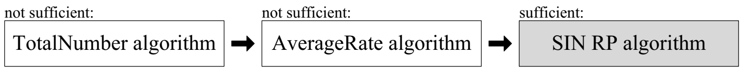


Fig. 7. Rate popularity algorithms

An approach to determine rate popularity $R(v)$ of an information object v is to rank information objects by their *total number* of ratings $|R(v)|$:

$$RP(v) = |R(v)| \tag{8}$$

Another approach is to determine the rate popularity $RP(v)$ based on the *average user rating* using $avg(R(v))$ of an information object v:

$$RP(v) = \sum_{r \in R(v)} \frac{r}{|R(v)|} \tag{9}$$

However, applying Formulas (8) or (9) is not appropriate in a SIN. Both formulas tend to prefer older information objects available for a longer time (i.e., there was more time for users to rate for these information objects). This shortcoming is rather problematic in enterprise environments with continuously emerging information objects. Using Formula (9) results in another problem: Assume that in a "five stars rating" there is an information object with an overall weight of 4.8, which is based on hundreds of individual ratings. Additionally assume that another information object is rated by one knowledge worker with 5.0. The latter information object is then directly ranked on the first position. To avoid this, all ratings must be taken into account.

Thus, we calculate the rate popularity consistent with *Bayesian interpretation* [23]. Formula (10) allows calculating the average rating $avg(R)$ of all information objects. Formula (11) then calculates the rate popularity $RP(v)$ of a single information object v taking both the set of ratings R and the information objects' age into account. Thus, we avoid that information objects with few, but favorable ratings are ranked on the first positions:

$$avg(R) = \sum_{v \in V} \frac{|R(v)| * avg(R(v))}{|R|} \tag{10}$$

$$RP(v) = \frac{\frac{\left(\frac{|R|}{|\{v \in V \mid R(v)>0\}|} * avg(R)\right) + \left(|R(v)| * avg(R(v))\right)}{\frac{|R|}{|\{v \in V \mid R(v)>0\}|} + |R(v)|}}{age(v)} \tag{11}$$

Algorithm 2 shows how the rate popularity value for each information object v is calculated taking the set of available user ratings R into account[2]:

```
Input: $SIN = (V, E, L, W, f_l, f_w)$; $R$;
Result: $RP(v)$ for each $v \in V(SIN)$ where $|R(v)| > 0$;
foreach $v \in V(SIN)$ do
    if $|R(v)| > 0$ then
        $avg(R) \overset{+}{=} |R(v)| * avg(R(v))\ /\ |R|$;
    end
end
foreach $v \in V(SIN)$ do
    if $|R(v)| > 0$ then
        $pop = ((|R|\ /\ |\{v \in V \mid R(v) > 0\}| *$
            $avg(R)) + (|R(v)| * avg(R(v))))$;
        $pop = pop\ /\ (|R|\ /\ |\{v \in V \mid R(v) > 0\}| +$
            $|R(v)|)$;
        $RP(v) = pop\ /\ age(v)$;
    end
end
```

Algorithm 2. SIN Rate Popularity Algorithm

In summary, algorithm 2 allows determining the rate popularity of information objects based on user ratings in an easy way.

5 Validation

In order to prove that our algorithms support knowledge workers when performing knowledge-intensive tasks, we use a real-world scenario from the automotive domain (cf. Section 5.1). Specifically, we implement our algorithms (cf. Section 5.2) and then compare their outcome with results of a survey among automotive engineers who were asked to manually determine the relevance of process information related to the considered scenario (cf. Section 5.3). Doing so, we aim to show that our algorithmic results can indeed replace the costly and time-intensive human determination of relevant process information.

5.1 Real-World Scenario

Our scenario (cf. Fig. 8) deals with the review of product requirements documented in functional specifications at a large automotive manufacturer [19].

[2] Our implementation can be found at
`http://sourceforge.net/projects/ratinganalyzer/`

Goal is to improve as well as to approve such specifications. The underlying review process is knowledge-intensive, i.e., it comprises large amounts of process information (e.g., review protocols, checklists, review templates, guidelines), user interaction (e.g., "perform review meeting", "send review comments"), and decision-making (e.g., should the document be approved or not?). Three roles are involved: (1) The *author* provides the specification to be reviewed. (2) The *review moderator* organizes the review meetings. (3) The *reviewer* finally analyzes the provided specification and documents errors, ambiguities, and uncertainties.

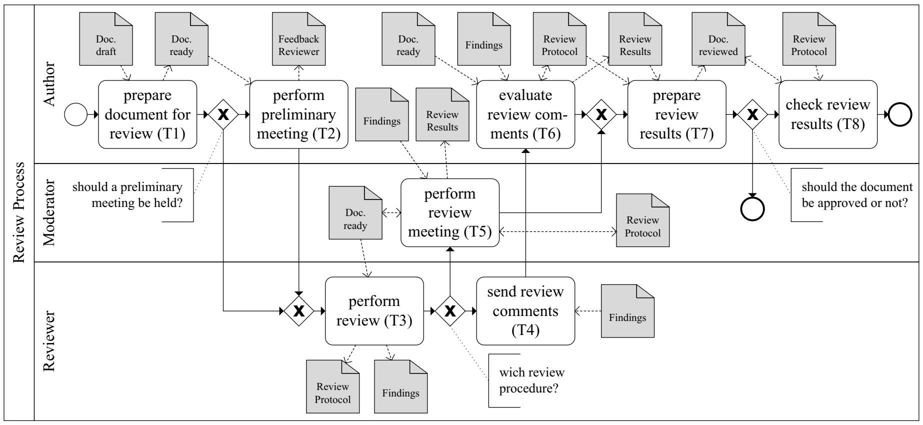
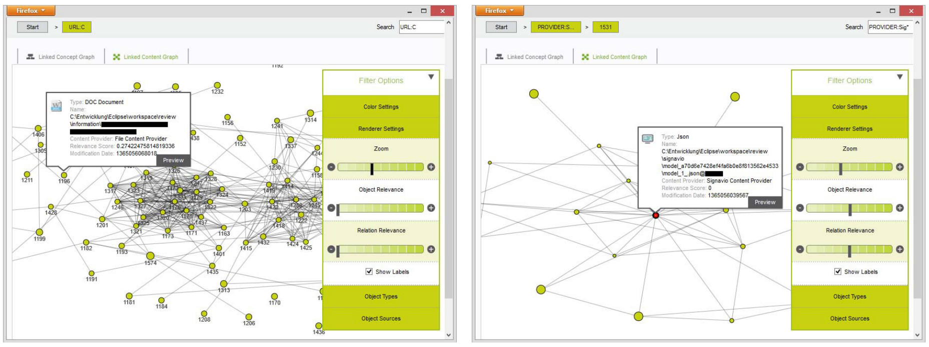

Review Process
Author
Moderator
Reviewer
Doc. draft
Doc. ready
Feedback Reviewer
Doc. ready
Findings
Review Protocol
Review Results
Doc. reviewed
Review Protocol
prepare document for review (T1)
perform preliminary meeting (T2)
evaluate review comments (T6)
prepare review results (T7)
check review results (T8)
Findings
Review Results
should a preliminary meeting be held?
Doc. ready
perform review meeting (T5)
Review Protocol
should the document be approved or not?
perform review (T3)
send review comments (T4)
Findings
Review Protocol
Findings
wich review procedure?

Fig. 8. Process schema of our automotive scenario (BPMN model)

The review process starts with the preparation of the document to be reviewed (task T1). This step is performed by the document's author. Based on this initial preparation, the author decides whether or not a preliminary review meeting becomes necessary (task T2). Afterwards, the document is reviewed (task T3). Based on the review's outcome, the reviewer decides whether an additional review meeting is needed (task T4) or whether it is sufficient to directly send findings and comments to the author (task T5). The latter then evaluates review results (task T6) and updates the document accordingly (task T7). If the document's overall review status is rejected, it will not be approved. In turn, if its overall review status is accepted, the author can finally approve the document (task T8). For each of these process steps, a variety of process information is needed; e.g., guidelines, templates, meeting protocols, or working instructions.

5.2 Implementation

Based on the scenario discussed we first implemented the corresponding SIN - altogether comprising one process schema modeled with Signavio Process Editor, three process instances created and managed with the Activiti BPM Platform, and about 300 documents (i.e., process information) such as reviews, review protocols, templates, guidelines etc. For creating the SIN we use the semantic middleware iQser GIN Server as well as several Java open-source plugins we developed[3]. The implemented SIN includes 348 objects (45 process objects, 303 information objects) and 65.991 relationships (77 process object relationships, 65.319 information object relationships, and 595 cross-object-relationships) [6]. While Fig. 9a shows the entire SIN of our scenario, Fig. 9b only depicts objects (i.e., information and process objects) directly related to task T3. Note that due to privacy reasons, the document names are blacked out.



Type: DOC Document
Name:
C:\Entwicklung\Eclipse\workspace\review
\information\[redacted]
Content Provider: File Content Provider
Relevance Score: 0.27422475814819336
Modification Date: 1365056068018

Type: Json
Name:
C:\Entwicklung\Eclipse\workspace\review
\signavio
\model_a70d6e7428ef4fa6b0e8f813562e4533
\model_1_.json@[redacted]
Content Provider: Signavio Content Provider
Relevance Score: 0
Modification Date: 1365056039567

(a) Entire SIN of our scenario. (b) Task T3 and related objects.

Fig. 9. The implemented SIN

We then additionally implemented our algorithms in a proof-of-concept prototype called iGraph[4], a web-based Java/Scala application. iGraph uses the web application framework Play, the Twitter Bootstrap framework, the JavaScript libraries Data-Driven Documents (D3) and jQuery, HyperText Markup Language (HTML) 5 templates, and Cascading Style Sheets (CSS) 3.

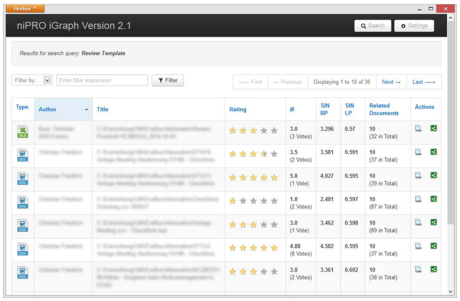
The iGraph user interface provides two views: a *table-based* and a *graph-based view*. The former lists information objects identified based on a document search query (cf. Fig. 10a). The latter illustrates the relationships of selected SIN objects (i.e., process or information objects); Fig. 10b, for example, depicts information objects linked to process task T3 of our scenario.

5.3 Empirical Validation

Using iGraph, we construct a survey (cf. Section 5.3). In this survey, automotive engineers evaluate previously calculated results of the link and rate popularity algorithms. Doing so, we aim to show that our algorithmic results can indeed replace the costly human determination of relevant process information.

[3] These plugins are available at `http://sourceforge.net/directory/?q=nipro`

[4] A screencast presenting the iGraph prototype is available at `http://nipro.hs-weingarten.de/screencast`

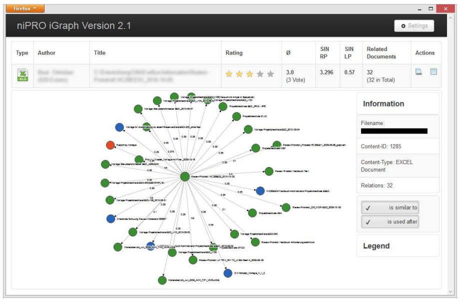


Results for search query: ***Review Template***

Displaying 1 to 10 of 36

Type	Author	Title	Rating	Ø	SIN RP	SIN LP	Related Documents	Actions
XLS	[illegible]	[illegible]	★★★☆☆	3.0 (3 Votes)	3.296	0.57	10 (32 in Total)	
DOC	[illegible]	[illegible]	★★★½☆	3.5 (2 Votes)	3.581	0.591	10 (37 in Total)	
DOC	[illegible]	[illegible]	★★★★★	5.0 (1 Vote)	4.027	0.595	10 (39 in Total)	
DOC	[illegible]	[illegible]	★☆☆☆☆	1.0 (2 Votes)	2.481	0.597	10 (87 in Total)	
DOC	[illegible]	[illegible]	★★★☆☆	3.0 (1 Vote)	3.462	0.598	10 (89 in Total)	
DOC	[illegible]	[illegible]	★★★★★	4.88 (8 Votes)	4.582	0.595	10 (37 in Total)	
DOC	[illegible]	[illegible]	★★★☆☆	3.0 (2 Votes)	3.361	0.602	10 (38 in Total)	

(a) Table-based view.



Type	Author	Title	Rating	Ø	SIN RP	SIN LP	Related Documents	Actions
XLS	[illegible]	[illegible]	★★★☆☆	3.0 (3 Vote)	3.296	0.57	32 (32 in Total)	

Filename: [redacted]

Content-ID: 1285

Content-Type: EXCEL Document

Relations: 32

✓ is similar to

✓ is used after

(b) Graph-based view.

Fig. 10. Different views of iGraph

More specifically, the goal is to prove the accuracy of our algorithms. Particularly, the survey was guided by two research questions: (RQ1) ”How do results of the SIN LP algorithm match with user-generated evaluations on the relevance of process information?” and (RQ2) ”How good is the ranking of process information based on our SIN RP algorithm compared to other ranking approaches?”

We performed the survey in late April 2013. The questionnaire comprised 18 questions. Overall, 20 automotive experts participated. Most of them work at electric/electronic engineering departments, but there were also participants from other departments. All participants were selected due to their expert knowledge regarding the considered review scenario.

RQ1 (Investigating Link Popularity). To investigate RQ1, we use iGraph to calculate two link popularity result lists (as input values we set $init = 0.45$, $i = 12$, $d = 0.5$, and double-weight ”is similar to”-relationships): (a) the top eight documents according to the SIN LP algorithm for process task T1 and (b) the top eight documents according to the SIN LP algorithm for process task T3. Table 1 shows the documents the SIN LP algorithm returns for T1 and T3.

We then asked survey participants to evaluate - based on their practical experiences - the relevance of the documents returned by the SIN LP algorithm for the tasks T1 (”prepare document for review”) and T3 (”perform review”). As can be seen in Table 1, the survey participants confirm the relevance for the majority of the 16 documents identified by our SIN LP algorithm. Note that we consider a document as being relevant if more than 50% of the survey participants confirm relevance. Results show that our algorithm is indeed well working, especially since the algorithm's overall accuracy can be further improved, for example, by combining it with other algorithms (e.g., the SIN RP algorithm).

RQ2 (Investigating Rate Popularity). To investigate RQ2, we first calculate a ranking of review templates applying the SIN RP algorithm (note that we use real ratings we obtained from the automotive manufacturer supporting the survey). Fig. 11 shows the calculated ranking of review templates. Additionally,

Table 1. SIN LP algorithm validation results

Case	ID	Type	$LP(v)$	#Marked	Ratio	Is Relevant?
Task T1	1231	Review Template	0.443	12	60.0 %	■
	1210	Process Overview	0.442	20	100.0 %	■
	439	Review Template	0.441	4	20.0 %	□
	432	Specific Review	0.439	17	85.0 %	■
	811	Guideline	0.435	4	20.0 %	□
	439	Protocol	0.434	2	10.0 %	□
	578	Checklist	0.434	19	95.0 %	■
	777	Guideline	0.432	19	95.0 %	■
Task T3	1210	Process Overview	0.443	17	85.0 %	■
	879	Protocol	0.442	19	95.0 %	■
	431	Specific Review	0.441	10	50.0 %	□
	432	Specific Review	0.439	9	45.0 %	□
	741	Review Template	0.435	7	35.0 %	□
	439	Review Template	0.434	6	30.0 %	□
	578	Checklist	0.434	18	90.0 %	■
	729	Review Template	0.432	19	95.0 %	■

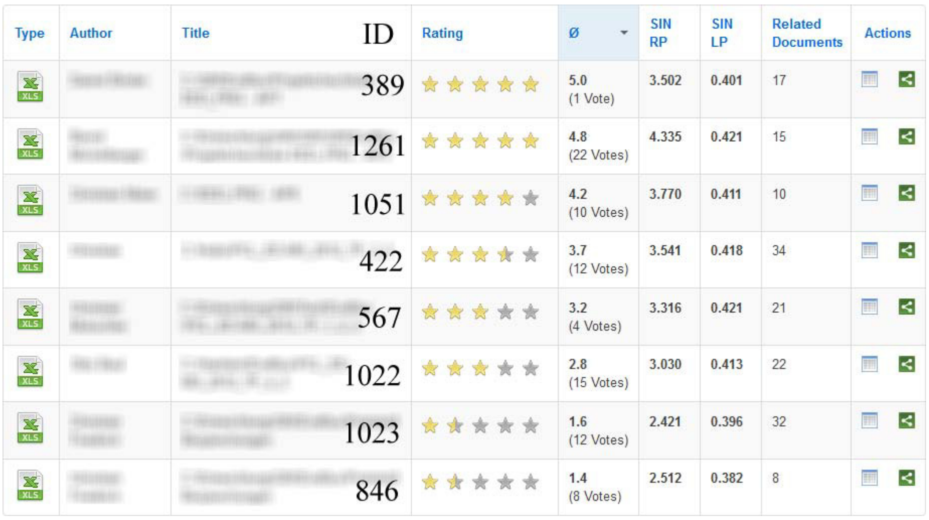
□ = no ■ = yes

in order to evaluate the SIN RP rating, we calculate three further rate-based rankings. More specifically, we calculate the additional rankings based on Formula (8) (a ranking based on the total number of ratings) and Formula (9) (a ranking based on the average rating). Finally, we also create a random ranking.

We then asked survey participants to evaluate - based on their practical experiences - both the plausibility and the usefulness of the four rankings. Fig. 12 shows that 16 out of 20 participants consider the ranking created with our SIN RP algorithm as the most plausible one. The ranking based on the total number of ratings is considered as the second most plausible one (three votes). The ranking based on the average rating only received one vote.

As aforementioned, we also asked the participants to evaluate the usefulness of the rankings based on a Likert Scale [24] ranging from "not at all useful" (1) to "very useful" (5)). Fig. 12 shows that 87.5% of the participants state that the ranking created with our SIN RP algorithm is "useful" or "very useful". Again, survey results show that our algorithm is indeed well working.

Conclusion. Our empirical validation confirms that most of the documents returned by our SIN LP algorithm are indeed relevant ones. Moreover, our empirical research also shows that the link popularity is a good indicator for identifying relevant process information, especially since results of the SIN LP algorithm can be further refined for specific process tasks by applying the SIN LP algorithm to only specific parts of a SIN (e.g., to a specific process task, corresponding task instances, or related information objects).



Type	Author	Title ID	Rating	Ø	SIN RP	SIN LP	Related Documents	Actions
XLS	[redacted]	[redacted] 389	★★★★★	5.0 (1 Vote)	3.502	0.401	17	
XLS	[redacted]	[redacted] 1261	★★★★★	4.8 (22 Votes)	4.335	0.421	15	
XLS	[redacted]	[redacted] 1051	★★★★☆	4.2 (10 Votes)	3.770	0.411	10	
XLS	[redacted]	[redacted] 422	★★★★☆	3.7 (12 Votes)	3.541	0.418	34	
XLS	[redacted]	[redacted] 567	★★★☆☆	3.2 (4 Votes)	3.316	0.421	21	
XLS	[redacted]	[redacted] 1022	★★★☆☆	2.8 (15 Votes)	3.030	0.413	22	
XLS	[redacted]	[redacted] 1023	★★☆☆☆	1.6 (12 Votes)	2.421	0.396	32	
XLS	[redacted]	[redacted] 846	★★☆☆☆	1.4 (8 Votes)	2.512	0.382	8	

Fig. 11. Rating

The results of the SIN RP algorithm are considered as useful by the survey participants. In fact, most participants state that the ranking of documents as suggested by the SIN RP algorithm is both plausible and useful. Additionally, our SIN RP algorithm avoids the problematic situation that process information with only a few good user ratings is directly ranked on the first position of a ranking. Finally note that the results of the SIN RP algorithm can be easily further improved, for example, by taking into account the expertise of knowledge workers, i.e., ratings of experienced knowledge workers might be weighted higher.

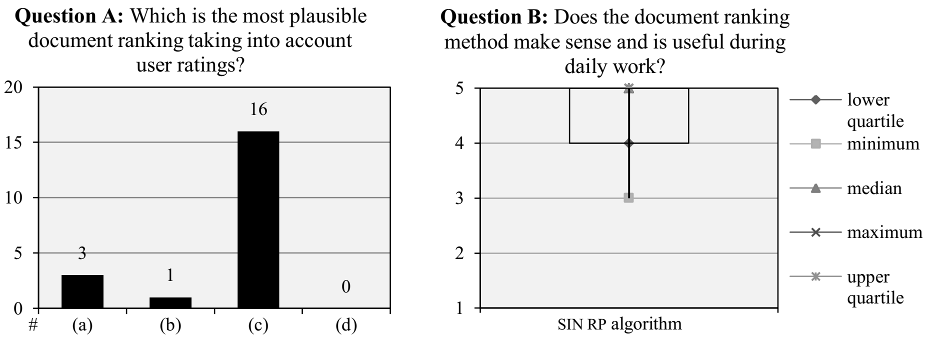


Fig. 12. SIN RP algorithm validation results

In summary, the popularity values of our algorithms clearly help to determine the relevance of process information. However, as it is difficult to determine the overall relevance of process information based on a single algorithm, we will combine our algorithms when further extending our POIL framework.

6 Related Work

As discussed in Section 2, various ICT solutions have been proposed to enable IL and hence to identify relevant information. As examples consider data warehousing (DWH), business intelligence (BI) solutions, decision support systems (DSS), and enterprise content management (ECM). However, these approaches suffer from several weaknesses. For example, DWH rather focuses on the creation of an integrated database [25]. Traditional BI, in turn, addresses data analytics and is usually isolated from business process execution [26]. Conventional DSS support complex business decision-making at the management level [27]. By contrast, ECM deals with the management of information across enterprises referring to related strategies, methods, and tools [28].

There exists a wide range of link popularity algorithms. Best known is the PageRank algorithm [22]. However, relationships being more valuable than others are picked up by other algorithms as well, e.g., the Hits algorithm [29] or the weighted PageRank algorithm [30]. An algorithm combining both PageRank and Hits is the Salsa algorithm [31]. Another evolution of the PageRank is the Topic-Sensitive PageRank algorithm [32], which additionally considers topics. However, all these algorithms have been originally developed for the web and cannot be directly, i.e., without modification, applied to POIL. Particularly, they do not allow dealing with the specific characteristics of a SIN.

Research done by others also influenced the development of our rating popularity algorithm. An approach to improve search results based on user ratings, for example, is presented in [33]. In [34], a study on rate popularity algorithms and their pros and cons is presented. Similar to our algorithm, a self-learning algorithm is presented in [35], which addresses both user ratings and content relevance. Notwithstanding, like the link popularity algorithms, existing rate popularity algorithms cannot be directly applied to a SIN.

7 Summary and Outlook

This paper presented two algorithms for determining the relevance of process information in POIL. The first one determines the popularity of process information based on the relationships of a SIN. The second one determines the popularity of process information based on user ratings. We applied our algorithms to a real-world scenario, i.e., validated them based on an implementation and a survey in the automotive domain.

In future, we will develop additional algorithms for determining the relevance of process information. In particular, we will focus on self-learning algorithms enabling us to take into account our POIL context framework [36].

References

1. Öhgren, A., Sandkuhl, K.: Information Overload in Industrial Enterprises - Results of an Empirical Investigation. In: Proc. 2nd European Conf. on Information Management and Evaluation, London, pp. 343–350 (2008)

2. Mundbrod, N., Kolb, J., Reichert, M.: Towards a System Support of Collaborative Knowledge Work. In: Proc. 1st Int'l Workshop on Adaptive Case Management (ACM 2012), Tallinn, pp. 31–42 (2012)
3. Bocij, P., Chaffey, D., Greasley, A., Hickie, S.: Business Information Systems: Technology, Development and Management for the E-Business. Prentice Hall (2006)
4. Rowley, J.: The wisdom hierarchy: representations of the DIKW hierarchy. J. of Information Science 33(2), 163–180 (2006)
5. Michelberger, B., Mutschler, B., Reichert, M.: Towards Process-oriented Information Logistics: Why Quality Dimensions of Process Information Matter. In: Proc. 4th Int'l Workshop on Enterprise Modelling and Information Systems Architectures (EMISA 2011), Hamburg. LNI, vol. 190, pp. 107–120 (2011)
6. Michelberger, B., Mutschler, B., Reichert, M.: Process-oriented Information Logistics: Aligning Enterprise Information with Business Processes. In: Proc. 16th IEEE Int'l EDOC Conf. (EDOC 2012), Beijing, pp. 21–30 (2012)
7. Hipp, M., Michelberger, B., Mutschler, B., Reichert, M.: A Framework for the Intelligent Delivery and User-adequate Visualization of Process Information. In: Proc. 28th Symp. on Applied Computing (SAC 2013), Coimbra, pp. 1383–1390 (2013)
8. Heuwinkel, K., Deiters, W.: Information logistics, E-Healthcare and Trust. In: Proc. Int'l Conf. e-Society (IADIS 2003), Lisbon, vol. 2, pp. 791–794 (2003)
9. Deiters, W., Löffeler, T., Pfennigschmidt, S.: The Information Logistics Approach Toward User Demand-driven Information Supply. In: Proc. Conf. on Cross-Media Service Delivery (CMSD 2003), Santorini, pp. 37–48 (2003)
10. Womack, J.P., Jones, D.T.: Lean Thinking: Banish Waste and Create Wealth in Your Corporation. Free Press (2003)
11. Michelberger, B., Andris, R., Girit, H., Mutschler, B.: A Literature Survey on Information Logistics. In: Abramowicz, W. (ed.) BIS 2013. LNBIP, vol. 157, pp. 138–150. Springer, Heidelberg (2013)
12. Dinter, B., Winter, R.: Information Logistics Strategy - Analysis of Current Practices and Proposal of a Framework. In: Proc. 42nd Hawaii Int'l Conf. on System Sciences (HICSS-42), Hawaii, pp. 1–10 (2009)
13. Winter, R.: Enterprise-wide Information Logistics: Conceptual Foundations, Technology Enablers, and Management Challenges. In: Proc. 30th Int'l Conf. on Information Technology Interfaces (ITI 2008), Dubrovnik, pp. 41–50 (2008)
14. Rinderle, S., Reichert, M., Dadam, P.: On Dealing with Structural Conflicts between Process Type and Instance Changes. In: Desel, J., Pernici, B., Weske, M. (eds.) BPM 2004. LNCS, vol. 3080, pp. 274–289. Springer, Heidelberg (2004)
15. Gronau, N., Müller, C., Korf, R.: KMDL - Capturing, Analysing and Improving Knowledge-Intensive Business Processes. J. of Universal Computer Science (JUCS) 11(4), 452–472 (2005)
16. Wurzer, J., Mutschler, B.: Bringing Innovative Semantic Technology to Practice: The iQser Approach and its Use Cases. In: Proc. 4th Int'l Workshop on Applications of Semantic Technologies (AST 2009), Lübeck, pp. 3026–3040 (2009)
17. Diestel, R.: Graph Theory. Springer (2010)
18. Gyöngyi, Z., Garcia-Molina, H., Pedersen, J.: Combating Web Spam with TrustRank. In: Proc. 13th Int'l Conf. on Very Large Data Bases (VLDB 2004), Toronto, vol. 30, pp. 576–587 (2004)
19. Michelberger, B., Mutschler, B., Reichert, M.: On Handling Process Information: Results from Case Studies and a Survey. In: Daniel, F., Barkaoui, K., Dustdar, S. (eds.) BPM Workshops 2011, Part I. LNBIP, vol. 99, pp. 333–344. Springer, Heidelberg (2012)

20. Hipp, M., Mutschler, B., Reichert, M.: On the Context-aware, Personalized Delivery of Process Information: Viewpoints, Problems, and Requirements. In: Proc. 6th Int'l Conf. on Availability, Reliability and Security, ARES 2011, Vienna, pp. 390–397 (2011)
21. Borodin, A., Roberts, G.O., Rosenthal, J.S., Tsaparas, P.: Link Analysis Ranking: Algorithms, Theory, and Experiments. J. of ACM Transactions on Internet Technology (TOIT) 5(1), 231–297 (2005)
22. Page, L., Brin, S., Motwani, R., Winograd, T.: The PageRank Citation Ranking: Bringing Order to the Web. Technical Report, Stanford University (1999)
23. MacKay, D.J.C.: Information Theory, Inference and Learning Algorithms. Cambridge University Press (2003)
24. Likert, R.: A Technique for the Measurement of Attitudes. Archives of Psychology 140, 1–55 (1932)
25. Lechtenbörger, J.: Data warehouse schema design. Infix Akademische Verlagsgesellschaft Aka GmbH, PhD Thesis, University of Münster (2001)
26. Bucher, T., Dinter, B.: Process Orientation of Information Logistics - An Empirical Analysis to Assess Benefits, Design Factors, and Realization Approaches. In: Proc. 41st Hawaii Int'l Conf. on System Sciences (HICSS-41), Hawaii, pp. 392–402 (2008)
27. Janakiraman, V.S., Sarukesi, K.: Decision Support Systems. Prentice-Hall (2004)
28. Cameron, S.A.: Enterprise Content Management: A Business and Technical Guide. British Informatics Society (2011)
29. Kleinberg, J.M., Kumar, R., Raghavan, P., Rajagopalan, S., Tomkins, A.S.: The Web as a Graph: Measurements, Models, and Methods. In: Asano, T., Imai, H., Lee, D.T., Nakano, S.-i., Tokuyama, T. (eds.) COCOON 1999. LNCS, vol. 1627, pp. 1–17. Springer, Heidelberg (1999)
30. Xing, W., Ghorbani, A.: Weighted PageRank Algorithm. In: Proc. of the 2nd Annual Conf. on Communication Networks and Services Research, Fredericton, pp. 305–314 (2004)
31. Lempel, R., Moran, S.: The Stochastic Approach for Link-Structure Analysis (SALSA) and the TKC Effect. In: Proc. 9th Int'l World Wide Web Conf. on Computer Networks, Amsterdam, pp. 387–401 (2000)
32. Haveliwala, T.H.: Topic-Sensitive PageRank: A Context-Sensitive Ranking Algorithm for Web Search. J. of IEEE Transactions on Knowledge and Data Engineering 15(4), 784–796 (2003)
33. Vassilvitskii, S., Brill, E.: Using Web-Graph Distance for Relevance Feedback in Web Search. In: Proc. 29th Int'l ACM SIGIR Conf. on Research and Development in Information Retrieval, Seattle, pp. 147–153 (2006)
34. Lowd, D., Godde, O., McLaughlin, M., Nong, S., Wang, Y., Herlocker, J.L.: Challenges and Solutions for Synthesis of Knowledge Regarding Collaborative Filtering Algorithms. Technical Report, Oregon State University (2004)
35. Bian, J., Liu, Y., Agichtein, E., Zha, H.: A few Bad Votes too Many?: Towards Robust Ranking in Social Media. In: Proc. 4th Int'l Workshop on Adversarial Information Retrieval on the Web, Bejing, pp. 53–60 (2008)
36. Michelberger, B., Mutschler, B., Reichert, M.: A Context Framework for Process-oriented Information Logistics. In: Abramowicz, W., Kriksciuniene, D., Sakalauskas, V. (eds.) BIS 2012. LNBIP, vol. 117, pp. 260–271. Springer, Heidelberg (2012)

Integrated Cloud Application Provisioning: Interconnecting Service-Centric and Script-Centric Management Technologies

Uwe Breitenbücher, Tobias Binz, Oliver Kopp,
Frank Leymann, and Johannes Wettinger

Institute of Architecture of Application Systems, University of Stuttgart, Germany
Universitätsstraße 38, 70569 Stuttgart, Germany
{breitenbuecher,lastname}@iaas.uni-stuttgart.de

Abstract Modern Cloud applications employ a plethora of components and XaaS offerings that need to be configured during provisioning. Due to increased heterogeneity, complexity is growing and existing approaches reach their limits if multiple different provisioning and configuration technologies are involved. They are not able to integrate them in an automated, flexible, and customizable way. Especially combining proprietary management services with script-centric configuration management technologies is currently a major challenge. To enable automated provisioning of such applications, we introduce Generic Lifecycle Management Planlets that provide a means to combine custom provisioning logic with common provisioning tasks. We implemented planlets for provisioning and customization of components and XaaS offerings based on both SOAP and RESTful Web services as well as configuration management technologies such as Chef to show the feasibility of the approach. By using our approach, multiple technologies can be combined seamlessly.

Keywords: Cloud Application Provisioning, Integration, Management Scripts, Management Services.

1 Introduction

With growing adoption of Cloud computing, the automated provisioning of composite Cloud applications becomes a major issue as this is key to enable Cloud properties such as on-demand self-service, pay-as-you-go pricing, and elasticity. However, due to various kinds of different components and XaaS offerings employed in modern composite Cloud applications and the dependencies among them, the complexity and heterogeneity is constantly increasing. This becomes a challenge if the components and XaaS offerings employ different management technologies and need to be combined and customized during provisioning. Especially application-specific provisioning and customization tasks such as wiring custom components and standard XaaS offerings together cannot be implemented in a generic and reusable way. In addition, these tasks are typically implemented by various kinds of different heterogeneous provisioning technologies. Although wiring and configuration of (custom) components are typically implemented using script-based technologies such as Puppet[1], Chef[2], CFEngine[3], or Juju[4], provisioning and configuration of XaaS Cloud offerings such as *Infrastructure as a Service* or *Database as a Service* are typically provided through Web service APIs—mostly HTTP-based Query services, RESTful Web services, or SOAP Web services. As a result, available provisioning approaches reach their limits: in case multiple standard components, custom components, and XaaS offerings provided by different vendors and Cloud providers are combined and different provisioning and configuration technologies are involved, available solutions are unable to integrate them. In this paper, we tackle this issue. We present an approach to enable the seamless integration of script-centric and service-centric provisioning and configuration technologies in order to customize and automate provisioning of composite Cloud applications. Therefore, we extend our concept of Management Planlets [2] by *Generic Lifecycle Management Planlets (GLMPs)* that provide a means to bind abstract lifecycle tasks to script- or service-based operation implementations. This enables the seamless integration of different technologies to customize the generation of an overall provisioning flow that provisions the application fully automated. The extension enables application developers to benefit from reusable common provisioning logic implemented by third parties and individual customization possibilities. We validate the approach by creating several GLMPs that support the integration of service-based technologies such as RESTful and HTTP Query Web services as well as script-based technologies such as Chef. To prove the benefits, we evaluate the concept against existing approaches in terms of functionality and features. In addition, we implemented a prototype to show its practical applicability.

The remainder of this paper is structured as follows. In Section 2, we motivate our approach, introduce a motivating scenario, and describe why the related work is not able to tackle the analyzed issues. Afterwards, we describe Management Planlets and Provisioning Topologies in Section 3. In Section 4, we present our approach to integrate script- and service-centric provisioning technologies. We present a case study in Section 5 and evaluate the approach in Section 6. Finally, Section 7 concludes and provides an outlook on future work.

2 Motivation, Scenario, and Related Work

In this section, we motivate our approach and describe the type of applications whose provisioning is the focus of this paper. Afterwards, we describe the provisioning of a motivating scenario and identify the occurring challenges and problems. The related work, that does not provide a means to tackle these issues completely, is discussed in Section 2.3.

[1] `http://puppetlabs.com/puppet/what-is-puppet`
[2] `http://www.opscode.com/chef`
[3] `http://cfengine.com`
[4] `https://juju.ubuntu.com`

2.1 The Cloud Applications to be Provisioned

This paper considers Cloud applications that are of small and medium size and complexity such as CRM systems. They are based on multiple XaaS offerings possibly of different providers and employ common as well as individual software components. We use a PHP-based Web shop application that stores product data

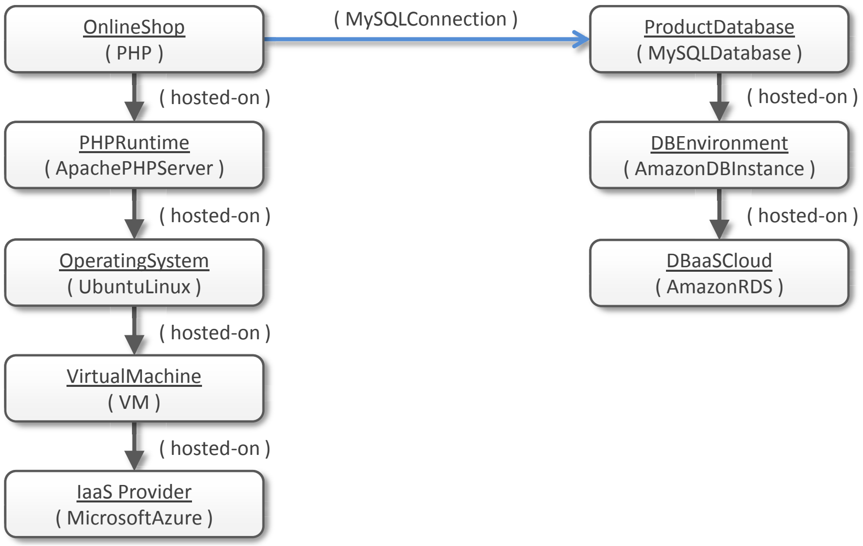


Fig. 1. Motivating Scenario

in a relational database as running example throughout this paper. The application is based on two Cloud offerings of type infrastructure and database as a service: the infrastructure is provided by Microsoft's Windows Azure Cloud offering[5] and the database by Amazon's Relational Database Service (AmazonRDS[6]). Figure 1 shows the application modeled as application topology. A topology is a graph consisting of nodes, which represent the components, and edges, which represent the relations between the components. We refer to nodes and relations as *elements* in the following. Each element has a certain type that defines its semantics and properties, which are key-value pairs. Types may inherit from a super type, e. g., Ubuntu inherits from Linux. We use Vino4TOSCA [1] to render topologies. Thus, types are denoted as text enclosed by parentheses and element ids as underlined text. The application itself consists of two connected stacks. The left stack hosts the business logic implemented in PHP. This is denoted by the node of type PHP on the top left. The PHP node is hosted on a PHP Runtime of type ApachePHPServer which runs on an Ubuntu Linux operating system. This Linux runs in a virtual machine (VM) hosted on Azure.

[5] http://www.windowsazure.com/
[6] http://aws.amazon.com/rds/

The product data are stored in a database node of type MySQLDatabase hosted on AmazonRDS. The connection between these stacks is established by a relation of type MySQLConnection which connects business logic with database backend. For simplicity, all other relations are modeled as "hosted-on" relations, which is the super type for "installed-on", etc. The architecture is a result of Cloud-related design rationales [5]. Reasons for using multiple Cloud providers are differences in pricing or quality of service and that a provider may not offer required services or features, e. g., AmazonRDS offers an automated backup functionality which is not supported by Azure currently.

2.2 Provisioning of the Web Shop Application

In this section, we describe the provisioning of our Web shop in detail. To provision the Ubuntu operating system and the virtual machine on Azure, the Windows Azure Service Management REST API[7] is invoked. The thereby instantiated VM is accessible via SSH. Hence, Chef can be used to install the Apache PHP Web server on it. Therefore, a Chef agent is installed on the operating system via SSH before. After the Web server is installed by executing the corresponding Chef recipes, we install God[8], which is a monitoring framework written in Ruby. To ensure high availability, we use God to make sure that the Web server is up—otherwise, a restart will be triggered automatically. After that, the PHP application files are transferred from an external storage onto the operating system and copied into the `htdocs` folder, which contains all applications that are hosted on the server. This is done via SSH and Secure Copy (scp), which is a means to securely transfer data between different hosts. To create the MySQL database instance on AmazonRDS, a single HTTPS call to Amazon's Query API[9] is sufficient. However, by default, network access to AmazonRDS instances is disabled. Thus, we authorize access before by creating a so called *security group* that defines the rules to make the database accessible for the PHP frontend hosted on another provider. This requires two HTTPS calls to Amazon's Query API. Afterwards, we setup frequently automated backups for the database to prevent data loss. This is also done by an HTTPS service call to the same API. After both application stacks are provisioned, initial product data is imported to the database. To do this, we employ an SQL batch update. In the last step, the PHP application needs to be connected to the database. Establishing this connection is application-specific as there is no standard or common way defining how to set such database endpoint information. Thus, only the Web shop developer knows how to configure the application to connect to the database. In our scenario, we employ a shell script that writes the database's endpoint information into a configuration file which is read by the PHP application. Such shell scripts typically need parameterization: the endpoint is passed to the script through environment variables which are read by the script.

[7] `http://msdn.microsoft.com/en-us/library/windowsazure/ee460799.aspx`
[8] `http://godrb.com/`
[9] `http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/`

2.3 The Challenge: How to Provision This Application Fully Automated? Tools and Related Work

The presented application architecture constitutes a set of problems. The fully automated provisioning is a challenge as various kinds of technologies are involved. Proprietary vendor- and Cloud-specific solutions such as Amazon Cloud Formation[10] are not able to tackle this provisioning issue completely as they consider a certain Cloud provider environment only and lack the integration of XaaS offerings from other providers. Proprietary multi-Cloud management services such as RightScale[11] aim to enable the provisioning and management of business-critical Cloud infrastructures across multiple IaaS providers based on automation scripts. However, the wiring of custom components and integration of various XaaS offerings is not possible. Cloud abstraction APIs such as OpenStack[12] provide a means to decouple dependencies to the underlying Cloud infrastructure and platform services in order to ease multi-Cloud applications. However, they do not solve the problem of integrating different management APIs, technologies, and XaaS offerings. The DevOps communities provide tools such as Puppet, Chef, or CFEngine and frameworks such as Marionette Collective[13] or Spiceweasel[14] to enable sophisticated configuration management [3, 6]. In addition, there are tools such as Juju to enable the orchestration of configuration management scripts. However, these script-centric approaches are mostly limited to installing and configuring software components on existing virtual machines. The deployment of complex composite Cloud applications that include the fully automated provisioning of various XaaS offerings with custom dependencies among each other is not trivial using these approaches as low-level scripts need to be written for integration. Another deficit of most approaches is that the wiring of different components across different machines, such as connecting the Web shop PHP frontend to the product database, cannot be modeled separately. Therefore, custom low level scripts must be manually embedded into the overall process what requires a lot of technical knowledge. Juju supports this kind of wiring, but also comes with the deficit that the used configuration management scripts are made to be executed on the target infrastructure such as a virtual machine that was provisioned before. To summarize, the main problem of all these tools is that a complete support for the provisioning, configuration, and wiring of virtual machines, storage, and other XaaS offerings, provided by different providers, is currently out of scope. Thus, they do not provide a means to integrate needed technologies to enable interoperable, multi-Cloud, and multi-vendor applications such as our Web shop. Besides implementing custom software or low level scripts from scratch that orchestrate all these technologies on their own, a common solution for this problem is implementing provisioning workflows that integrate required technologies, APIs, and abstraction frameworks as shown

[10] `http://aws.amazon.com/cloudformation/`
[11] `https://www.rightscale.com`
[12] `https://www.openstack.org/`
[13] `http://docs.puppetlabs.com/mcollective`
[14] `http://wiki.opscode.com/display/chef/Spiceweasel`

by Keller and Badonnel [7]. Provisioning workflows provide significant advantages in contrast to solutions from scratch: They inherit features from workflow technology such as recoverability, traceability, compensation-based recovery, and transactional support and they provide an accepted means for orchestration of heterogeneous software [9]. In addition, they support long-running processes and enable people involvement through human tasks, which may be needed to manage utilized physical hardware. Thus, provisioning workflows enable a flexible, reliable, and robust way to provision applications—even if multiple providers, vendors, and technologies are involved. However, implementing such workflows manually has two crucial drawbacks. (i) The required knowledge and effort is high. Developers do not only need the knowledge about the workflow language and its semantics itself but also have to know how to integrate and wrap technologies to make them accessible for workflows. This is a difficult, time-consuming, and error-prone task and often requires deep technical knowledge about certain technologies. Even using common workflow languages such as BPMN or BPEL needs a lot of detailed knowledge and, in addition, script-centric technologies have to be integrated which is not supported by BPMN and BPEL natively, for example. This causes a lot of glue code to wrap APIs and technologies. Especially script-based technologies provide some difficult challenges as they are typically tightly coupled to operating systems, need to be copied to target machines in advance, and employ different parameterization mechanisms, which are not interoperable. Thus, several steps are required before scripts can be executed. The seamless and transparent integration of different heterogeneous technologies is the major challenge if workflows are created manually. We developed a BPMN extension [8] that eases implementing provisioning workflows. However, this extension also does not solve the aforementioned problems completely as it also relies on the invocation of management services. Especially the second problem is not tackled by this extension: (ii) provisioning workflows are typically tightly coupled to a single application and hard to reuse and maintain [2]. If components or relations change, this needs to be adapted in the workflow. Thus, provisioning workflows must be created from scratch or by copying workflow fragments from other applications, which is an error-prone task. In summary, the manual implementation of provisioning workflows is hard, costly, and inefficient. Thus, we need a means to generate provisioning workflows for individual applications fully automated. The literature presents approaches that deal with this issue: Cafe is a framework that enables automating the provisioning of composite service-oriented Cloud applications [11]. It generates provisioning workflows by orchestrating so called “component flows” that implement a uniform interface to manage the provisioning of individual components. The work of Maghraoui et al. presents an operation oriented approach that enables transferring the current state of a data center into a desired state by orchestrating provisioning operations [10]. This orchestration is based on planning algorithms that investigate the preconditions and effects of each operation in order to determine the correct set and order of operations. The work of Eilam et al. also uses desired state models to provision applications [4]. In contrast to the previous work, it is based on graph covering techniques to orchestrate so called "automation signatures" that implement provisioning logic. However, none of these approaches supports the direct and generic integration of script-centric and service-centric operations that provide custom provisioning logic implemented by the application developer itself. This is especially needed for wiring custom components as discussed in Section 2.2. Using the available approaches, application developers need to write glue code to embed custom logic, i. e., custom component flows, provisioning operations, and automation signatures need to be implemented. Thus, the application developer requires technical knowledge about the technologies that makes the automated provisioning of custom applications complicated and costly.

3 Management Planlets and Provisioning Topologies

In this section, we explain Management Planlets and Provisioning Topologies, which are a means to automate the provisioning of applications. We introduced Management Planlets in a former work [2] and extend them in this paper to support the direct and explicit integration of script-centric and service-centric provisioning technologies, which is not supported by the original approach. Management Planlets provide small reusable workflows that perform low level management tasks on a certain combination of nodes and relations, e. g., installing a Web server or instantiating a virtual machine. The purpose of planlets is to be orchestrated into an overall workflow that provides a higher-level functionality. Thus, they serve as generic building blocks for the generation of provisioning plans that provision an application fully automated. Planlets consist of two parts: (i) An *Annotated Topology Fragment* that depicts the management tasks performed by the planlet on the nodes and relations and (ii) a workflow implementing this functionality. The topology fragment contains (i) a graph of typed nodes that may be interconnected by typed relations and (ii) so called *Management Annotations* that are attached to the nodes or relations. These annotations describe abstract tasks to be performed on the associated element: each annotation has well-defined semantics but exposes no details about its actual implementation. Thus, they hide complexity and describe tasks decoupled from concrete implementations. All details about the technical implementation are hidden behind the topology fragment and implemented by the workflow. For example, the *Create-Annotation* specifies that the associated element gets instantiated or installed by the respective planlet. The concrete implementation is up to the planlet. In addition, planlets implement a uniform interface for invoking them and define input parameters that have to be provided by the caller, e. g., required account credentials. Due to these properties, planlets are capable of integrating different technologies into a common model without exposing technical details. As planlets implement their functionality as workflows, they inherit the features from workflow technology as described in Section 2.3.

Provisioning Topologies are used to define the provisioning of applications. A Provisioning Topology is an application topology that consists of nodes and relations annotated with Management Annotations. These annotations define which tasks have to be performed to provision the application. Elements in Provisioning Topologies may specify properties and operations they provide. Operations provide information such as file references to scripts or URLs to service endpoints. This topology serves as input for a plan generator that generates a provisioning plan by orchestrating multiple planlets.

3.1 Management Annotations

There are two classes of Management Annotations: (i) *Structural Annotations* and (ii) *Domain-specific Annotations*. The first class is fixed and contains two generic annotations defining that the associated element gets *created* or *destroyed* by the planlet. Thus, they modify the topology in terms of its structure. The second class contains custom annotations that are needed to describe tasks of a certain domain. For example, an *ExportTables-Annotation* for database nodes defines that tables are exported by the planlet. Thus, this class makes the approach extensible towards all conceivable kinds of tasks. Furthermore, domain-specific annotations may define additional information in the form of properties which are used by the planlet for customization. For instance, the ExportTables-Annotation defines the tables that have to be exported and the target storage.

3.2 Preconditions and Effects

Planlets may define preconditions to be fulfilled prior to the execution of the planlet. Preconditions include value range restrictions on properties. Each property specified on an element contained in the planlet's topology fragment must be initially available in the topology model or set by another planlet before. To manage these properties, planlets write and read them from an *instance model*, which is based on the Provisioning Topology. The effects of planlets are expressed through Management Annotations and properties: Properties that are annotated with a Create-Annotation are created by the planlet and may be used by other planlets. For example, a planlet that deploys an application on a Web server may have preconditions in the form of server IP-Address and credentials which are set by another planlet that installed the Web server before. Thus, planlets use properties to communicate with each other indirectly. The preconditions and effects of planlets are matched against Provisioning Topologies to find a set of planlets that are able to provision the whole application.

Figure 2 presents an example. The Provisioning Topology depicted on the left gets provisioned by the two planlets shown on the right. The planlet on the bottom instantiates an Ubuntu Linux operating system running in a virtual machine on Amazon EC2. This is depicted by the topology fragment on the left of the planlet: the Ubuntu and VM nodes as well as the underlying hosted-on relations have a Create-Annotation attached and, thus, get instantiated by the planlet. This topology fragment is matched by the corresponding nodes and relations contained in the Provisioning Topology based on the types and annotations of the elements (depicted by the blue arrows). In addition, the state properties of both nodes get set to "instantiated" and the credentials and IP-address properties of the Ubuntu node

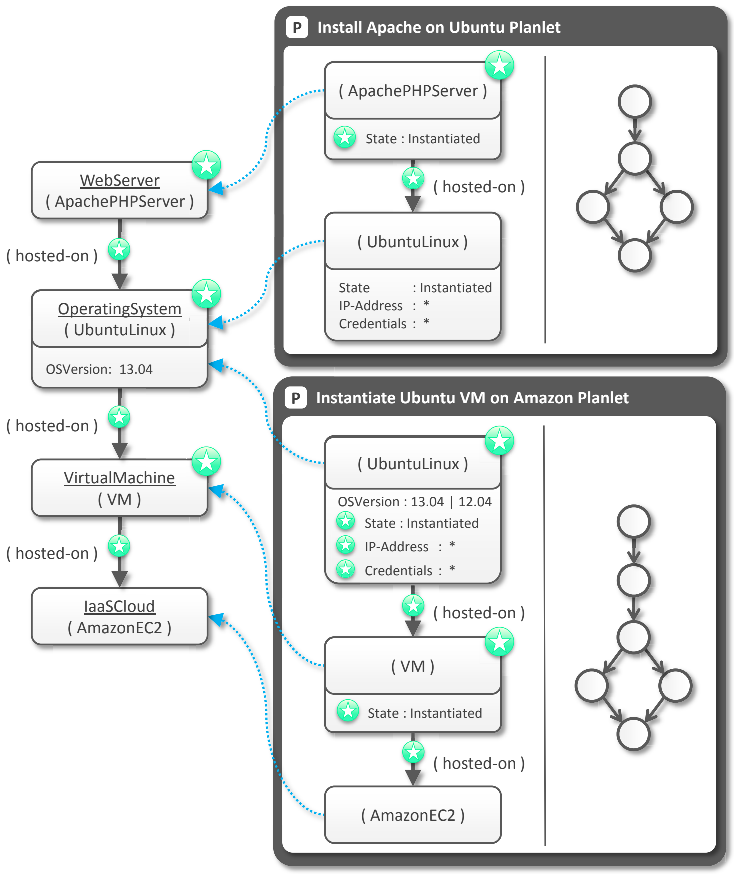


Fig. 2. Provisioning Topology (left) and matching Management Planlets (right)

to a meaningful value. This is expressed by the Create-Annotations attached to the corresponding properties. The planlet also defines a precondition in the form of the OSVersion property of the Ubuntu node: It is able to provision this stack for versions 13.04 and 12.04. This precondition is matched by the Provisioning Topology. As Amazon EC2 is always running, there is no need to instantiate that node explicitly. Thus, the planlet is applicable. The planlet on the top is able to install an ApachePHPServer on Ubuntu. It specifies preconditions on the Ubuntu node in terms of state, credentials, and IP-address, which are all fulfilled after the former planlet was executed. In general, preconditions determine the execution order of planlets.

3.3 Transparent Integration

In this section, we describe how Management Planlets are used to integrate script-centric and service-centric provisioning technologies transparently. Technical details are hidden by planlets as the topology fragment exposes the functionality only in an abstract fashion. This abstraction allows combining service-centric and script-centric technologies without exposing any details about the implementation to the plan generator: The implementation details are up to the planlet creator. As a consequence, the plan generator and the application developer, who specifies the Provisioning Topology, do not have to care about the technical details. We call this *Transparent Integration* because all technical details are invisible. This kind of integration is suited for regularly used tasks in which standard or common components are involved that are both not specific to a certain application. For example, the instantiation of an Ubuntu virtual machine on Amazon EC2 is such a task in which service invocations are involved. Installing an Apache Web server on this virtual machine afterwards is a typical script task. Therefore the Chef community, for instance, provides recipes for this installation. All in all, both tasks are suited to be integrated transparently through *Generic Planlets* because each component as well as both tasks are common and likely to be reused. This transparent integration supports the reusability of expert knowledge across different domains and technologies. We implemented the plain provisioning of both stacks of our motivating scenario as Generic Planlets. That includes provisioning of VM and operating system, installation of the Web server, deployment of the PHP application, and the instantiation of DBEnvironment and product database.

3.4 Standards Compliance

Proprietary approaches as discussed in the related work (Section 2.3) are based on proprietary domain-specific languages (DSLs). DSLs prevent these approaches to be portable and accepted by a broad audience as the languages require special knowledge and technical skills. A main strength of Management Planlets is the applicability to the OASIS TOSCA standard [12], which provides a standardized format to describe application topologies and their management in a portable fashion. We proved this by a prototypical implementation [2].

3.5 Customization Drawback

As shown in the previous sections, planlets provide a means to abstract tasks from concrete operation implementations to automate their combined execution. Application developers only need to specify the desired tasks in a Provisioning Topology without having any doubt about the final execution. However, these Generic Planlets cannot serve all customization requirements the application developers may have. Custom components such as the Web shop frontend of our motivating scenario often need special consideration. For example, to establish the connection to the database, the application needs a configuration in the form of database credentials and endpoint information. As there is no standardized or common way for this task, it is not possible to implement a planlet that deals with this in a generic and reusable way. Thus, application developers need to implement specific planlets to inject custom behavior for an actually simple task. These so called *Custom Planlets* can be combined with the Generic Planlets in a seamless way. This is, however, not always sufficient as writing planlets causes unnecessary overhead if only a simple service call or script execution is needed to serve the needs. In addition, the Custom Planlets are hardly reusable as they are targeted to a very special custom task. To tackle this issue, we extend the concept of Management Planlets in the following section. The presented extension enables configuration of provisioning by integrating service calls or script executions directly into the generated provisioning plan without corrupting the overall concept. The new approach enables application developers to customize provisioning without the need to write planlets by themselves.

4 Integrating Script- and Service-Centric Technologies

Management Planlets currently support the Transparent Integration of script-centric and service-centric provisioning technologies implicitly. However, if special tasks are needed, implementing Custom Planlets is not appropriate as discussed in Section 3.5. Thus, we need a means to *explicitly* integrate customization scripts and service calls generically. Therefore, we extend the concept of Management Planlets to enable an *Explicit Integration* of different technologies into the generated provisioning workflow. The combination of Transparent and Explicit Integration enables integrating script-centric and service-centric provisioning technologies in a seamless fashion. This allows application developers to benefit from Generic Planlets and to customize the provisioning at any point through using the Explicit Integration without the need to write own Custom Planlets.

4.1 Explicit Integration

In this section, we present the main contribution of this paper that provides a means to integrate script executions and service invocations of various types explicitly into the overall automatically generated provisioning flow. Therefore, we introduce *Generic Lifecycle Management Planlets* (*GLMP* from now on) that serve as generic technology integration mechanism to implement lifecycle actions. GLMPs enable custom implementation of provisioning and configuration logic specifically for a certain application, component, or relation. They are able to directly execute a specific script- or service-based operation implementation that implements one or multiple Management Annotations. Thus, GLMPs enable binding custom operation implementations to abstract tasks which are represented by Management Annotations. This enables application developers to inject own provisioning and configuration logic without the need to implement Custom Planlets. Thus, the operational logic gets distributed over the Provisioning Topology of an application and Management Planlets.

Figure 3 shows the general concept by an example describing how an HTTPS service call can be used to configure a node. On the left, it depicts the DBEnvironment of the motivating scenario. This node needs to be created and configured, as denoted by a Create-Annotation and a Configure-Annotation. It provides a SetupFrequentBackup operation of type HTTPS (1 in Figure 3) that implements (2)

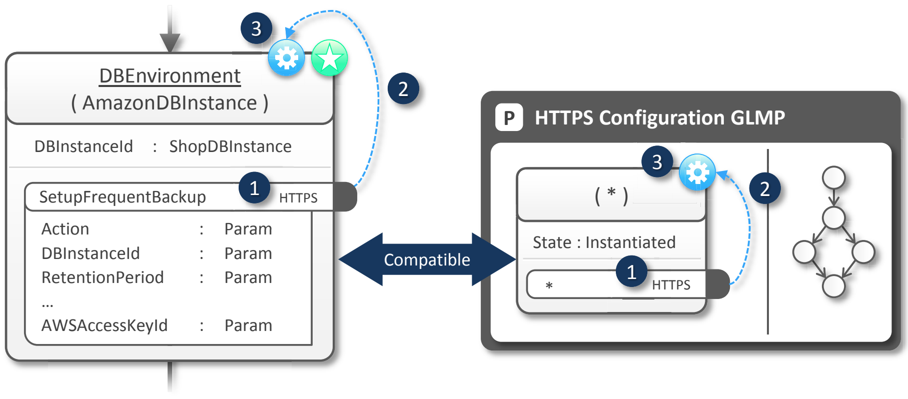
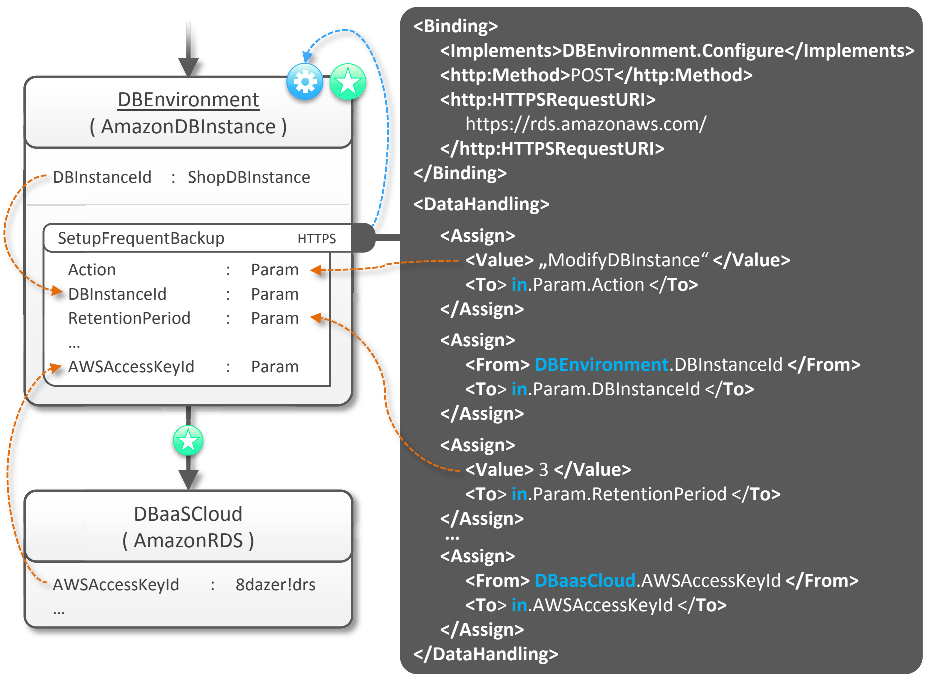


Fig. 3. DBEnvironment node with custom HTTPS operation bound to Configure-Annotation (left) and compatible HTTPS Configuration GLMP (right)

the Configure-Annotation (3). On the right hand side, the figure shows an HTTPS Configuration GLMP that matches the DBEnvironment node: The GLMP is applicable to single nodes of any type (denoted by the star symbol in the type parentheses). It is able to invoke services via HTTPS (1 in Figure 3) in order to configure a node (3). This is shown by the operation-annotation-binding (2). The planlet, however, has a precondition in the form of a state property that must be set to instantiated. Thus, this GLMP is applicable after the DBEnvironment node was created by another planlet that sets the state-property accordingly.

A GLMP is responsible to perform one or multiple lifecycle actions on an element. For this paper, we use a simple lifecycle that is sufficient for most provisioning scenarios [4]: each node and relation goes through the lifecycle actions *instantiation*, *configuration*, and *termination*. The instantiation action is represented by the Create-Annotation. Each planlet that performs this annotation sets the state-property on the corresponding element to "instantiated" after completion. This state-property serves as precondition for planlets that perform the Configure-Annotation on that element. Thus, the Configure-Annotation is always processed after the Create-Annotation. This enables a fine grained injection of provisioning logic in the desired phase of an element's lifecycle. Termination is out of scope for this paper. This lifecycle may be extended to support more complex needs, e. g., by executing Prepare-Annotations before instantiation.

Operation-Annotation-Binding. An operation may be implemented through various kinds of technologies. Thus, their integration mechanisms differ significantly from each other and need specific additional information. GLMPs offer a way to define information for each operation type individually: Each operation type defines its custom binding information as shown in Figure 4. The type of the operation defines its semantics and the meaning of this information. The example shows binding information for an HTTPS call. All elements contained in the binding having the prefix "http" are technology-specific and ignored by the plan generator. They are used to provide needed information used by the GLMP, such as request URI and HTTP method in this example. The *Implements* elements specify the annotations that are implemented by the operation. This information is used by the plan generator to select GLMPs, which is explained in Section 4.1. Binding-information may not be available at design time, e. g., endpoints of configuration services provided by components themselves are typically not known until the component is provisioned. The *Data Handling Specification* introduced in the next section enables defining lazy bindings.



```
<Binding>
    <Implements>DBEnvironment.Configure</Implements>
    <http:Method>POST</http:Method>
    <http:HTTPSRequestURI>
        https://rds.amazonaws.com/
    </http:HTTPSRequestURI>
</Binding>
<DataHandling>
    <Assign>
        <Value> „ModifyDBInstance“ </Value>
        <To> in.Param.Action </To>
    </Assign>
    <Assign>
        <From> DBEnvironment.DBInstanceId </From>
        <To> in.Param.DBInstanceId </To>
    </Assign>
    <Assign>
        <Value> 3 </Value>
        <To> in.Param.RetentionPeriod </To>
    </Assign>
    ...
    <Assign>
        <From> DBaasCloud.AWSAccessKeyId </From>
        <To> in.AWSAccessKeyId </To>
    </Assign>
</DataHandling>
```

Fig. 4. Data Handling Specification for GLMP that invokes an HTTPS Web service

Data Handling Specification. Operations may have several input and output parameters. This depends on the implementation technology: Services have typically one output parameter, scripts may define several environment variables as output parameters. Thus, data handling is operation type specific and up to the corresponding GLMP. Many kinds of operations need input parameters that depend on properties in the topology. For example, to setup the frequent backup on the DBEnvironment node of our motivating scenario, an HTTPS call to the Amazon Query API is required. This call needs query parameters such as DBInstanceId and AWSAccessKeyId, which are properties of nodes: The DBInstanceId is a property of the DBEnvironment node itself whereas the AWSAccessKeyId is a property of the underlying DBaaSCloud node. Therefore, we introduce a Data Handling Specification as shown in Figure 4. This allows assigning properties and default values to input parameters (path element "in") and output parameters (path element "out") to element properties as shown in Figure 4. This specification is read by the plan generator and GLMP. The plan generator uses the information to check the applicability of a GLMP by analyzing assigns: If an assigned topology property is not available, the implemented annotation is not executable. GLMPs use this specification to retrieve the needed input parameters by accessing the specified element properties and default values. Converting data is up to the GLMPs, e. g., a string property to an environment variable for scripts. All input parameters which are not assigned with a default value or property are exposed to the input message of the generated provisioning plan. These parameters have to be set by the caller and are routed by the provisioning plan to the corresponding GLMP that receives an additional list of parameters that are not defined in the specification as input. This kind of data handling is similar to data assign activities in BPEL and Variability Points in Cafe [11]. To enable lazy binding, the Data Handling Specification may be used also to assign properties to operation-specific elements in the binding specification. This information can be read by the GLMP to complete the binding at runtime.

Technology-Specific Preconditions. The preconditions of a GLMP depend on the technology to be integrated. Services offered by nodes have other preconditions than scripts that typically run on the underlying operating system. Thus, the planlet's topology fragment depends on characteristics of the respective technology and provides a means to define preconditions in the form of nodes, their dependencies, and properties. To provide an additional means that enables defining very specific requirements of operations, we extend management operations by properties: each property of an operation defined in the topology must be compatible with a property of the corresponding operation defined in the planlet's topology fragment. The set of properties is prescribed by the operation's type. In contrast to the Data Handling Specification, which is processed by the planlets, these properties are evaluated by the framework to select appropriate planlets during plan generation. Properties are required, for example, for defining script-based management operations on relationships. If a script implements the Create-Annotation of a directed relationship in the Provisioning Topology, a property is used to define if it has to be executed on the infrastructure of the source node or on the infrastructure of the target node. Depending on the used operating systems, different planlets may be needed for different property values.

Extended Workflow Generation. In Breitenbücher et al. [2], we presented a concept for provisioning plan generation based on planlets. We extend this algorithm to support GLMPs. In general, GLMPs are used similarly to normal planlets: they are checked for compatibility in terms of types, preconditions, effects, and Management Annotations. However, the generation algorithm needs to be extended: (i) first, the matchmaking of topology and planlet fragments is extended to support operation-annotation binding specification of GLMPs: If the plan generator checks compatibility of annotations that are implemented by an operation, both annotation and implementing operation in GLMP and Provisioning Topology must have the same types. In addition, the properties of both operations must be compatible. (ii) The generator has to consider the Data Handling Specification, which influences applicability of GLMPs: an annotation implemented by an operation must not be processed by a GLMP until all assigned properties are available. In addition, output parameters that are written into properties must be considered, too. (iii) The generator prefers GLMPs whenever possible to Generic Planlets in order to treat custom implementations with a higher priority. This enables application developers to customize the provisioning.

5 Case Study

In this section, we present a case study to prove the approach's feasibility based on the Web shop. As already mentioned in Section 3.3, the plain provisioning of both stacks is completely done by Generic Planlets. To perform the custom tasks, we developed GLMPs that support the following technologies and tasks:

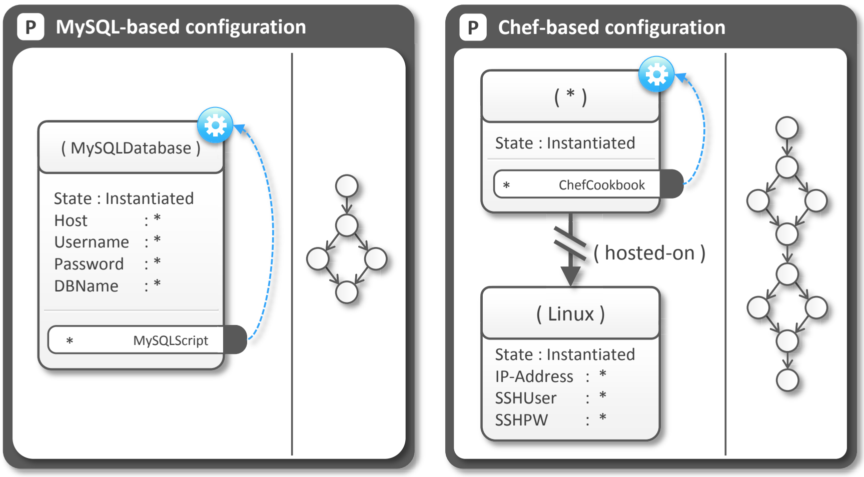


Fig. 5. GLMPs executing MySQL scripts (left) and Chef cookbooks (right)

To establish the MySQLConnection from PHP application to database, we implemented a small shell script that gets the endpoint of the database and credentials as input via environment variables. To setup the frequent database backup, we use the Amazon Query API that provides HTTPS services for this task. The corresponding GLMP is shown in Figure 3, the corresponding binding and Data Handling Specification in Figure 4. The initial data import into the product database is done via an SQL script. The corresponding GLMP is shown in Figure 5 on the top left. The binding specifies the location of the SQL script, database endpoint and credentials are extracted by the GLMP from the MySQL-Database node automatically. To ensure that this information is available, the GLMP defines a target node of type MySQLDatabase that must be running already. Thus, this example shows how technology specific preconditions can be used to model the requirements and characteristics of technologies. To install God to monitor the Apache Web server, we use a Chef cookbook to implement the corresponding operation which is bound to a Configure-Annotation attached to the Web server. The GLMP shown in Figure 5 on the right executes this. It uses a transitive relation, which ignores all elements between source and target, of type hosted-on and requires an underlying Linux operating system with corresponding SSH credentials and IP-Address. Internally, it installs the Chef agent on the operating system by using SSH and executes the specified cookbook.

6 Evaluation

In this Section, we evaluate our approach against the most similar publicly accessible implemented approaches. The features compared in Table 1 are derived from the challenges discussed in Section 2.3 and explained in the following. Thus, they represent requirements that must be fulfilled to be able to provision the kind of Cloud applications used in our motivating scenario (cf. Section 2.1) fully automated. An x denotes that the approach supports the corresponding functionality without limitations. An x in parentheses denotes partial support.

Table 1. Feature Evaluation

Feature	GLMP	CloudForm.	Heat	Puppet	Chef	Juju	Workflows	Cafe
Component Wiring	x	(x)	(x)	(x)	(x)	x	x	x
XaaS Integration	x						x	x
Multi-Cloud	x		(x)	x	x	(x)	x	x
Full Customization	x						x	x
Multi-Script	x	(x)	(x)	(x)	(x)	x	(x)	x
Multi-Service	x	(x)	(x)	(x)	(x)	(x)	x	x
Explicit Integration	x							
Transparent Integration	x			x	x	x		x
Fully-Automated	x	x	x	(x)	(x)			x
Standards Compliant	x						(x)	(x)
Complete Top. Model	x							

Component Wiring means establishing relations between nodes, e. g., connecting a PHP application to a database. CloudFormation enables this by embedding hard-wired scripts into the template that access properties of resources. This is similar to our data flow definitions but limited to resources that are contained in the template and, thus, provisioned on Amazon. Heat implements the CloudFormation specification and comes with similar problems. The script-centric technologies are able to wire components on a very low level based on custom scripts. However, this approach is limited as discussed in Section 2.3. *XaaS Integration* means the capability to provision and configure multiple service offerings of different types. CloudFormation and Heat are coupled to the service types provided by Amazon and cannot deal with others in a practicable way. The *Multi-Cloud* feature enables integrating different Cloud providers. All approaches support this except CloudFormation that is bound to Amazon. Heat supports this partially as OpenStack is used as Cloud operating system. All script-centric technologies are not affected by the underlying Cloud infrastructure and, thus, multi-Cloud ready. Juju can be used in conjunction with different Cloud providers, but a particular composite service instance is bound to one provider. *Full Customization* means that provisioning may be customized by the application developer in each detail. The concept of GLMPs enables this in three ways: (i) custom Planlets may be implemented to customize certain node combinations, (ii) GLMPs are able to integrate low level logic in a seamless fashion, and (iii) as Management Planlets are used to generate a Provisioning Workflow that is executed after the generation, this workflow may be adapted arbitrarily. All other features except workflows and Cafe are not able to provide this feature. *Multi-Script* and *Multi-Service* means that various kinds of script and management service calls respectively can be integrated seamlessly into the provisioning process. GLMPs support this integration directly as shown in Section 5. With Juju it is possible to combine different scripting languages. All other technologies, except workflows, that are made to orchestrate services, and Cafe, that employs workflows, only indirectly as they need wrapping code (scripts) for integration. *Explicit Integration* means that no visible glue code is needed for integrating any technology. GLMPs support this feature as technical execution details are hidden. Script-centric approaches do not support this feature as they need to create glue code, e. g., for invoking SOAP services. Even workflows, and thus Cafe, do not support this feature as they do not abstract from fine-grained tasks and wrapping code is needed to invoke scripts. *Fully-Automated* provisioning is the key feature for Cloud computing. The script-centric technologies Puppet and Chef do not support this feature directly. To enable the fully-automated provisioning of multi-stack applications, there are tools such as Marionette Collective or Spiceweasel. Workflows need to be created by hand. *Standards Compliant* ("de jure") are only Management Planlets as they support TOSCA as topology model and BPEL as workflow language. Workflows and Cafe are partially compliant as they may use the BPEL or BPMN standards but do not support any standard for the modeling of applications. This feature is important to create portable applications. *Complete Topology Model* means that nodes as well as relations are explicitly modeled. Only Management Planlets support this feature that is important to maintain the application: if relations are modeled implicitly only, it is hard to recognize them. Of course, most of the missing features of CloudFormation, Heat, and the script-based approaches can be emulated by using low-level shell scripts. However, this is not efficient and the result is hard to maintain.

7 Conclusion and Future Work

In this paper, we presented an approach that enables the integration of script-centric and service-centric provisioning and configuration technologies. The approach is based on Management Planlets and enables to customize and automate the provisioning of composite Cloud applications. The validation showed the feasibility of our approach and the detailed evaluation, comparing the supported features to other approaches in this area, proved its relevance. We plan to extend our approach in the future to support influencing the execution order of Management Annotations in order to provide a fully customizable approach independent from restricted lifecycle operations. In addition, we focus on management of applications and apply the presented approach also in this area.

Acknowledgments. This work was partially funded by the BMWi project CloudCycle (01MD11023).

References

1. Breitenbücher, U., Binz, T., Kopp, O., Leymann, F., Schumm, D.: Vino4TOSCA: A Visual Notation for Application Topologies Based on TOSCA. In: Meersman, R., et al. (eds.) OTM 2012, Part I. LNCS, vol. 7565, pp. 416–424. Springer, Heidelberg (2012)
2. Breitenbücher, U., et al.: Pattern-based runtime management of composite cloud applications. In: CLOSER (2013)
3. Delaet, T., Joosen, W., Vanbrabant, B.: A Survey of System Configuration Tools. In: 24th Large Installations Systems Administration Conference (2010)
4. Eilam, T., et al.: Pattern-based composite application deployment. In: Integrated Network Management. IEEE (2011)
5. Fehling, C., et al.: Cloud Computing Patterns: Fundamentals to Design, Build, and Manage Cloud Applications. Springer (2013)
6. Günther, S., Haupt, M., Splieth, M.: Utilizing Internal Domain-Specific Languages for Deployment and Maintenance of IT Infrastructures. Tech. rep., Very Large Business Applications Lab Magdeburg (2010)
7. Keller, A., Badonnel, R.: Automating the provisioning of application services with the BPEL4WS workflow language. In: Sahai, A., Wu, F. (eds.) DSOM 2004. LNCS, vol. 3278, pp. 15–27. Springer, Heidelberg (2004)
8. Kopp, O., Binz, T., Breitenbücher, U., Leymann, F.: BPMN4TOSCA: A domain-specific language to model management plans for composite applications. In: Mendling, J., Weidlich, M. (eds.) BPMN 2012. LNBIP, vol. 125, pp. 38–52. Springer, Heidelberg (2012)
9. Leymann, F., Roller, D.: Production workflow: concepts and techniques. Prentice Hall PTR (2000)

10. El Maghraoui, K., Meghranjani, A., Eilam, T., Kalantar, M., Konstantinou, A.V.: Model driven provisioning: Bridging the gap between declarative object models and procedural provisioning tools. In: van Steen, M., Henning, M. (eds.) Middleware 2006. LNCS, vol. 4290, pp. 404–423. Springer, Heidelberg (2006)
11. Mietzner, R.: A method and implementation to define and provision variable composite applications, and its usage in cloud computing. Dissertation, Universität Stuttgart (August 2010)
12. OASIS: Topology and Orchestration Specification for Cloud Applications Version 1.0 (May 2013)

SCaaS: A Platform for Managing Adaptation in Collaborative Pervasive Applications

Muhammad Ashad Kabir[1], Jun Han[1], Alan Colman[1], and Jian Yu[1,2]

[1] Faculty of Information and Communication Technologies, Swinburne University of Technology, Melbourne, Australia
{akabir,jhan,acolman,jianyu}@swin.edu.au
[2] School of Computing and Mathematical Sciences, Auckland University of Technology, New Zealand

Abstract. In this paper, we present a *social context as a service (SCaaS)* platform for managing adaptations in collaborative pervasive applications that support interactions among a dynamic group of actors such as users, stakeholders, infrastructure services, businesses and so on. Such interactions are based on predefined agreements and constraints that characterize the *relationships* between the actors and are modeled with the notion of *social context*. In complex and changing environments, such interaction relationships, and thus social contexts, are also subject to change. In existing approaches, the relationships among actors are not modeled explicitly, and instead are often hard-coded into the application. Furthermore, these approaches do not provide adequate adaptation support for such relationships as the changes occur in user requirements and environments. In our approach, inter-actor relationships in an application are modeled explicitly using social contexts, and their execution environment is generated and adaptations are managed by the SCaaS platform. The key features of our approach include externalization of the interaction relationships from the applications, representation and modeling of such relationships from a domain and actor perspectives, their implementation using a service oriented paradigm, and support for their runtime adaptation. We quantify the platform's adaptation overhead and demonstrate its feasibility and applicability by developing a telematics application that supports cooperative convoy.

1 Introduction

Mobile computing has brought a wireless revolution in recent years, enabling mobile internet access as an indispensable way of modern life. It has radically changed the way people perform tasks, access information and interact with one another. At the same time, the emergence of service-oriented technology and interoperability standards (e.g., Web Services) has made it possible to develop systems intended to support people's social activities and organizations' work. This trend has paved the way of a new breed of software systems that can mediate both tasks of individuals and collaborative tasks of a group in pervasive environments [1].

We view a collaborative pervasive environment (CPE) as an *interaction space* among users where users collaborate with each other towards a common objective by sharing information that has been provided by other participating actors. These actors could be users or services or sensors embedded in the environment. Such collaboration is subject to the *agreements* and *constraints* relevant to the *relationships* among actors, which are dynamic and need to *adapt* in response to the changes in user requirements and environmental factors.

Collaborative pervasive applications (CPAs), raise major challenges in terms of their development and management, including the dynamic aspect of them and their environments. The use of services [15] offers the possibility to design, deploy and manage these applications dynamically, providing the flexibility required.

Most of the approaches in the pervasive computing literature mainly focus on tasks of individual users, and provide very limited support to collaborative tasks of a group of actors [2]. Furthermore, existing approaches in developing collaborative pervasive applications (e.g., [4–8]), and middleware architectures and frameworks for pervasive computing [14] are limited in supporting the dynamic *relationships* between actors, which themselves are an important aspect of context. In particular, there is a lack of support for managing the *adaptation* in such relationships in response to *changes* in the requirements and environments.

This research explores the concept of *social context* as a means to represent the relationships among actors and presents a *platform* to provide support for *adaptation* by managing such social contexts and their changes, in a service-oriented manner. Social context in computing is often used to refer to the people, groups and organizations that an individual interacts with [3]. Taking this view, we define *social context* as a representation of the interactions among the relevant actors. That is, social context defines the *constructed relationships* between social roles, and these relationships *define* and *constrain* the interactions between the actors playing those roles. To model social context, we employ Role-Oriented Adaptive Design (ROAD) [16] among many different approaches to design role-based software systems using *agent* paradigm, as ROAD brings a number of design principles that support flexible management and runtime adaptations. One of the key principles of ROAD is separation of *functional* and *management* operations. We model social context from two perspectives: domain-centric and player-centric [12]. A *domain-centric social context (DCSC)* model captures a collaborative view of the interaction relationships among the actors whereas a *player-centric social context (PCSC)* model captures an actor's coordinated view of all its interactions (across domains).

In this paper, we present a SCaaS platform for developing collaborative pervasive applications from their *high-level* specifications (represented in terms of DCSCs and PCSCs), and for managing their *adaptations* to cope with *runtime* changes. Figure 1 presents an overview of the SCaaS platform. SCaaS takes the DCSC and PCSC models (specified by an application designer) as inputs, and *instantiates* these models by generating *management* and *interaction* interfaces (as Web Services) for applications to invoke. Thus, using applications (running on mobile devices) actors can interact with each other and manage their social

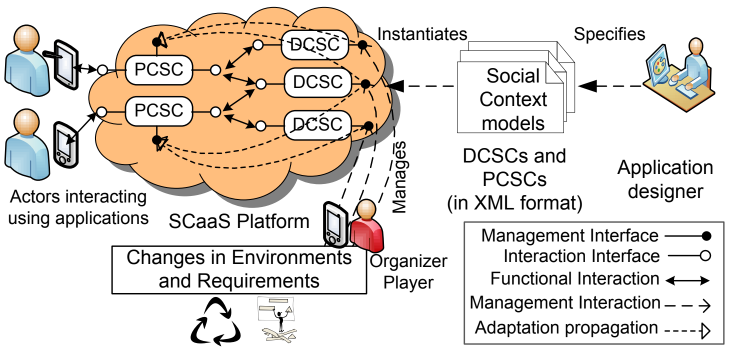


Fig. 1. Overview of the *SCaaS* platform

contexts. In particular, we focus on how the SCaaS platform supports runtime adaptation in social contexts to cope with changes in the evolving user requirements and environmental factors. In relation to the SCaaS platform, this paper makes the following four major contributions: (1) we identify different types of changes and the various adaptations to cope with such changes, and we propose management operations and present possible ways to perform adaptations using these operations, (2) we introduce *states* of the social context and its elements to enable adaptation in a safe manner, (3) we propose a *protocol* for adaptation propagation from DCSCs to PCSCs, and (4) Finally, we implement a *SCaaS platform* for creating the *execution* environment of social contexts and managing their runtime *adaptations*, and quantify the platform's *adaptation overhead*.

The paper is organized as follows. Section 2 presents a scenario where applications need to interact, collaborate and adapt in pervasive environments. After giving an overview of our social context models in Sect. 3, we present the SCaaS platform for managing adaptations in social contexts in Sect. 4. Section 5 discusses its prototype implementation, while Sect. 6 presents the experimental evaluation and a case study. After reviewing related research in Sect. 7, we conclude the paper in Sect. 8.

2 Application Scenario

Consider that two groups of tourists in two cars hired from two different rental companies want to drive together from Melbourne to Sydney. The car rental companies provide different types of support to their customers based on the customers' insurance policies, service availability, and so on. Let us assume that the two cars, Car#1 and Car#2, are rented from Budget and AVIS respectively.

In a *cooperative convoy*, a vehicle interacts with other vehicles, service providers and infrastructure systems to make the travel safe and convenient. Through these interactions a vehicle can share information (acquired from the service providers and infrastructure systems) with other vehicles. Such interactions are subject to defined agreements and constraints among the entities (i.e., vehicle to vehicle, vehicle to service providers, and vehicle to infrastructure). For instance, drivers of Car#1 and Car#2 want to form a cooperative convoy to make their travel safe and convenient by collaborating and interacting with each other. These two cars have access to different types of services: Car#1 has access to a Travel Guide Service (TGS) while Car#2 has access to a real time Traffic Management Service (TMS). In the cooperative convoy, they decide that Car#1 is the leading car (LC) whilst Car#2 is the following car (FC). The two cars follow the same route chosen by the leading car as it has access to a TGS. In addition, the drivers of both cars agree on a number of issues. For instance, they will always keep the distance between them less than 1000m. Car#2 (the following car) will send road blocks information (obtained from its TMS) to Car#1 (the leading car) if there is any. Car#1 gets the updated route plan from its TGS by specifying its preferences (e.g., avoid the route with blocked road) and notifies that route information to Car#2. Both cars notify each other of their positions every 10 seconds. If either vehicle experiences mechanical problems (e.g., flat tyre, engine issue) it needs to notify the other vehicle.

Applications facilitating the cooperative convoy need to fulfill two major requirements. **First**, the applications should support interactions complying with the *agreed* interaction relationships (i.e., constraints and obligations). For the drivers to perform the additional tasks (e.g., forwarding information) may cause distraction and have undesirable consequences. Thus, to facilitate collaboration with less distraction, the applications need to provide a coordinated view of the interactions, allow drivers to specify their coordination preferences and perform the coordination in an automated manner. **Second**, the applications need to support runtime adaptation as the interaction relationships evolve over time, and need to *adapt* with the *changes* in requirements and environments. For instance, a mechanical problem of the leading car may require it to handover the leading car role to one of the following cars (assuming there are multiple following cars). Because of heavy rain, the maximum distance may need to be reduced from 1000m to 600m. A third vehicle could join when the convoy is on the way; or the break-down of a following vehicle might result in its leaving the convoy before reaching the destination.

To address the first requirement, in our previous work [12], we have proposed an approach to modeling interaction relationships from both the domain and player perspectives. The DCSC model allows the interactions associated with a domain such as Budget or AVIS or Cooperative Convoy to be captured, while the PCSC model provides an overall view of all the interactions of a particular individual (e.g., driver of Car#1) and allows coordination among its interactions.

To address the second requirement, in this paper, we propose the SCaaS platform where the DCSC and PCSC are the basis of this platform. At runtime, the interaction relationships captured by DCSCs need to *adapt* with the *changes* in user requirements and environments. However, the PCSCs are dependent on DCSCs. The SCaaS platform *manages* both the DCSCs and PCSCs, and their dependencies in a consistent manner by supporting the adaptations in the DCSCs and the *adaptation propagation* from DCSCs to PCSCs as changes occur.

3 Social Context Models: An Overview

In this section, we briefly discuss the DCSC and PCSC, and illustrate how these social context modeling perspectives allow us to capture the above cooperative convoy scenario. A detailed discussion can be found in [12].

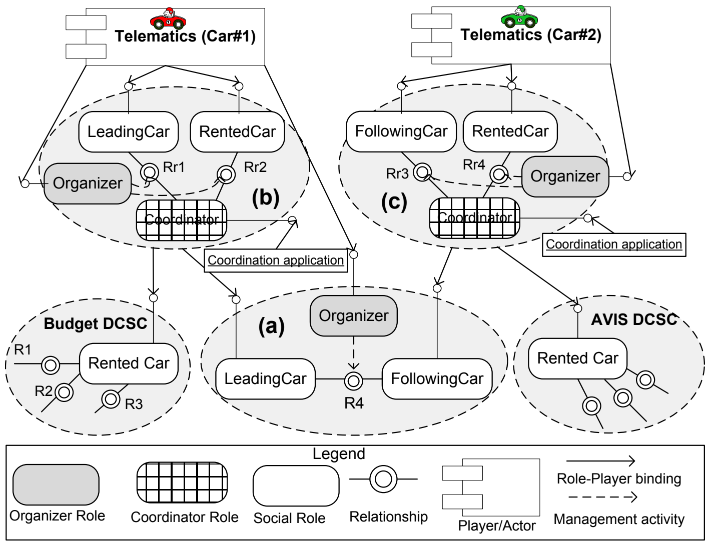


Fig. 2. Social context models for a cooperative convoy: (a) ConvoyDCSC, (b) Car#1PCSC, (c) Car#2PCSC

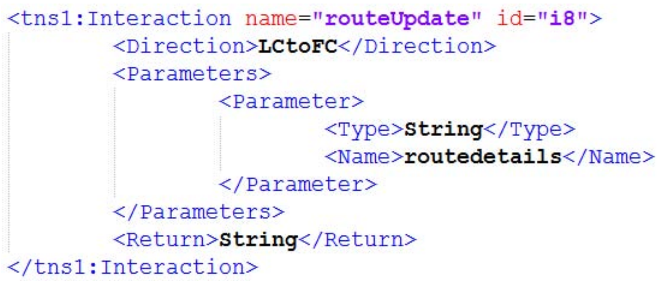
3.1 Domain-Centric Social Context Models

The *Domain-Centric Social Context* (DCSC) model captures the relationships among social roles associated with a particular domain or environment such as a company, a cooperative convoy, and so on. A DCSC model comprises of four key elements: social role, relationship, player, and organizer role. A *social role* represents the expected functional interactions of a participating actor with respect to the social context. Social roles are loosely-coupled elements and are modeled as first class entities, and as such they are separated from their players (e.g., actors) who play those roles. A *relationship* is an association between two social roles, which represents the interactions and interdependencies between those roles (or their corresponding players or actors). It mediates the interactions between social roles by defining what functional *interactions* can occur between the social roles and the sequences of these interactions (named *conversations*). In addition, a relationship also defines the *non-functional* requirements of the interactions in terms of *operational parameters* and *obligations* (e.g., time constraints on interactions) imposed on the players associated with that relationship. The *organizer role* and its player provide the capability for *managing* and *adapting* a social context to cope with *changes* in the requirements and dynamic environments. While social roles, players and relationships are entities at the social context's functional layer, the organizer role and its player are entities at the social context's management layer.

In the above scenario, there are three domains that need to be modeled, namely, the two car rental companies (Budget and AVIS) and the cooperative convoy. According to the agreements between the drivers of the two cars, the *ConvoyDCSC* model (see Fig. 2a) consists of two roles: *LeadingCar* (LC) and *FollowingCar* (FC), and their interactions are captured in the R4 (LC-FC) relationship (see Table 1) where an interaction is represented using a message signature and a direction of the message (i.e., AtoB, BtoA or both). Figure 3 shows an XML representation of the *i*8 interaction. Also, there may be other constraints and behavioral properties (e.g., conversations, obligations (e.g., o1 in Table 1)), but for simplicity we do not include all of them. As Car#1 plays the leading car role, it is also designated to play the organizer role of the *ConvoyDCSC* model. So, by playing the organizer role (through an application interface), the leading car driver can change the model (e.g., add, delete or update roles and relationships) at runtime. In a similar way, we can also model *BudgetDCSC* and *AVISDCSC* where Car#1 and Car#2 play the *RentedCar* role. Due to page limit, we do not present the details of these models which can be found in [17].

Table 1. Partial description of R4 (LC-FC) relationship in *ConvoyDCSC*

Specification from scenario	Notational representation
FC sends ahead road blocks information to the LC	i7:{notifyRoadBlock, FCtoLC, ack}
LC updates the route information to the FC	i8:{routeUpdate, LCtoFC, ack}
Both cars notify each other of their positions every 10 seconds	i9:{positionUpdate}//bi-directional o1:{i9,Time,periodic,=,10,seconds}
One car notifies mechanical problems to the other car	i10:{notifyMechanicalIssue}
Either vehicle may leave the convoy	i11:{leaveConvoy}
Maximum distance between the LC and FC is 1000m	p1:{maxDistance=1000m}



```
<tns1:Interaction name="routeUpdate" id="i8">
        <Direction>LCtoFC</Direction>
        <Parameters>
                <Parameter>
                        <Type>String</Type>
                        <Name>routedetails</Name>
                </Parameter>
        </Parameters>
        <Return>String</Return>
</tns1:Interaction>
```

Fig. 3. XML representation of the *i*8 social interaction

3.2 Player-Centric Social Context Models

In addition to the common view of a domain-centric social context model, an actor may have its own perception or view of the domain with respect to the role(s) it plays and the interactions it participates in that domain. Moreover, an actor may operate in different domains. Thus, we also model the social context from an actor's perspective, namely *Player-Centric Social Context* (PCSC). The PCSC model provides an overall view of all the interactions of an individual (across different domains) and allows coordination of its interactions.

Fig. 2b and Fig. 2c show the player-centric models of Car#1 and Car#2 respectively (and how they relate to the domain models). Like a DCSC, a PCSC contains social roles, actors/players and an organizer role. In addition, it contains a *coordinator role* and *role-centric relationships.* In the PCSC, all social roles are played by the actor (application) in question and are connected with the coordinator role through the role-centric relationships. The *coordinator role* is a means of achieving inter-domain coordination, i.e., interactions in one domain can be used for interactions in another domain. An actor can coordinate its interactions explicitly through the application's user interface or it may define some rules or use an intelligent application to coordinate its interactions on its behalf [12]. A *role-centric relationship* is the aggregation of all the relationships associated with a particular *social role* in a DCSC model, but localized in the player-centric model. For example, the $Rr2$ role-centric relationship in the PCSC of Car#1 is the aggregation of $R1$, $R2$ and $R3$ in the *BudgeDCSC.*

4 The SCaaS Platform

4.1 Social Context at Runtime

Social contexts are not just a modeling or design-time construct, and they are also *runtime* entities that *mediate* runtime interactions between actors. Interactions among an actor, its player-centric social context model, and the relevant domain-centric social context models are *loosely* coupled and use a messaging style. A *(runtime) social context* acts as a message *router* that (1) receives messages from an actor, (2) evaluates conditions specified in associated relationships, and (3) passes the messages to another actor or a social context (as a player) or notifies the actor(s) in case of any condition violation. For the PCSC, all the incoming and outgoing messages are intercepted by the *coordinator role* which is played by the actor's *coordination application.* The application coordinates messages on behalf of the actor based on her preferences.

SCaaS facilitates the runtime realization of social contexts for supporting mediated interactions between collaborative actors, such as those of Car#1 and Car#2. However, the *requirements* of such applications are subject to continuous *change.* Thus, social contexts need to be *managed* and *adapted* to ensure the proper functioning and evolution of the applications in which the social contexts play a part. In the rest of this section, we discuss the management and adaptation support provided by the SCaaS platform in offering social context as a service.

4.2 Types of Changes

In general, there are two types of changes that require adaptation in a social context as well as across social contexts.

Changes in Environments. During convoy, it may start to rain heavily or the cars may move from one jurisdiction to another which operates different traffic management systems. Such changes cause social context adaptation and are referred to as *changes in environments.*

Changes in Requirements. A third vehicle could join when the convoy is already on the way; a broken-down following car might leave the convoy before reaching its destination; or the leading car might have a mechanical problem which requires to handover the leading car role to one of the following cars (assuming multiple following cars). Such situations also cause adaptation and are referred to as *changes in requirements.*

4.3 Adaptations in a Social Context

Runtime adaptation, often called *dynamic* adaptation, is a widely used term and is extensively studied in multiple disciplines. In pervasive computing, this term is used to denote any kind of modification at the running phase of the system [19]. In general, such runtime adaptation can be classified into two categories: *parameter* adaptation and *structure (compositional)* adaptation [18]. The adaptation in a social context to cope with the changes in user requirements and environments also can be of these two types: structural (compositional) and parametric.

We achieve *structural* adaptation in two ways:

- *Modifying topology* – Adding and removing social roles, players and relationships are carried out. For instance, a third vehicle could join the convoy when it is already on the way, or a broken-down following car leaves the convoy before reaching the destination. These situations lead to the addition or removal of roles, players and relationships in the *ConvoyDCSC.*
- *Modifying the binding between a social role and its player* – The same social role can be played by different players at different times. The *binding* between the role and the players is dynamic. For instance, in the *AVISDCSC*, the traffic management role can be played by different traffic management systems in the convoy at different times as the vehicle moves from one jurisdiction to another. Also due to a mechanical problem of the leading car (Car#1), the *Car#1PCSC* needs to unbind from the leading car role in the *ConvoyDCSC* and one of the following cars can be assigned to play the leading car role by binding that car's PCSC to the leading car role in the *ConvoyDCSC.*

We achieve *parametric* adaptation through the *modifying relationships* where adding, removing or updating *interactions*, *obligations*, *conversations* and *operational parameters* are carried out. For instance, in the *ConvoyDCSC*, the *maxDistance* parameter value in $R4$ relationship may be required to be reduced, from 1000m to 600m because of heavy rain.

Management Operations and Adaptation Rules. The organizer role provides the management capability of a social context. The principle of separation between a role and its player is also applied to the organizer role. The organizer role is internal to a social context and allows its player to manage both the *structure* and *parameters* of the social context. The organizer role presents management rights over a runtime social context model, for example, to an actor who owns the runtime model. The organizer role exposes a management interface that contains methods for *manipulating* the structure and parameter of the runtime social context model such as *addRole, deleteRole, addRelationship (addRel), deleteRelationship, addInteraction, deleteInteraction, addConversation, deleteConversation, addObligation, deleteObligation, addOperationalParameter, deleteOperationalParameter, bindRolePlayer*, and *unbindRolePlayer*.

By playing the organizer role, a *human* can perform adaptation manually using a graphical interface. On the other hand, automatic adaptation can be defined and performed through a *computer program* or an *agent* or a set of predefined adaptation *rules* as a player of organizer. For example, the *maxDistance* parameter value can be reset to 600m using the following Event-Condition-Action (ECA) rule:

```
adaptation-rule "Update maximum distance"
when
 EnvironmentChangeEvent(name=="RainingStatusValueChanged") //Event
then
 if(rainingStatus == HEAVY_RAIN) //Condition
 callMethodInOrgInterface("updateOperationalParam("R4",maxDistance,600m)")
```

Social Context States and Safe Change. To perform adaptation in a safe manner without affecting the message flow and loss of messages, we maintain the state of each entity, i.e., social role, relationship and runtime social context as a whole. Figure 4 shows the states and their transitions. When a social context is deployed all of its entities enter into *Idle* state. When a conversation is started, the associated social roles and relationship move to the *Active* state and remain there until the conversation completes. A social context enters into *Active* state when any of its social roles or relationships becomes *Active* and remains there until all of its roles and relationships become *Idle*. When an adaptation operation (structural or parametric) starts, the entity enters into the *Reconfiguration* state.

The time when a change cannot be made is the moment when the entity is in *Active* state. For instance, an adaptation operation cannot be performed in a social role or relationship when a conversation (request/response) associated with these entities is in progress. In that case, the adaptation request will be buffered and executed in the future after the entities enter into the *Idle* state.

4.4 Adaptations across Social Contexts

As stated in the previous section, a PCSC provides a coordinated view of all the interactions of an individual across different domains, and an individual plays

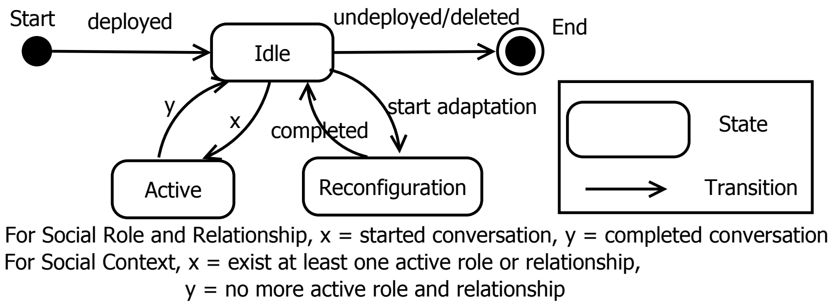

Start
deployed
Idle
undeployed/deleted
End
y
x
completed
start adaptation
Active
Reconfiguration
State
Transition
For Social Role and Relationship, x = started conversation, y = completed conversation
For Social Context, x = exist at least one active role or relationship,
y = no more active role and relationship

Fig. 4. State machine for *Social Role*, *Social Relationship* and *Social Context*

roles in multiple domains/DCSCs through her PCSC. Thus, an adaptation in a DCSC should be propagated to its corresponding PCSCs. Figure 5 shows a basic protocol for such propagation. In this protocol, the DCSC organizer triggers the adaptation in a PCSC by invoking the following methods: *triggerRoleAcquisition*, *triggerRoleRelinquishment* and *triggerUpdateRelationship*.

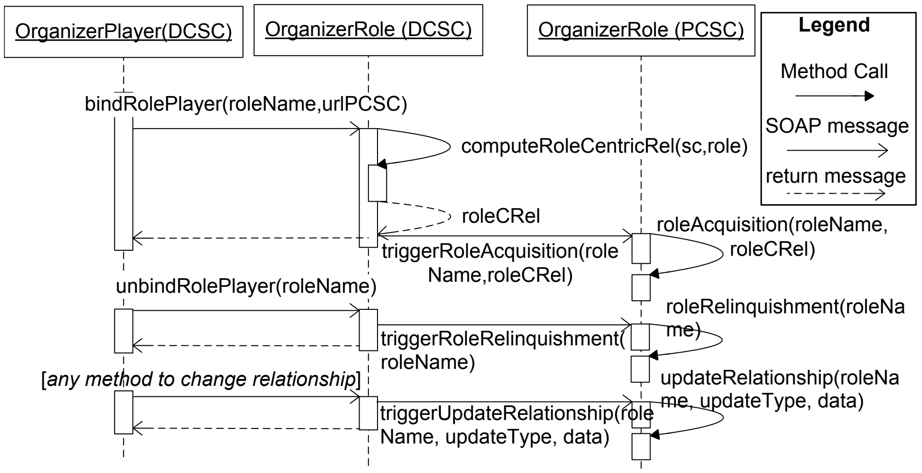

OrganizerPlayer(DCSC)
OrganizerRole (DCSC)
OrganizerRole (PCSC)
Legend
Method Call
SOAP message
return message
bindRolePlayer(roleName,urlPCSC)
computeRoleCentricRel(sc,role)
roleCRel
roleAcquisition(roleName, roleCRel)
triggerRoleAcquisition(roleName,roleCRel)
unbindRolePlayer(roleName)
roleRelinquishment(roleName)
triggerRoleRelinquishment(roleName)
[any method to change relationship]
updateRelationship(roleName, updateType, data)
triggerUpdateRelationship(roleName, updateType, data)

Fig. 5. Cross-DCSC/PCSC adaptation propagation

The adaptations across social contexts have three aspects:

1. *Binding a player to a social role in the DCSC* – When a player binds to a role in a DCSC, her PCSC should add that role and its role-centric relationship. Thus, for the *bindRolePlayer* request, the DCSC invokes *computeRoleCentricRel* method to compute the role-centric relationship of a particular social role which is the aggregation of all the relationships associated with that

Algorithm 1. Computing Role-centric Relationship

```
1: procedure COMPUTEROLECENTRICREL($sc, r$)    ▷ $r$ is a social role and $sc$ is a social context
2:     $roleCentricRel \leftarrow empty$    ▷ create an empty relationship
3:     $relList \leftarrow getAllRel4Role(sc, r)$    ▷ relationships connected to $r$
4:     for all $rel \in relList$ do    ▷ $rel$ is a relationship in the relList
5:         for all $i \in rel$ do    ▷ $i$ is an interaction in $rel$
6:             $roleCentricRel.addInteraction(i)$
7:         end for
8:         for all $c \in rel$ do    ▷ $c$ is a conversation in $rel$
9:             $roleCentricRel.addConversation(c)$
10:        end for
11:        for all $o \in rel$ do    ▷ $o$ is an obligation in $rel$
12:            $roleCentricRel.addObligation(o)$
13:        end for
14:        for all $p \in rel$ do    ▷ $p$ is an operational parameter in $rel$
15:            $roleCentricRel.addOperationalParam(p)$
16:        end for
17:    end for
18:    return $roleCentricRel$
19: end procedure
```

social role in the DCSC (see Algorithm 1). Then the DCSC invokes the *triggerRoleAcquisition* method in the PCSC organizer with the social role and its role-centric relationship, as parameters. The PCSC organizer executes *roleAcquisition* method to adapt its structure by adding a social role and relationship based on the received information.

2. *Unbinding a player from a role in the DCSC* – When a player is unbound from the DCSC, her PCSC should be adapted by removing that role and its role-centric relationship. Thus, for the *unbindRolePlayer* request the DCSC invokes the *triggerRoleRelinquishment* method with the social role name as the parameter. The PCSC organizer executes *roleRelinquishment* method to adapt its structure by deleting the social role, and the relationship between that social role and the coordinator role, when those entities are in *Idle* state (i.e., safe to delete).
3. *Updating a relationship in the DCSC* – All the updates in a relationship and/or a social role in a DCSC should be propagated to the corresponding PCSC(s). Thus, for any modification request in a relationship (i.e., to add, delete or update an interaction, conversation or obligation), the DCSC organizer invokes the *triggerUpdateRelationship* method in the PCSCs which are bound to the associated social roles in that relationship. Then the PCSC organizer executes the *updateRelationship* method to reflect the changes.

4.5 Revisiting the Scenario

Let us consider a situation that requires management and adaptation in a social context and across social contexts which can be addressed using the SCaaS platform.

If the leading car (Car#1) breaks-down, the *ConvoyDCSC* organizer player (the leading car driver) invokes (through a user interface) the *unbindRolePlayer* method to unbind *Car#1PCSC* from the LC and then the *ConvoyDCSC* (organizer) invokes the *triggerRoleRelinquishment("LC")* method in *Car#1PCSC*. As a result, the *Car#1PCSC* updates its structure by deleting the LC role and the relationship between the LC and Coordinator role (see Fig. 6ⓐ). Furthermore, to assign a following car (say Car#2) to play the leading car role, the *ConvoyDCSC* organizer player invokes the *unbindRolePlayer("FC", urlCar#2PCSC)* followed by *bindRolePlayer("LC",urlCar#2PCSC)* to first unbind *Car#2PCSC* from the following car role and then bind it to the leading car role. As a result, the *ConvoyDCSC* organizer invokes the *triggerRoleRelinquishment* and *triggerRoleAcquisition* methods respectively which ultimately updates the *Car#2PCSC* by deleting the FC role and its associated relationship (see Fig. 6ⓐ) followed by adding the LC role and its associated relationship (see Fig. 6ⓑ).

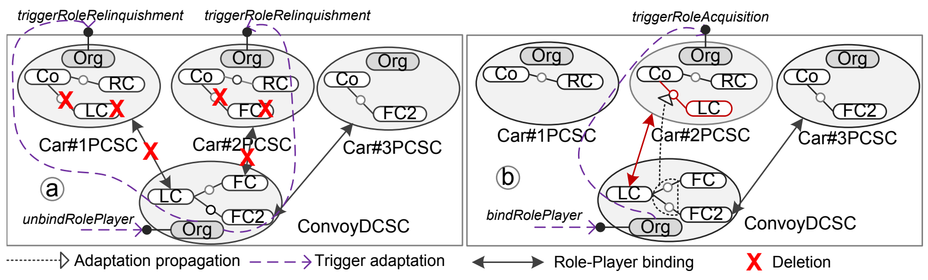


Fig. 6. Runtime adaptation in social context models due to break-down of leading car

5 Prototype Implementation

We have implemented the SCaaS platform by adopting and extending the ROAD4WS [13] which is an extension to the Apache Axis2[1] web service engine for deploying adaptive service compositions. The SCaaS platform exploits JAXB 2.0[2] for creating DCSCs and PCSCs runtime from their XML descriptors. JAXB helps the generation of classes and interfaces of runtime models automatically using an XML schema. The platform exposes each *social role* as a *service*, the associated *interactions* of the role as operations of that service.

[1] `http://axis.apache.org/`
[2] `http://jcp.org/en/jsr/detail?id=22`

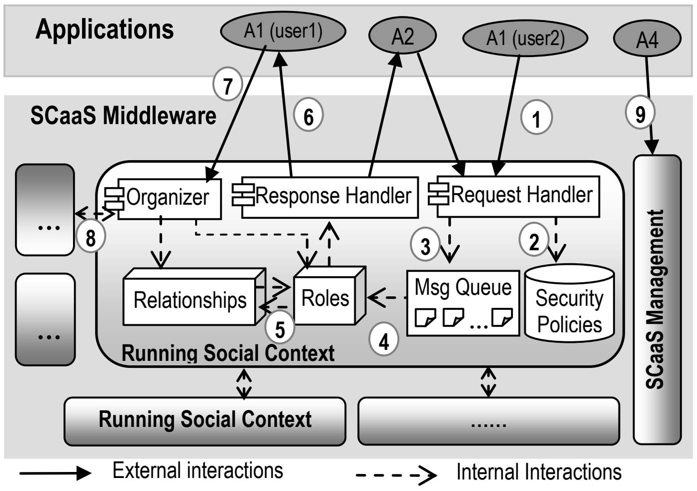


Fig. 7. The SCaaS platform architecture

The *conversation* and *obligations* specified in the *social relationship* are evaluated as *event-condition-action* rules and implemented using Drools[3]. Actors (players) playing the roles invoke the operations which create messages. Such messages are routed to other players who they are collaborating with. As illustrated in Fig. 7, the SCaaS platform generates *runtime social contexts* which are able to (1) handle requests received from players (i.e., applications); (2) check security settings for authorized access; (3) allocate requests into a message queue; (4) forward messages to corresponding social roles; (5) evaluate conditions specified in the relationships; (6) send requests to relevant players. A runtime social context also can (7) receive a management request (from a user/application) and adapt itself accordingly, and (8) propagate the adaptation to other social contexts as necessary. The SCaaS Management module handles (9) platform level management requests such as create, delete, deploy and undeploy social contexts as required by the user/application. Interactions between the runtime models and their players (i.e., external interactions) are supported by exchanging SOAP[4] messages.

The *runtime adaptations* are supported by the Java reflection mechanism and the Drools engine. To cope with the changes in environments and requirements, at runtime, Javassist[5] allows generation of *new* classes and *modification* of existing classes, which helps to add new social roles/relationships and change existing roles/relationships, respectively. Drools engine allows the SCaaS to inject new rules and delete existing rules from the working memory which facilitates the addition and deletion of conversations, obligations and parameters in the

[3] `http://www.jboss.org/drools/`
[4] `http://www.w3.org/TR/soap/`
[5] `http://www.jboss.org/javassist`

relationships. This way of implementation provides flexible and easy runtime adaptation in a particular entity of a social context without interrupting the other entities of that social context.

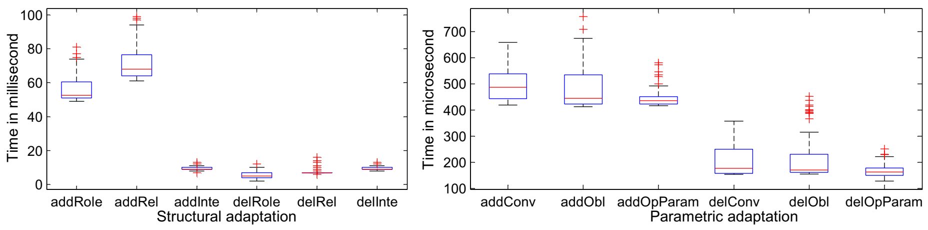
6 Experimental Evaluation and Case Study

The goal of our experiment had to quantify the SCaaS platform's *adaptation overhead*. We installed the SCaaS platform on a machine with Core i3 2.2 GHz CPU, 8GB RAM and Windows 7 OS. We used Java 1.6, Drools 2, Tomcat 7.0.21 and Axis2 1.6.1 in this experiment. As a case study, we also implemented the motivating scenario as a proof-of-concept application, and measured the application's adaptation overhead in real-life experiment to demonstrate the feasibility and applicability of the SCaaS-based application.

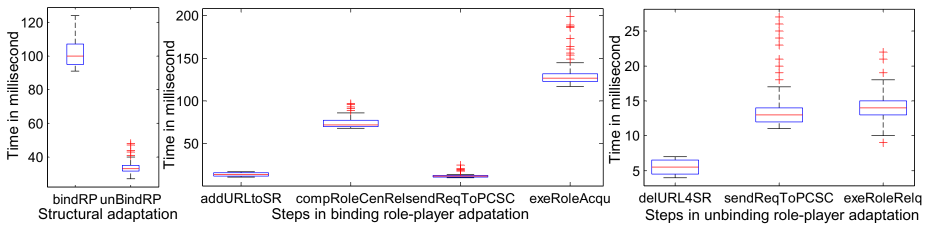
6.1 Adaptation Overhead

To evaluate the adaptation overhead, we deployed 100 DCSCs where each DCSC is consisted of 10 roles connected in a ring topology using 10 relationships, and each relationship is comprised 6 interactions, 6 conversations and 6 obligations. Once we deployed 100 DCSCs to the SCaaS platform, SCaaS created 10 PCSCs for 10 players where each player plays a role in each of the 100 DCSCs. Thus, each PCSC contained 100 social roles and 100 role-centric relationships. We executed each of the structural and parametric adaptation operations (e.g, addRole, addConv) 1000 times over the 100 DCSCs. We measured the time from the moment the adaptation was requested, to the moment Axis2 updated the services. The box plots in Fig. 8 show the summary of the results where the horizontal line inside each of the boxes represents the median (average time). The results show that the deletion operations (e.g., *delRole*, *delRel*, and *unbindRP*) take less time compared to the addition operations (i.e., *addRole*, *addRel*, and *bindRP*). Figure 8a shows the time required to perform different structural and parametric adaptations in a social context. The results show that the structural adaptations take more time than the parametric adaptations. Among the structural adaptation operations, adding a relationship (*addRel*) in a social context takes the longest time, around 68 millisecond (ms) (on average), as it needs to update the configuration of two social roles, where as deleting a social role (*delRole*) takes the least time, around 5ms. For different parametric adaptation operations, the required time is related to rule injections and deletions in the Drools engine and lies between 162 and 486 microseconds.

The box plots in Fig. 8b illustrate the adaptation overhead results across a DCSC and a PCSC. The leftmost figure shows the total time required for the *bindRP* (bind role-player) and *unbindRP* (unbind role-player) structural adaptations. The middle figure shows the time required for each step in the *bindRP* adaptation, including the time to add a URL to a social role (*addURLtoSR*), to compute a role centric relationship (*compRoleCenRel*) using Algorithm 1, to send a request to a PCSC (*sendReqToPCSC*), and to execute the role acquisition method (*exeRoleAcq*). The rightmost figure shows the time required for each



(a) Adaptation in a social context



(b) Adaptation across social contexts

Fig. 8. Adaptation overhead

step in the *unbindRP* adaptation, including the time to delete an URL from a social role (*delURL4SR*), to send a request to a PCSC (*sendReqToPCSC*), and to execute the role relinquishment method (*exeRoleRelq*). The results show that on average the *bindRP* and *unbindRP* operations take 100ms and 33ms, respectively, which we believe is an acceptable overhead in collaborative applications.

6.2 Case Study

To demonstrate the real-world applicability and feasibility of our approach, we have developed an adaptive collaborative application on top of the SCaaS platform, called *SocioTelematics* that enables multiple cars to form cooperative convoys. This application allows the drivers to see each other's positions on the Google Maps. Using this application, drivers in the convoy can adapt and manage their social contexts and interactions. The SCaaS makes it easy to develop this application based on their supposed interactions and without worrying about the underlying message communication and the evaluation of the messages, as the runtime support and adaptation of these social contexts and interactions are externalized to and managed by the SCaaS platform. Moreover, the runtime adaptation capability provided by the SCaaS platform allows the application to respond to changes in requirements and environmental factors, without any change in the application code.

We evaluated the application's adaptation overhead using two cars in a cooperative convoy over 50 kilometres of driving where the SCaaS platform was deployed in the Amazon EC2 and the two client *SocioTelematics* applications were running on two in-car Android Samsung Galaxy Tabs with 3G connections. The results in Table 2 show that given the 1.12 second communication latency

Table 2. Time required to perform adaptation operations

Operations	Time
Send an adaptation request from the *SocioTelematics* application to the Amazon server over a 3G network	1.121 sec
Add a new car to the *ConvoyDCSC* at runtime, i.e., *addRole*, *addRel* and *bindRP*	0.209 sec
Remove a car from the *ConvoyDCSC* at runtime, i.e., *unbindRP*, *delRel* and *delRole*	0.046 sec
Change the $o1$ operational parameter in $R4$ relationship	0.422ms

between the application and the server, on average the time to add and remove a car to and from the convoy at runtime take 1.33 sec (i.e., 1.121+0.209) and 1.167 sec, respectively, which we believe are acceptable times in a cooperative convoy.

7 Related Work

7.1 Platform for Collaborative Pervasive Applications

The need for supporting collaboration in pervasive computing environments has emerged in recent years (e.g., [4], [8]). Such research has focused on collaborative interactions between different types of actors such as user-user and device-device, for various purposes. The SAPERE [4] middleware exploits social network graph to establish collaboration for sharing data among spatially collocated users devices. CoCA [5] is a ontology-based context-aware service platform for sharing of computational resources among devices in a neighbourhood. Both SAPERE and CoCA focus on collaboration among *devices*, where SCaaS focuses on collaboration among *users*. Similar to SCaaS, MoCA [6], a middleware architecture for developing context-aware collaborative applications, focuses on collaboration among *users*. But unlike SCaaS, the collaborations among users in MoCA are not based on *predefined* goals or tasks, rather driven by spontaneous and occasional initiatives. CASMAS [7] and UseNet [8] focus on collaborative activities among users to achieve a common goal like SCaaS. But none of them explicitly *model* the interactions among users.

Moreover, all of the above approaches lack support for *managing* the dynamicity and complexity of the social context as highlighted in this paper. The relationships between actors and their adaptations are not modeled explicitly, and instead are often *hard-coded* directly into the applications. To the best of our knowledge, there is no work to date that addresses the runtime adaptation of social context models in response to the changes in requirements and environments.

7.2 Middleware Support for Runtime Adaptation

Much research has been carried out into middleware support for runtime adaptation in context-aware systems (e.g., MADAM [9] and 3PC [10]) and service-oriented systems (e.g., MUSIC [11] and MOSES [20]). These middleware solutions mainly target the tasks of individual users/applications and have focused on reconfiguring applications' settings (rather than interaction relationships) based on physical context information (e.g., place, time)/quality of service requirements (e.g., performance, reliability), rather than interaction relationships. Moreover, their proposed runtime models are application-specific and cannot be used to model interaction-relationships among collaborative actors.

In contrast to these solutions, the SCaaS platform targets collaborative pervasive applications, and focuses on executing adaptation by explicitly realizing interaction relationships using social contexts and providing an organizer interface to change such social contexts. On the other hand, SCaaS does not address the monitoring of environment changes (i.e., physical context information), analyzing such information or making adaptation decisions. In that sense, the SCaaS middleware is not a substitute for existing middleware solutions that manages physical context information, rather can be built on top of those solutions as appropriate, in order to manage (as a service) social interactions and context adaptation for collaborative pervasive applications.

8 Conclusion

We have presented a novel *Social Context as a Service* platform for supporting application-level adaptations and enabling mediated-interactions among actors (individuals with their applications) in collaborative pervasive environments. Our approach *externalizes* interaction-relationships from the application implementation, explicitly *models* the interactions in terms of social contexts, *separates* functional interactions from management operations, and provides runtime realization of social contexts. All these facilitate the systematic *management* of dynamic interaction-relationships between actors and support their *adaptation* to cope with the *changes* in user requirements and environments.

SCaaS facilitates both *structural* and *parametric* adaptations in social contexts which are realized through the *management* (organizer) interface of the social contexts. SCaaS also maintains the inherent *dependencies* among social contexts and keeps them consistent through coordinated *cross-social context adaptation*. Our model-driven approach and service-oriented implementation make it easier to develop different adaptive collaborative applications on top of SCaaS. We have quantified the adaptation overhead of the SCaaS platform through an experimental evaluation and demonstrated its applicability with a cooperative convoy telematics application.

References

1. Conti, M., et al.: Looking ahead in pervasive computing: Challenges and opportunities in the era of cyber-physical convergence. Pervasive Mob. Comput. 8(1), 2–21 (2012)

2. Sancho, G., et al.: What about collaboration in ubiquitous environments? In: Proc. of 10th Int. Conf. on New Techn. of Distr. Syst., NOTERE (2010)
3. Endler, M., et al.: Defining Situated Social Context for pervasive social computing. In: Proc. of PerCom Workshop (2011)
4. Castelli, G., Rosi, A., Zambonelli, F.: Design and implementation of a socially-enhanced pervasive middleware. In: Proc. of PerCom Workshop, pp. 137–142 (2012)
5. Ejigu, D., et al.: CoCA: A Collaborative Context-Aware Service Platform for Pervasive Computing. In: Proc. of ITNG, pp. 297–302 (2007)
6. Sacramento, V., et al.: MoCA: A Middleware for Developing Collaborative Applications for Mobile Users. IEEE Distributed Systems (2004)
7. Cabitza, F., et al.: CASMAS: Supporting Collaboration in Pervasive Environments. In: Proc. of PerCom, pp. 286–295 (2006)
8. Rodriguez, I.B., et al.: A model-driven adaptive approach for collaborative ubiquitous systems. In: Proc. of the AUPC Workshop (2009)
9. Geihs, K., et al.: A comprehensive solution for application-level adaptation. Softw. Pract. Exper. 39(4) (2009)
10. Handte, M., et al.: 3PC: System support for adaptive peer-to-peer pervasive computing. ACM Trans. Auton. Adapt. Syst. 7(1) (2012)
11. Rouvoy, R., et al.: MUSIC: Middleware Support for Self-Adaptation in Ubiquitous and Service-Oriented Environments. In: Cheng, B.H.C., de Lemos, R., Giese, H., Inverardi, P., Magee, J. (eds.) Software Engineering for Self-Adaptive Systems. LNCS, vol. 5525, pp. 164–182. Springer, Heidelberg (2009)
12. Kabir, M.A., Han, J., Colman, A.: Modeling and Coordinating Social Interactions in Pervasive Environments. In: Proc. of 16th Int. Conf. on Eng. of Complex Compu. Syst. (ICECCS), pp. 243–252 (2011)
13. Kapurge, M., Colman, A., King, J.: ROAD4WS - Extending Apache Axis2 for Adaptive Service Composition. In: Proc. of the EDOC (2011)
14. Raychoudhury, V., et al.: Middleware for pervasive computing: A survey. Pervasive and Mob. Comput. (2012)
15. Papazoglou, M.P., van den Heuvel, W.-J.: Service-Oriented Architectures: Approaches, Technologies and Research Issues. VLDB J. 16(3) (2007)
16. Colman, A., Han, J.: Roles, Players and Adaptive Organisations. Applied Ontology: An Interdisciplinary Journal of Ontological Analysis and Conceptual Modeling 2, 105–126
17. Kabir, M.A., et al.: SocioTelematics: Leveraging Interaction-Relationships in Developing Telematics Systems to Support Cooperative Convoys. In: Proc. of Ubiquitous Intelligence & Comput., pp. 40–47 (2012)
18. McKinley, P.K., Sadjadi, S.M., Kasten, E.P., Cheng, B.H.C.: Composing adaptive software. Computer 37, 56–64 (2004)
19. Kakousis, K., Paspallis, N., Papadopoulos, G.A.: A survey of software adaptation in mobile and ubiquitous computing. Enterp. Inf. Syst. 4 (2010)
20. Cardellini, V., et al.: MOSES: A Framework for QoS Driven Runtime Adaptation of Service-Oriented Systems. IEEE Transactions on Software Engineering 38, 1138–1159 (2012)

Simulation-Based Modeling and Evaluation of Incentive Schemes in Crowdsourcing Environments

Ognjen Scekic, Christoph Dorn, and Schahram Dustdar

Distributed Systems Group, Vienna University of Technology
{oscekic,dorn,dustdar}@dsg.tuwien.ac.at
http://www.dsg.tuwien.ac.at

Abstract. Conventional incentive mechanisms were designed for business environments involving static business processes and a limited number of actors. They are not easily applicable to crowdsourcing and other social computing platforms, characterized by dynamic collaboration patterns and high numbers of actors, because the effects of incentives in these environments are often unforeseen and more costly than in a well-controlled environment of a traditional company.

In this paper we investigate how to design and calibrate incentive schemes for crowdsourcing processes by simulating joint effects of a combination of different participation and incentive mechanisms applied to a working crowd. More specifically, we present a simulation model of incentive schemes and evaluate it on a relevant real-world scenario. We show how the model is used to simulate different compositions of incentive mechanisms and model parameters, and how these choices influence the costs on the system provider side and the number of malicious workers.

Keywords: rewards, incentives, crowdsourcing, social computing, collective adaptive systems.

1 Introduction

Research on incentives in crowdsourcing systems has been increasingly attracting interest recently (e.g., [19,11,15,9]). Today's commercial crowdsourcing systems mostly deal with simple tasks and lack worker interactions and dependencies. Such collaborative patterns in many ways resemble traditional piece-work, enabling use of conventional pay-for-performance incentive mechanisms [16]. These existing incentive mechanisms are based upon statistical models (e.g., *agency theory*) that take into consideration workers engaging in contractual, long-term relationships with a traditional company and seeking to maximize their utility metrics (see Section 2). However, as social computing systems grow more complex (e.g., Collective Adaptive Systems[1]) the web scale and unstable nature of crowd worker interactions with the system makes the use of traditional incentives unpredictable and inappropriate.

[1] http://focas.eu/

Conventional incentive models completely disregard social characteristics of the crowd, such as coordinated group actions, social/regional/ethnic peculiaritites, voluntary work [6], importance of reputation/flaunting [18], or web-scale malicious behavior [21]. An additional complication is that these phenomena change often and characterize different subsets of the crowd differently in different moments. This makes development of appropriate mathematical incentive models difficult. Specifically, the root cause lies in insufficient understanding of the implications arising from a particular combination of worker participation patterns and applied incentive schemes.

The system designer needs to consider additional factors, such as: emerging, unexpected and malicious worker behavior, incentive applicability, range of stability, reward fairness, expected costs, reward values and timing. Failing to do so leads to exploding costs and work overload, as the system cannot scale with the extent of user participation. Unbalanced rewards keep new members from joining or cause established members to feel unappreciated and leave. Ill-conceived incentives allow users to game the system, prove ineffective against vandalism, or assign too many privileges to particular members tempting them to abuse their power [13].

This calls for a systematic approach in designing and evaluating incentive schemes before deployment on real-world social computing systems. In [16] we surveyed existing incentive practices in traditional companies and different crowdsourcing platforms and illustrated the shortcomings of applying conventional incentive mechanisms in crowdsourcing environments. We then proposed how to combine proven atomic incentive mechanisms into scalable and portable incentive schemes suitable for social computing systems. Based on these conclusions, in [15,17] we presented a model and a system capable of deploying and executing such mechanisms. Continuing on this line of research in this paper we now investigate how to select, customize and evaluate appropriate atomic incentive mechanisms and how to compose them for a given crowdsourcing scenario. Specifically, we propose modeling and simulating various participation options available to workers, activities and costs on the system provider side, how user actions are transformed into rewards, and how these rewards in turn influence user behavior. Further justification for this approach is presented in Section 2.

The contributions of this paper are:

1. Abstract simulation model of incentive mechanisms for crowdsourcing (Section 3).
2. Concrete incentive model for a real-world crowdsourcing scenario based on 1.
3. Complete modeling and simulation methodology, detailing the implementation and evaluation processes to design a concrete incentive model such as 2. (Section 5)

The validity and capabilities of the model and the methodology are evaluated through a relevant simulation scenario.

The remainder of this paper is structured as follows. Section 2 provides a discussion on related work and our previous work. Section 3 provides an overview of our approach, the abstract incentive model and the simulation rationale. Section 4 presents two relevant scenarios that we use as the environment to demonstrate and evaluate the methodology. We discuss the methodology and concrete design decisions in Section 5. A scenario case-study in Section 6 demonstrates the simulation's usefulness to provide insights into behavior and effectiveness of selected incentive mechanisms and other design decisions. Section 7 gives an outlook on future work and concludes this paper.

2 Background and Related Work

Previous research on incentives can be roughly categorized in two groups. One group seeks to find optimal incentives in formally defined environments through precise mathematical models (e.g., principal-agent theory [8,2], game theory [4,7]). Both the agent (worker) and the authority (employer) are seen as entities deciding on their actions with the exclusive goal of maximizing gain or achieving a predefined goal. Although successfully used in microeconomic models, these incentive models do not fully capture the diversity and unpredictability of human behavior that becomes accentuated in a crowdsourcing environment. These disadvantages (see [11]) prompted the advent of another direction in research of incentives in crowdsourcing.

The other group examines the effects of incentives by running experiments on existing crowdsourcing platforms and rewarding real human subjects with actual monetary rewards (e.g., [9,11]). The major disadvantages of this approach are its high cost and duration. Furthermore, although seemingly yielding realistic findings, there is evidence that the low amounts of monetary rewards used in these experiments make the findings applicable only for a very limited range of simple activities, such as image tagging and text translation. These and other shortcomings that this type of research suffers from are listed in [1].

In contrast to these two approaches, our intention is not to devise novel nor optimal incentive mechanisms for crowdsourcing, but rather to offer system designers a methodology for quickly selecting, composing and customizing existing, real-world atomic incentive mechanisms [16,19], and roughly predicting the effects of their composition in dynamic crowdsourcing environments. The model and simulation parameters can be changed dynamically, allowing quick testing of different incentive scheme setups and behavioral responses at low cost. The schemes can then be deployed on systems such as [15,17] and provided as a service to the third parties.

Our simulation approach allows modeling of incentives and responses of workers of arbitrary complexity. Specifically, we employ principles of agent-based social simulation [10,5], an effective and inexpensive scientific method for investigating behavioral responses of large sets of human subjects. As we are primarily interested in investigating how reputation affects (malicious) behavior, we characterize each agent by reputation metric, as laboratory experiments confirmed that reputation promotes desirable behavior in a variety of different experimental settings [20,12,14,18].

However, unlike the usual approach where agents interact directly (and thus benefit from cooperative behavior or suffer from defective behavior), we introduce a provider that facilitates interactions and determines the benefits or costs of those interactions. Reputation allows the provider to assess an agents reliability. Consequently, pure reputation sharing alone is insufficient. We require additional incentive mechanisms to obtain cooperative behavior beyond the users intrinsic level. Further differences to the conventional agent-based simulation include the explicit, detailed modeling of the underlying collaboration patterns, thereby building upon our previous work [3].

3 Incentive Mechanisms for Crowdsourcing Processes

Any incentive mechanism in general involves two interested parties - an *authority* and a *worker* (actor, agent). The authority is interested in stimulating, promoting or discouraging certain behavioral responses in workers. The incentive exhibits its psychological effect by promising the worker a reward or a punishment based on the actions the worker will perform. The wish to get the reward or escape the punishment drives the worker's decisions on future actions. The reward (punishment) can be material or psychological (e.g., a change of status in a community – ranking, promotion). The type, timings and amounts of reward need to be carefully considered to achieve the wanted effect of influencing a specific behavior in a planned direction. In addition, introduction of incentives introduces additional costs for the authority who hopes to compensate for them through the newly arisen worker behavior (e.g., increased productivity).

However, as soon as an incentive mechanism is introduced, it produces dysfunctional behavioral responses in the worker population. The workers adapt to the new rules and change their working patterns, often in unpredictable or even malicious ways, trying to misuse the new incentive to profit more than the rest of the population [13]. The authority compensates for this by introducing other incentive mechanisms targeting the dysfunctional behavior, further increasing the authority-side costs, and causing new types of dysfunctional behavior. However, once the proper combination of incentive mechanisms is put in place and calibrated, the system enters a stable state. The problem with the crowdsourcing processes is that the system may not stay long in a stable state due to an unforeseen change in worker participation or collaboration pattern. Therefore, the incentive setup needs to be reconfigured and re-calibrated as quickly as possible, in order to avoid incurring high costs to the authority. This feedback control-loop involving the authority and the worker represents the actual incentive mechanism that we model and simulate in this paper.

Modeling an incentive mechanism, therefore, always involves modeling both the authority and the worker side, as well as the possible interactions between them. In Figure 1 we show an abstract representation of the model of incentive mechanism that we implement in the following sections.

Workers differentiate from each other by having different sets of personal characteristics (e.g., accuracy, speed, experience). The characteristics are determined by a private set of variables stored in the *internal state* S. Usually, the variables are normally distributed across the worker population, although particular variables can be intentionally given predefined values to provoke a certain type of behavior. The internal state also contains records of worker's past actions. The internal state is private to the worker, and is used as one of the inputs for the *decision-making function* f_a that chooses the next action to perform.

Apart from the internal state, each worker is characterized by the publicly exposed set of *performance metrics* M that are defined and constantly updated by the authority for each worker. The performance metrics reflect the authority's perception of the worker's past interactions with the system (e.g., trust, rank, expertise, responsiveness). Knowing this allows the worker to decide better on his future actions. For example, knowing that a poor reputation will disqualify him from getting a reward in future may drive the worker to work better or to quit the system altogether. It also allows him to compare

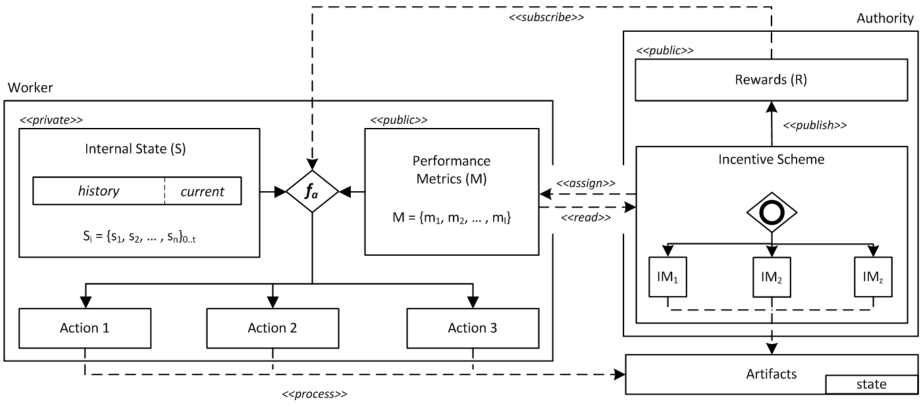


Fig. 1. Incentive mechanisms need to capture the interaction between worker and authority

with other workers. Therefore, the set of performance metrics is another input for the decision-making function f_a.

The third input for the decision-making function f_a is the set of promised *rewards (punishments) R*. Rewards are expressed as publicly advertised amounts/increments in certain parameters that serve as the recognized means of payment/prestige within the system (e.g., money, points, stakes/shares, badges). They are specified per action per artifact and per performance metrics, thus making them also dependent per user. For example, a reward may promise an increase of at least 100 points to any/first user who performs the action of rating an artifact. The amount of points can then be further increased or decreased depending on the user's reputation.

Worker interacts with the authority solely by performing actions over *artifacts* (K) offered to the worker population by the authority. Worker's behavior can thus be described as a sequence of actions in time, interleaved with periods of idling (idling being a special-case of action). The set of possible actions is the same for every worker. However, the effects of the execution of an action may be different, depending on the worker's personal characteristics from the internal state S. For example, a worker with innate precision and bigger experience can improve an artifact better than the worker not possessing those qualities.

As previously stated, worker's next action is selected through the use of a decision-making function $f_a = f(S, M, R)$ potentially considering all of the following factors: a) the statistically or intentionally determined personality of the worker; b) historical record of past actions; c) authority's view of one's own performance; d) performance of other workers; and e) promised rewards, with respect to the current state of one's performance metrics. The decision-making function is arbitrarily defined by the system designer. For example, we can use a utility-maximization function, as described in the papers cited in Section 2.

The authority's motivation for offering artifacts for processing to the crowd is to exploit the crowd's numerosity to either achieve higher quality of the artifacts (e.g., in terms of accuracy, relevance, creativity), or lower the cost (e.g., in terms of time or money). This motivation guides the authority's choice of incentive mechanisms. Authority has at its disposal a number of *incentive mechanisms* IM_i. Each one of them should be designed to target/modify only a small number of very specific parameters (see later). Thus, it is the proper addition or composition of incentive mechanisms that allows the overall effect of an incentive scheme, as well as fine-tuning and runtime modifications.

An incentive mechanism IM takes as inputs: 1) the current state of an artifact K_i; 2) the current performance metrics of a worker M_j; and optionally 3) the output from another incentive mechanism returning the same type of reward – R'_{a_k}. The output of an incentive mechanism is the amount/increment of the reward R_{a_k} to offer to the worker M_j for the action a_k over artifact K_i.

$$IM : (K_i, M_j, R'_{a_k}) \rightarrow R_{a_k} \tag{1}$$

The true power of incentive mechanisms lies in the possibility of their combination. The reward (f_R) can be calculated through a number of additions (+) and/or functional compositions ($\circ$) of different incentive mechanisms. For example, a worker may be given an increment in points for each time he worked on an artifact in the past. Each of those increments can then be modified, depending on how many other workers worked on that same artifact. In addition, the total increment in points can be further modified according to the worker's current reputation. The finally calculated increment value represents the promised reward. The set of finally calculated rewards per worker $R_w = \{f_{R_1}, ..., f_{R_z}\}$ is then advertised to the workers, influencing their future behavior, and closing the feedback loop.

The major difficulty in designing a successful incentive scheme lies in properly choosing the set of *incentive parameters* (performance metrics, incentive mechanisms, and their compositions). Often, the possible effects when using one set of parameters are unclear at design time, and an experimental or a simulation evaluation is needed to determine them. A proven set of incentive parameters is usually called an *incentive scheme*.

4 Motivating Scenarios

Here we present two relevant scenarios for which our simulation model and methodology can be used to design and evaluate appropriate incentive schemes.

Citizen-Driven Traffic Reporting. Local governments have a responsibility to provide timely information on road travel conditions. This involves spending considerable resources on managing information sources as well as maintaining communication channels with the public. Encouraging citizens to share information on road damages, accidents, rockfalls, or flooding reduces these costs while providing better geographical coverage and more up to date information[2]. Such crowdsourcing process, however, poses data quality related challenges in terms of assessing data correctness, completeness, relevance, and duplication.

[2] For a real world example visit the Aberdeen City Council's SmartJourney initiative at `http://smartjourney.co.uk/`

Crowdsourced Software Testing. Traditional software testing is a lengthy and expensive process involving teams of dedicated engineers. Software companies[3] may decide to partially crowdsource this process to cut time and costs and increase the number and accuracy of detected defects. This involves letting the remote testers detect bugs in different software modules and usage environments and submitting bug reports. Testers with different reputations provide reports of varying quality and change the assigned bug severity. As single bugs can be reported multiple times in separate reports, testers can also declare two reports as duplicates.

The two scenarios exhibit great similarities. The expected savings in time and money can in both cases be outweighed by an incorrect setup and application of incentive mechanisms. Furthermore, the system could suffer from high numbers of purposely incorrect or inaccurate bug report submissions, driving the processing costs high. For the purpose of this paper, we join and generalize the two scenarios into a single, abstract one that we will use in our simulation setup:

The *Authority* seeks to lower the time and cost of processing a large number of *Reports* on various *Situations* occurring in the interest domain of the Authority. The *Workers* are independent agents, occasionally and irregularly engaging with the system managed by the Authority to perform one of the following *Actions*: *Submit* a new Report on a Situation, *Improve* an existing Report, *Rate* the accuracy and importance of an existing Report, inform the Authority of his belief that two existing Reports should be considered *Duplicates*. The Worker actions are driven by the combination of the following factors: a) possibility to earn *Points* (translating to increased chances of exchanging them for money); b) possibility to earn *Reputation* (translating to a higher status in the community); and c) the intrinsic property of people to contribute and help or to behave maliciously. In order to influence and (de-)motivate workers, the Authority employs a number of *Incentive Mechanisms*, collectively referred to as *Incentive Scheme*.

This scenario also needs to address the following challenges:

– *Crowdsourced report assessment.* The effort required for manual validation of worker-provided reports may easily outweigh the gained effort and cost reduction from crowdsourced reporting in the first place. Hence, workers need to be properly stimulated to supplement and enrich existing reports as well as vote on their importance, thereby lifting the verification burden off the authority. The system also needs to strike a balance not to collect too much information.
– *Worker reputation (trust).* A worker's reputation serves as one potential indicator for data reliability, assuming that reputable workers are likely to provide mostly accurate information. Subsequently, reports from workers with unknown or low reputation need to undergo more thorough peer assessment. The system must support continuous adjustment of workers' reputation.
– *Adjustable and composable incentive scheme.* An effective incentive scheme needs to consider all past citizen actions, the current state of a report and the predicted costs of processing a report manually in order to decide whether and how to stimulate workers to provide additional information. It also needs to correctly identify and punish undesirable and selfish behavior (e.g., false information, deliberate duplication of reports, intentional up/downgrading of reports).

[3] For example, `www.utest.com`

The resulting complexity arising from the possible combination and configuration of worker behavior, incentive schemes, and processing costs requires a detailed analysis to identify a stable and predictable system configuration and its boundaries.

5 Modeling and Simulation Methodology

Our methodology for simulating worker participation and incentive mechanisms in crowdsourcing processes is depicted in Figure 2. It consists of four basic steps, usually performed in multiple iterations: *i)* defining a domain-specific meta-model by extending a core meta-model; *ii)* capturing worker's behavioral/participation patterns and reward calculation into an executable model; *iii)* defining scenarios, assumptions, and configurations for individual simulation runs; and *iv)* evaluating and interpreting simulation results. These steps are described in more detail below.

We use the DomainPro[4] modeling and simulation tool suite in each of the outlined methodology steps to design and instantiate executable models of incentive mechanisms and run simulations of those models. The tool allows creating custom simulation languages through metamodeling and supports agent-based and discrete event simulation semantics (see [3]). However, our overall approach is generic and can be easily applied using a different modeling and simulation environment.

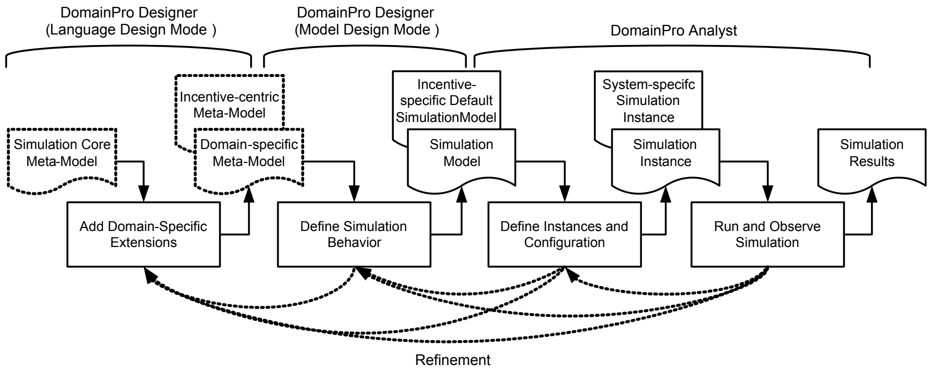

DomainPro Designer
(Language Design Mode)
DomainPro Designer
(Model Design Mode)
DomainPro Analyst
Incentive-centric Meta-Model
Simulation Core Meta-Model
Domain-specific Meta-Model
Incentive-specific Default SimulationModel
Simulation Model
System-specifc Simulation Instance
Simulation Instance
Simulation Results
Add Domain-Specific Extensions
Define Simulation Behavior
Define Instances and Configuration
Run and Observe Simulation
Refinement

Fig. 2. The methodology of simulation design and development

The simulation core meta-model is implemented in the DomainPro Modeling Language. Optional extensions result in a domain-specific meta-model that defines which component types, connector types, configuration parameters, and links a simulation model may exhibit. In our case, we extend the core meta-model to obtain what we refer to as *incentive-centric meta-model* (Section 5.1). The obtained incentive-centric meta-model serves as the basis for defining the simulation behavior, i.e., the *executable simulation model* (Section 5.2). Obtaining the executable simulation model requires defining workers' behavioral parameters, authority's business logic (including incentive

[4] `www.quandarypeak.com`. Open-source version forthcoming.

mechanisms and cost metrics), the environment and the control flow conditions between them. Finally, prior to each execution, the executable simulation model requires a quick runtime configuration in terms of the number of worker instances and monitored performance metrics (Section 6.1). During the execution, we do near real-time monitoring of metrics, and if necessary, perform simulation stepping and premature termination of the simulation run to execute model refinements.

The tool we use enables refinement at any modeling phase. A designer will typically start with simple meta- and simulation models to explore the basic system behavior. She will subsequently refine the meta-model to add, for example, configuration parameters and extend the functionality at the modeling level. This enables testing simple incentive mechanisms first, and then extending and composing them once their idiosyncrasies are well understood.

5.1 Incentive-Centric Meta-Model

The derived meta-model (Fig.3) reflects the conceptual view of incentive mechanisms as presented in Section 3. A *ParticipationPattern* consists of *Actors* (*User*s or *Providers*) and the *InteractionObjects*. Actors exhibit *Behavior* that encapsulates different *UserActivities*. InteractionObjects contain *ObjActivities* that define allowed and reward-yielding activities on an InteractionObject. *InternalSequences*, *ExternalSequence*, and *ObjectSequence* determine the control flow among activities by specifying trigger conditions. The *EnvGenerator* drives the simulation by controlling the generation of interaction objects (artifacts) for the *Workers* to act upon, and for the *Authority* to check and further process. The *AtomicData* within a *SimulationElementType* defines which data may be passed between UserActivities and/or ObjActivities when an InternalSequence, ExternalSequence, or ObjectSequence fires. While arbitrary data types can be passed along, only AtomicData of type int, double, long, or boolean may be used as observable metrics during simulation execution. The exact applicable metrics are defined later on, on the simulation instance level.

As previously outlined, the iterative nature of the modeling process usually requires extending the core meta-model with domain-specific elements, as the need for them is identified. The domain-specific extensions we introduce are highlighted in bold/blue in Figure 3.

5.2 Simulation Model of the Real-World Scenario

In this section we derive an executable simulation model for evaluating the impact of various design decisions taken during the modeling of the case-study scenario from Section 4. Specifically, the goals of the simulation are:

i) prototyping and evaluating various incentive schemes;
ii) determining the impact of malicious user behavior;
iii) observing trends in processing costs, reward payments, report accuracy, and user activities.

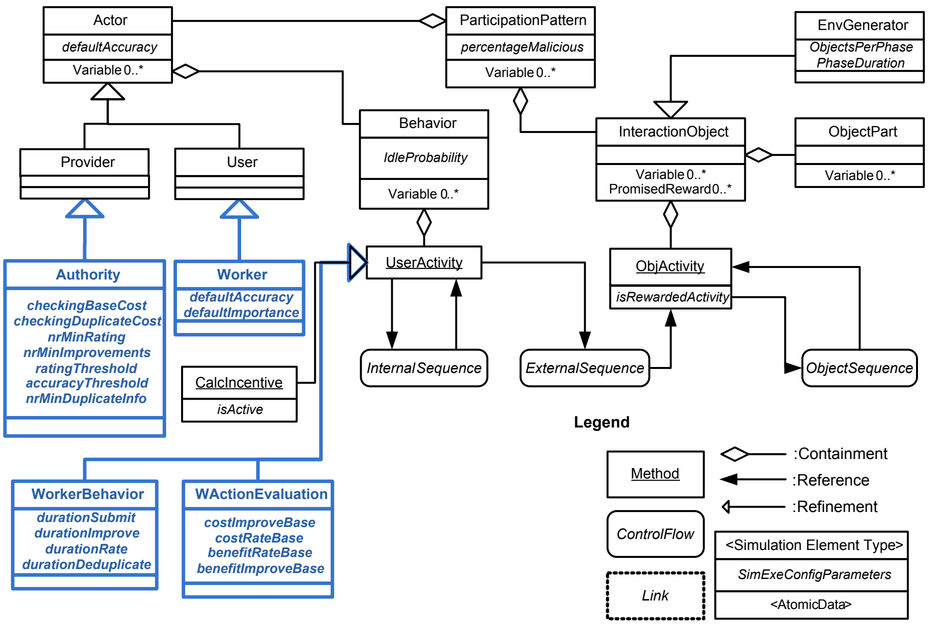


Fig. 3. Simulation meta-model including domain specific extensions in bold/blue

Figure 4 provides a partial screenshot of the case-study simulation model.

The simulation model comprises over 40 simulation parameters, determining various factors, such as: distribution of various personality characteristics in the worker population, injected worker roles (e.g., malicious, lazy), base costs for the authority, selection and composition of incentive mechanisms. Due to space limitations, describing them all in detail/formally is not possible. Therefore, the rest of this section is written in a narrative style.

Location and importance characterize a *Situation*. Situations can be generated with user-determined time, location and importance distributions, allowing us to concentrate more problematic (important) situations around a predefined location in selected time intervals, if needed. For the purpose of this paper, we generate situations with uniform probability across all the three dimensions. The *SituationGenerator* contains the activities for creating new situations and calculating phase-specific simulation metrics on cost, reputation, points, actions, and importance across reports, situations and workers.

The *Worker*'s *SetNextStep* activity represents the implementation of the worker's decision-making function f_a, introduced in Section 3. As previously explained, the worker here considers the next action to perform based on: 1) internal state (e.g., *location*), including innate, population-distributed personality characteristics (e.g, *laziness*, *isMalicious*); 2) current performance metrics (e.g., *reputation*, *points*); 3) advertized rewards (*detectionReward*, *ratingReward*, *improvementReward*).

Worker's location determines his/her proximity to a situation, and, thus, the likelihood to detect or act upon that situation (the smaller the distance, the higher the probability). However, two workers at the same distance from a situation will not equally

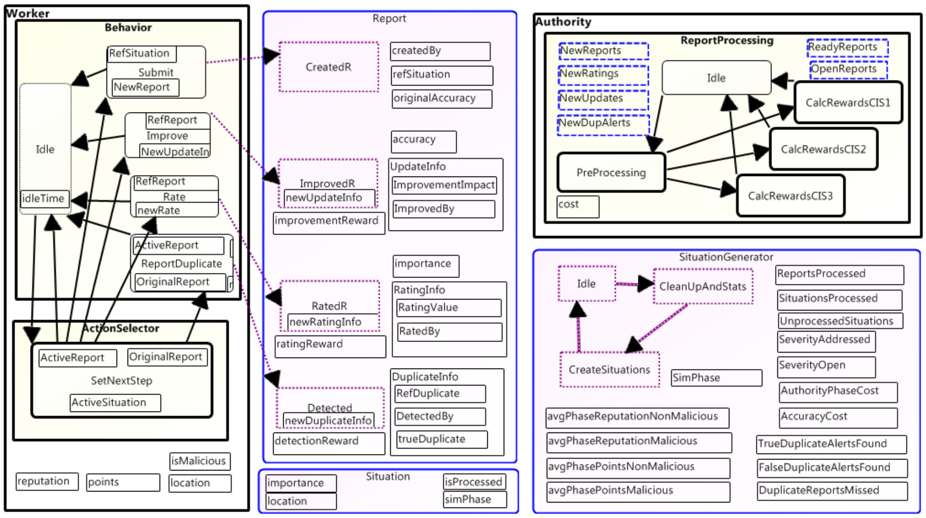


Fig. 4. Partial screenshot of the implemented case-study simulation model in DomainPro Designer

likely act upon it. This depends on their personality, past behavior, and the number of points they currently have.

Points and *reputation* are the principal two metrics by which the authority assesses Workers in our scenario. In principle, points are used by the Authority as the main factor to stimulate activity of a Worker. The more points, the less likely will a worker idle. On the other hand, a higher reputation implies that the Worker will more likely produce artifacts of higher quality. Each new worker joining the system starts with the same default point and reputation values. Precisely how the two metrics are interpreted and changed thereafter depends on the incentive mechanisms used (see below).

The four *Behavior* activities produce the respective artifacts – *Reports*, *UpdateInfos*, *RatingInfos* and *DuplicateInfos*. Worker's internal state determines the deviations of accuracy, importance, improvement effect, and rating value of the newly created artifacts. The subsequently triggered *Report*-located activities (*CreatedR*, *ImprovedR*, *RatedR* and *Detected*) determine the worker action's effect on the two metrics that represent the artifact's state and data quality metrics at the same time – report *accuracy* and *importance*. We use Bayes estimation to tackle the cold-start assessment of report accuracy and importance, taking into account average values of existing reports and the reputation of the worker itself.

The produced artifacts are queued at the *Authority* side for batch processing. In *PreProcessing* activity we determine whether a Report is ready for being processed. This depends on the report's quality metrics, which in turn depend on the amount and value of worker-provided inputs.

Processing reports causes costs for the Authority. The primary cost factors are low quality reports and undetected duplicate reports. Secondary costs arise when workers focus their actions on unimportant reports while ignoring more important ones.

Therefore, the Authority incentivizes the workers to submit required amounts of quality artifacts. As noted in Section 4, gathering as much inexpensive data from the crowd as possible was the original reason for the introduction of a crowdsourced process in the first place.

Our proof-of-concept simulation model for the given scenario defines three basic incentive mechanisms:

– IM_1**:** Users are assigned fixed amounts of points per action, independent of the artifact. Submitting yields most points.
– IM_2**:** The amount of points is increased before assignment, depending on the current quality metrics of the report. E.g., the fewer ratings or improvements the higher the increment in points.
– IM_3**:** Users are assigned a reputation. The reputation rises with accurately submitted reports, useful report improvements, correctly rated importance and correctly flagged duplicates.

As we shall see in Section 6.1, we can compose these three mechanisms in different ways to produce different incentive schemes which we can run and compare.

Workers that behave differently than usual can be easily injected into the system for simulation purposes by defining new *roles* and generating new workers or converting existing workers to take up those roles. For demonstration purposes we define only a single additional role - that of a malicious worker.

Malicious worker behavior is designed to cause maximum cost for the Authority. To this end, we assume malicious workers to have a good perception of the actual situation characteristics. Hence, upon submission they will set initial report importance low and provide very inaccurate information subsequently. For important existing reports they submit negative improvements (i.e., conflicting or irrelevant information) and rate them low and while doing the opposite for unimportant reports.

6 Evaluation

For evaluating our approach, we keep using the case-study scenario from the previous sections and perform a set of experiments on it. All provided experimental data is averaged from multiple, identically configured simulation runs. Details on the experiment setup are followed by experiment result[5] presentation and gained insights.

6.1 Experiment Setup

Timing Aspects. We control the pace of the simulation by determining the amount of situations created per phase. Taking a reading of all relevant (i.e., experiment-specific) metrics at the end of each phase provides an insight on how these metrics change over time. All our simulations last for 250 time units (t), consisting of 10 phases of $25t$ each. Batch creation of situations is representative for real world environments such as bugs that typically emerge upon a major software release or spikes in traffic impediments

[5] Data available here: `http://tinyurl.com/scekic-dorn-dustdar-coopis13`

coinciding with sudden weather changes. Report submission takes $5t$, while improving, rating, and duplication flagging require only $1t$. The exact values are irrelevant as we only need to express the fact that reporting requires considerably more time than the other actions. Processing of worker-provided data on the provider side occurs every 1t. Note here, that for the purpose of the case study, we are only interested in the generic processing costs rather than the time it takes to process that data. Each report is assumed to cause 10 cost units for zero-quality, and almost no cost when quality (through worker-provided improvements) approaches 1.

Scenario-Specific Thresholds. As we aim for high-quality data and significant crowd-base confirmation, the following thresholds need to be met before a report is considered for processing: at least three updates and high accuracy (> 0.75); or five ratings and medium importance (> 0.5); or four duplication alerts; or being reported by a worker of high reputation (> 0.8) and having high importance (> 0.7). Workers obtain various amounts of points for (correct) actions, the amount depending on the value of the action to the provider and the incentive scheme used.

Worker Behavior Configuration. A worker's base behavior is defined as 70% probability idling for $1t$, 20% submitting or duplication reporting, and 10% rating or improving. Obtained points and reputation increase the likelihood to engage in an action rather than idle. The base behavior represents rather active workers. We deliberately simulate only the top-k most involved workers in a community as these have most impact on benefits as well as on costs. Unless noted otherwise, $k = 100$ for all experiments.

Composite Incentive Schemes. The experiments utilize one or more of the following three *Composite Incentive Schemes – CIS*, introduced in Section 5.2:

- $CIS1 = IM_1$
- $CIS2 = IM_2 \circ IM_1 = IM_2(IM_1)$
- $CIS3 = CIS2 + IM_3 = IM_2 \circ IM_1 + IM_3$

CIS1 promises and pays a stable amount of points for all actions. CIS2 dynamically adjusts assigned points based on the currently available worker-provided data, but at least as high rewards as CIS1. CIS3 additionally introduces reputation calculation.

6.2 Experiments

Experiment 1: Comparing Composite Incentive Schemes. Here we compare the impact of CIS1, CIS2, and CIS3 on costs, assigned rewards, report accuracy, and timely processing. Figure 5 displays incurred costs across the simulation duration. All three schemes prove suitable as they allow 100 workers to provide sufficient data to have 20 situations processed at equally high accuracy. They differ, however, significantly in cost development (Fig.5 inset), primarily caused by undetected duplicate reports (on average 0.2, 0.25, and 0.4 duplicates per report per phase for CIS1, CIS2, and CIS3, respectively). CIS1 yields stable and overall lowest costs as the points paid induce just the right level of activity to avoid workers getting too active and thus causing duplicates. This is exactly the shortcoming of CIS2 which overpays workers that subsequently become overly active. CIS3 pays even more, and additionally encourages worker activity through reputation. The cost fluctuations are caused by the unpredictable number of

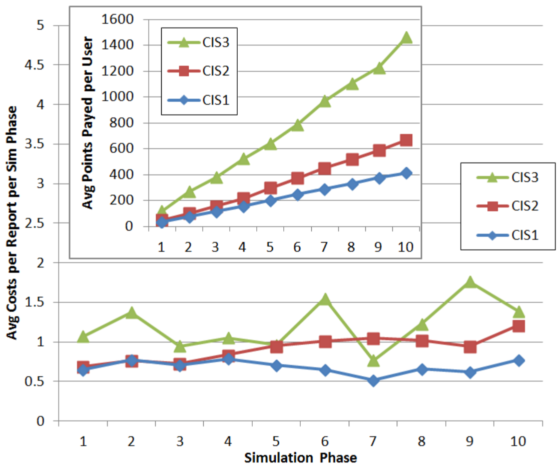


Fig. 5. Incurred report processing costs for CIS1, CIS2, and CIS3. Inset: average paid points per worker

duplicates (however remaining within bounds). Although more costly and less stable, CIS3 is able to identify and subsequently mitigate malicious workers (see Experiment 3 below).

Experiment 2: The Effect of Worker/Situation Mismatch. Here we analyze the effects of having too few or too many workers per situation. In particular, we observe per phase: the cost, points assigned, report importance (as reflecting situation importance), and reputation when: *i)* the active core community shrinks to 20 workers while encountering 50 situations (20u/50s); *ii)* a balance of workers and situations (100u/25s); *iii)* many active workers but only a few situations (100u/5s).

A surplus in situations (20u/50s) causes workers to become highly engaged, resulting in rapid reputation rise (Fig 7 bottom) coupled with extremely high values of accumulated rewarding points (Fig 6 inset). Costs per report remain low as duplicates become less likely with many situations to select from (0.18 duplicates per report). Here, CIS3 promises more reward for already highly-rated reports to counteract the expected inability to obtain sufficient worker input for all situation (on average 22 reports per phase out of 50). Subsequently, the authority receives correct ratings for reports and can focus on processing the most important ones. Compare the importance of addressed situations in Figure 7 top. A surplus in active workers (100u/5s) suffers from the inverse effect. As there is little to do, reputation and rewards grow very slowly. Perceiving little benefit, workers may potentially leave while the authority has a difficult time distinguishing between malicious and non malicious workers. Configurations (100u/5s) and (100u/25s) manage to provide reports for all situations, therefore having average report importance remaining near 0.5, the average importance assigned across situations.

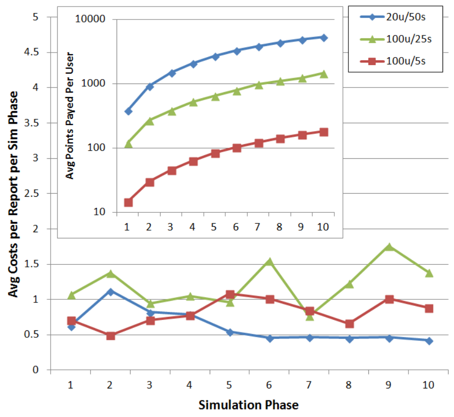


Fig. 6. Costs per report incurred at various combinations of worker and situation count

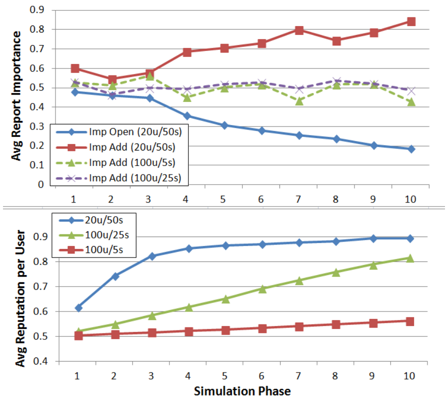


Fig. 7. Reputation acquired by workers (bottom), and report **imp**ortance **add**ressed, respectively remaining **open** (top)

Experiment 3: Effect of Malicious Workers. Here we evaluate the effects of an increasing amount of malicious workers on cost when applying CIS3. Figures 8 and 9 detail cost and reputation for 0%, 20%, 30%, 40%, and 50% malicious workers. All workers are considered of equal, medium reputation 0.5 upon simulation start. The drop in costs across time (observed for all configurations) highlights that the mechanism indeed learns to distinguish between regular, trustworthy workers and malicious workers. The irregular occurrence of undetected duplicates cause the fluctuations in cost apparent for 0% and 20% malicious workers. Beyond that, however, costs are primarily determined by low accuracy induced by malicious workers. CIS3 appears to work acceptably well up to 20% malicious workers. Beyond this threshold harsher reputation penalties and worker blocking (when dropping below a certain reputation value) need to be put in place. In severe cases lowering the default reputation assessment might be applicable but requires consideration of side effects (i.e., thereby increasing the entry barrier for new workers).

6.3 Limitations and Discussion

Simulations of complex socio-technical processes such as the use-case presented here can only cover particular aspects of interest, never all details. Thus any results in terms of absolute numbers are unsuitable to be applied directly in a real-world systems. Instead, the simulation enables incentive scheme engineers to compare the impact of different design decisions and decide what trade-offs need to be made. The simulation outcome provides an understanding what mechanisms might fail earlier, which strategies behave more predictably, and which configurations result in a more robust system design.

In particular, the presented comparison of CISs in Experiment 1 gives insight into the impact of overpaying as well as indicating that the CIS3 would do well to additionally include a mechanism to limit submissions and better reward the action of flagging the

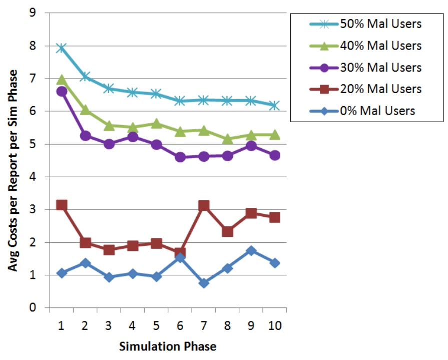


Fig. 8. Costs per report incurred due to various level of malicious workers.

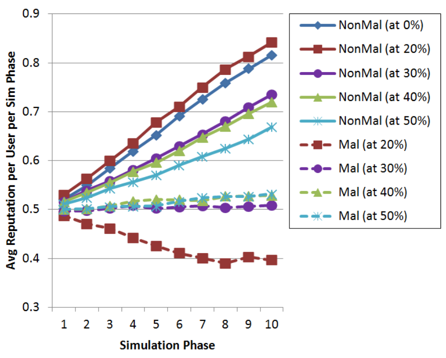


Fig. 9. Average reputation acquired by malicious and non-malicious workers.

duplicates. Experiment 2 provides insights on the effect of having too few or too many workers for a given number of situations. It highlights the need to adjust rewards and reputation in reaction to shifts in the environment and/or worker community structure. Experiment 3 provides insight into the cost development in the presence of malicious worker and highlights the potential for mechanism extension.

True advantages of our simulation approach can be appreciated when used together with an automated system for incentive management (e.g., [17]). The incentive management system can then be used to provide a number of simulation parameters, so that the system architect can quickly set up a simulation environment resembling the real system. The new incentive mechanisms and their combinations can then be tried out and the feedback sent to the incentive management system which can then re-adjust its incentive scheme setup.

7 Conclusion and Outlook

In this paper we presented a methodology for modeling and simulating incentives in crowdsourcing environments, highlighting the challenges with which the system architects are faced, and pointing to a possible way of alleviating them. We intend to continue our work on modeling incentives, trying to devise suitable models for different and more complex processes and environments. In the long run, we intend to develop a uniform and generally applicable language/notation for the description and ad-hoc instantiation of various incentive processes on real-world socio-technical systems.

Acknowledgments. This work has been partially supported by the EU FP7 SmartSociety project under Grant No. 600854 and the Austrian Science Fund (FWF) under Grant No. J3068-N23. The first author of this paper is a student of the Vienna PhD School of Informatics.

References

1. Adar, E.: Why I Hate Mechanical Turk Research (and Workshops). Control (2011), http://www.cond.org/eadar-crowdsourcing-workshop.pdf
2. Bloom, M., Milkovich, G.: The relationship between risk, incentive pay, and organizational performance. The Academy of Management Journal 41(3), 283–297 (1998), http://www.jstor.org/pss/256908
3. Dorn, C., Edwards, G., Medvidovic, N.: Analyzing design tradeoffs in large-scale socio-technical systems through simulation of dynamic collaboration patterns. In: Meersman, R., et al. (eds.) OTM 2012, Part I. LNCS, vol. 7565, pp. 362–379. Springer, Heidelberg (2012)
4. Gal, Y., Grosz, B., Kraus, S., Pfeffer, A., Shieber, S.: Agent decision-making in open mixed networks. Artif. Intell. 174(18), 1460–1480 (2010), http://dx.doi.org/10.1016/j.artint.2010.09.002
5. Gilbert, N., Troitzsch, K.: Simulation for the social scientist. Open University Press, McGraw-Hill Education (2005)
6. Heckman, J., Smith, J.: What do bureaucrats do? The effects of performance standards and bureaucratic preferences on acceptance into the JTPA program. Advances in the Study of Entrepreneurship Innovation and Economic Growth 7, 191–217 (1996)
7. Kraus, S., Hoz-Weiss, P., Wilkenfeld, J., Andersen, D.R., Pate, A.: Resolving crises through automated bilateral negotiations. Artificial Intelligence 172(1), 1–18 (2008), http://www.sciencedirect.com/science/article/pii/S0004370207001051
8. Laffont, J.J., Martimort, D.: The Theory of Incentives. Princeton University Press, New Jersey (2002)
9. Little, G., Chilton, L.B., Goldman, M., Miller, R.: Exploring iterative and parallel human computation processes. In: Proceedings of the 28th of the International Conference Extended Abstracts on Human Factors in Computing Systems - CHI EA 2010, p. 4309 (2010), http://portal.acm.org/citation.cfm?doid=1753846.1754145
10. Macal, C.M., North, M.J.: Agent-based modeling and simulation. In: Proceedings of the 2009 Winter Simulation Conference (WSC), pp. 86–98 (December 2009), http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5429318
11. Mason, W., Watts, D.J.: Financial incentives and the performance of crowds. In: Proceedings of the ACM SIGKDD Workshop on Human Computation (HCOMP 2009), vol. 11, pp. 77–85. ACM, Paris (May 2009), http://portal.acm.org/citation.cfm?doid=1809400.1809422
12. Milinski, M., Semmann, D., Krambeck, H.J.: Reputation helps solve the 'tragedy of the commons'. Nature 415(6870), 424–426 (2002), http://dx.doi.org/10.1038/415424a
13. Prendergast, C.: The provision of incentives in firms. Journal of Economic Literature 37(1), 7–63 (1999), http://www.jstor.org/stable/2564725
14. Rockenbach, B., Milinski, M.: The efficient interaction of indirect reciprocity and costly punishment. Nature 444(7120), 718–723, http://dx.doi.org/10.1038/nature05229
15. Scekic, O., Truong, H.-L., Dustdar, S.: Modeling rewards and incentive mechanisms for social BPM. In: Barros, A., Gal, A., Kindler, E. (eds.) BPM 2012. LNCS, vol. 7481, pp. 150–155. Springer, Heidelberg (2012), http://dx.doi.org/10.1007/978-3-642-32885-5_11
16. Scekic, O., Truong, H.-L., Dustdar, S.: Incentives and rewarding in social computing. Commun. ACM 56(6), 72–82 (2013), http://doi.acm.org/10.1145/2461256.2461275
17. Scekic, O., Truong, H.-L., Dustdar, S.: Programming incentives in information systems. In: Salinesi, C., Norrie, M.C., Pastor, O. (eds.) CAiSE 2013. LNCS, vol. 7908, pp. 688–703. Springer, Heidelberg (2013), http://dx.doi.org/10.1007/978-3-642-38709-8_44

18. Semmann, D., Krambeck, H.J., Milinski, M.: Strategic investment in reputation. Behavioral Ecology and Sociobiology 56(3), 248–252 (2004), http://dx.doi.org/10.1007/s00265-004-0782-9
19. Tokarchuk, O., Cuel, R., Zamarian, M.: Analyzing crowd labor and designing incentives for humans in the loop. IEEE Internet Computing 16, 45–51 (2012)
20. Wedekind, C., Milinski, M.: Cooperation Through Image Scoring in Humans. Science 288(5467), 850–852 (2000), http://dx.doi.org/10.1126/science.288.5467.850
21. Wegner, R.: Multi-agent malicious behaviour detection. Phd thesis, University of Manitoba (2012), http://hdl.handle.net/1993/9673

$\mathcal{DS}^4$: Introducing Semantic Friendship in Distributed Social Networks[⋆]

Paraskevi Raftopoulou[1], Christos Tryfonopoulos[1], Euripides G.M. Petrakis[2], and Nikos Zevlis[1]

[1] University of Peloponnese, Tripoli, Greece
[2] Technical University of Crete, Chania, Greece
{praftop,trifon,cst06027}@uop.gr,
petrakis@intelligence.tuc.gr

Abstract. Existing social networks are based on centralised architectures that manage users, store data, and monitor the security policy of the system. In this work, we present $\mathcal{DS}^4$, a Distributed Social and Semantic Search System that allows users to share and search for content among friends and clusters of users that specialise on the query topic. In $\mathcal{DS}^4$, nodes that are semantically, thematically, or socially similar are automatically discovered and logically organised into groups. Content retrieval is then performed by routing queries towards social friends and clusters of nodes that are likely to answer them. In this way, search receives two facets: the social facet, addressing friends, and the semantic facet, addressing nodes that are semantically close to the queries. Our experiments demonstrate that searching only among friends is not effective in distributed social networks, and showcase the necessity and importance of semantic friendship.

1 Introduction

In recent years a number of social networking services have been developed to offer users a new way of sharing, searching, and commenting on user-generated content. Following the development of such services, people have shown great interest in participating in "social" activities by generating and sharing vast amounts of content, ranging from personal vacation photos to blog posts, or comments and like/agree/disagree tags. All these social networking services are typically provided by a centralised site, where users need to upload their content, thus giving away access control and ownership rights as a requirement to making it available to others. This centralised administrative authority may sometimes utilise the content in any profitable way, from selling contact details to marketing firms to mining of user information for advertising purposes. Furthermore, the rate of growth of both content and user participation in such services raises concerns about the scalability of the centralised architectures used, as they are called to serve millions of users and gigabytes of content every day.

[⋆] Research leading to these results has received funding from the EC's 7th FP (FP7/2007-2013) under grant agreement No 604691 (Project Fi-Star).

Idea and Challenges. In this work, we present a *distributed social networking architecture* that allows users to share and search for content in a fully decentralised way, while at the same time maintaining access control and ownership of their content. Such a design is ideal for implementing scientific or enterprise social networks, where people are reluctant to upload their data to a third party. Our work builds upon research results from the peer-to-peer (P2P) paradigm, such as those utilising unstructured, small-world, and semantic overlay networks (SONs) [1,2,3]. Replacing the centralised authority with a distributed self-manageable community of nodes removes access control and ownership issues and ensures high-scalability and low maintenance costs. For this reason, recent efforts in the industry and the literature have also resorted to the P2P paradigm for building decentralised social networks and platforms (like Diaspora[1], KrawlerX[2], and OpenSocial[3]), and have developed architectures [4] and prototype systems [5] relying mainly on Distributed Hash Tables (DHTs). Contrary to DHTs that focus on providing accurate location mechanisms, $\mathcal{DS}^4$ emphasises on node autonomy, content-based grouping of nodes, and loose component architecture by relying on the SON paradigm. The designed system is scalable (requires no centralised component), privacy-aware (users maintain ownership and control over their content), automatic (requires no intervention by the user), general (works for any type of content), and adaptive (adjusts to changes of user content or interests).

In $\mathcal{DS}^4$, node organisation is achieved by executing *identification and grouping of semantic friends* (periodically) by each node. This protocol operates by establishing connections among *semantically similar nodes* (in addition to the social connections) and by discarding connections that are outdated or pointing to dissimilar nodes. The goal of this protocol is to create *groups/clusters of nodes* with similar interests. User queries can then be resolved by routing the query towards friends and nodes specialising to the query topic. In this way, content search is leveraged to another type of friendship often ignored in social networks: the *semantic friendship* emerging from common user interests or user profiles.

Contribution. In the light of the above, the contributions of this work are threefold:

- We define a distributed social networking architecture that offers fundamental social interactions, while emphasising on *content search*, *user autonomy*, and *data ownership*. To the best of our knowledge, this is the first approach to propose a *distributed social networking system* that introduces the notion of *semantic friendship* between users.
- We present the *protocols* and *services* that regulate node interactions. We provide details on the definition of the semantic friendship between users, present details on the distributed social and semantic search algorithms, and discuss *system implementation* issues.
- We show the importance of semantic friendship in such a distributed context by means of experimentation with real social networking data, both on a

[1] `https://joindiaspora.com/`
[2] `http://www.krawler.com/`
[3] `http://docs.opensocial.org/display/OS/Home`

simulated environment and on a *prototype system*. Our experiments show that by resorting only to friends for content retrieval we achieve recall as low as 10%, while introducing a basic notion of semantic similarity between nodes increases recall to 30%, but incurs high message traffic. Contrary, with the utilisation of semantic friendship we achieve a recall up to 80%, while message traffic is kept low.

Application Scenario. As an example of an application scenario let us consider Mary, a computer scientist whose main field of expertise is bioinformatics. Mary is interested in following the work of prominent researchers in the area and willing to share her own work and ideas with other researchers; this interaction would help her set-up her research agenda and generate innovative ideas. Currently, she would have to (i) resort to a number of digital libraries or scientific databanks, like DBLP or GenBank, to identify interesting bibliography, (ii) use one of the numerous centralised social networks, like LinkedIn or ResearchGate (aimed for professional or research use), to follow the work of other researchers, and (iii) use one of the numerous file hosting services, like Dropbox or Fileserve, to exchange her dataset with her colleagues. Clearly, Mary would benefit from accessing a Web 2.0-inspired solution that is able to provide socially-/semantically-aware search and data sharing with ownership and access control in an *integrated service*. This system would be a valuable tool, beyond anything supported in current centralised social networks, that would allow Mary to save both time and effort.

In our example scenario, consider a research community comprised of researchers working in different institutions and content providers of scientific material, like digital libraries, scientific publishers, or scientific databanks. In this context, each user will personalise and maintain its own node in the distributed social network that will act as an access point to the network services. A node will allow its user to socially connect with colleagues or collaborators and exchange ideas, manuscripts, or even datasets in a P2P fashion, thus maintaining full control of the data dissemination process. Additionally, a user might also be semantically connected in an automatic but user-centered fashion to other nodes with similar interests –maintained either by other researchers in the same field or by specialised content providers. Additionally, nodes may also be deployed by larger institutions, like research centers or content providers (e.g., CiteSeer, ACM, Springer), to provide access points for their content that may be freely disseminated or even priced (on the basis of pay-per-item or subscriptions). The $\mathcal{DS}^4$ architecture would be a promising solution to such a setting as it (i) emphasises the seamless integration of information sources, (ii) supports user-centered access control over the shared content, and (iii) enhances fault tolerance and requires no central administration authority.

The rest of the paper is organised as follows. Section 2 discusses related work. Section 3 introduces the proposed architecture, implemented services, and protocols that regulate node interactions, while Section 4 presents our experimental evaluation on a simulated environment and on the deployed prototype system. Finally, Section 5 concludes the paper and discusses future research directions.

2 Related Work

In this section, we discuss related work in the context of platforms for distributed social networks and social data management.

2.1 Distributed Social Platforms

All available social networks (e.g., Facebook, LinkedIn, Elgg) are currently based on centralised solutions both for storing and managing of content, which set scalability limitations on the system and reduce fault-tolerance. Industry has already detected these drawbacks and has lately turned into solutions that diverge from the centralised model of the existing systems by developing platforms, such as Diaspora, Krawler and OpenSocial, that provide APIs to support application hosting in remote application servers, owned and managed by the application providers. In a similar spirit, a strand of research work also moved towards hierarchical organisations for supporting distributed social networking. The distributed social systems SuperNova [6] and Scope [7] are based on a two-tier architecture, where nodes with higher computing capability become super-nodes and form an overlay to provide distributed data management of the P2P social network. Client nodes connect to super-nodes and rely on them for bootstrapping, sharing their content, and accessing the shared information. Although all of the platforms and system schemes propose the decentralisation of the social services, one of the main issues of the centralised architectures persists: the existence of a single point where user information is collected and may be exploited.

To alleviate the above disadvantage, distributed platforms for social online networks based on the P2P paradigm were proposed [8,9,10]. LifeSocial.KOM [10] is a plugin-based extendible social platform that provides secure communication and user-based data access control, and integrates a monitoring component that allows users and operators to observe the quality of the distributed system. Similar efforts aimed at spontaneous social networking; they include proposals for distributed social services in resource constrained devices (like tablets or smartphones) [11,12] or in environments with no infrastructure guarantees (e.g., high-attendance events) [7]. All these approaches offer different types of distributed social platforms that allow users to create communities, share content, and send messages, but do not emphasise expressive content search mechanisms.

Finally, other approaches in distributed social networking emphasise on delivering innovative and competitive services; SCIMS [13] relies on an ontology-based model for managing social relationships and status, the work in [14] aims at personalising search results based on user context and friendship relations, while Gemstone [15] targets data availability in the absence of the data owner. To achieve this, a replica storage scheme based on social relationships, online patterns of nodes, and user experiences is utilised.

2.2 Distributed Social Data Management

Our work fits mainly into the area of data management in distributed social networks and is inspired by previous approaches on SONs [1,2,3] and on works that emphasise on distributed content location in social networks. Works like the eXO [4] and SoNet [16] systems are, similarly to $\mathcal{DS}^4$, inspired by the P2P paradigm to provide content location and management services on large-scale decentralised social networks. To do so, the authors rely on a structured overlay and exploit the accurate location mechanisms, but de-emphasise node autonomy. Contrary to these approaches, $\mathcal{DS}^4$ employs a loose component architecture and introduces a new type of social relations between nodes: the semantic closeness of content. In this way, nodes that are similar in terms of content, create emergent groups likewise to the creation of social relations. Our work shares ideas with the SocialCDN system [17], where social caches (links among friends) are introduced as a way to alleviate the network traffic and optimise data dissemination (mainly by social updates). In [17], social cache selection is formulated as the neighbour-dominating set problem and a family of algorithms is proposed and evaluated. Contrary to $\mathcal{DS}^4$, where the emphasis is on efficiently supporting expressive content retrieval in the social paradigm, the emphasis on SocialCDN is on the reduction on network traffic to facilitate fundamental social interactions.

The loose component architecture and the emphasis on node autonomy of [18] resemble the architectural design of $\mathcal{DS}^4$, where an unstructured overlay network of nodes is utilised to support the distributed social infrastructure. However, the focus of [18] is on the design of gossip protocols for efficiently disseminating profile updates to all interested users and does not put any attention to the problem of content search and management.

Furthermore, the concept of creating and maintaining social connections in distributed infrastructures is affined with the problem of distributed data management in P2P networks. In SONs [3,19], "social" connections between the peers (e.g., similarity of content, pattern, or distance in a physical level) are exploited to direct the search to nodes with relevant data (e.g., as in [20] that studies query routing strategies based on "social" relationships). Other works on SONs (e.g., [2]) focus more on the organisation of P2P networks as small-world networks, where peers self-organise in groups of similar interests to facilitate message-efficient query answering. Our work on $\mathcal{DS}^4$ borrows concepts and ideas from research on SONs and extends them for facilitating efficient and effective data management in a social network setting. We suggest that SONs offer the most promising architectural solution inspired from the P2P paradigm; it is a perfect fit for a distributed social networking scenario providing high decentralisation, high node autonomy, support for emergent semantic and social structures, and effective object location mechanisms. Contrary, DHT-based architectures [4,5] ignore node autonomy (by enforcing deterministic key/content placement) and emergent structures (by enforcing network structure).

Finally, a large number of research in the domain of distributed social networks consists of studies on system security [5,21], user privacy [5,21,22,23,24], distributed access control [25,26], and authentication mechanisms [25]. Clearly, security issues are also relevant in our design and the $\mathcal{DS}^4$ system could benefit by adopting approaches like [25] or [23] that enforce user privacy and access control. However, the problem of security is orthogonal to our design and is not further analysed as it is not the emphasis of this work.

3 The $\mathcal{DS}^4$ Social Networking System

A distributed alternative for social networks should provide the same functionality as current solutions, while avoiding centralised storage and ownership of user data (that are anyway naturally distributed to users' personal computers). In our design, we aim at providing support for open social networks and innovative services, while preserving end-user privacy and information ownership. The benefits of distributed self-managing solutions extend over the architectural domain to economic figures of social networks. With the proliferation of high-bandwidth internet connections and powerful off-the-shelf personal computers, a distributed solution based on P2P technology will realise a cost-effective way to develop and manage social networks, avoiding maintenance, storage, and administrative costs of centralised solutions.

3.1 Protocol Overview

To demonstrate the necessity and effect of semantic friendship in a distributed social networking system we have designed and implemented four different protocols that may be executed by each node. All the protocols outlined below, together with their implementation details with regard to each of the system services, are described in detail in Section 3.3. The performance of each protocol in terms of retrieval effectiveness and message traffic is evaluated in Section 4.

The FI Protocol. Under the FI protocol, each network node maintains a Friend Index (FI) routing table containing the contact details (i.e., IP and port) of the social neighbourhood of the node, comprised of explicitly declared friends that participate in the $\mathcal{DS}^4$ network. This is the most straightforward form of a distributed social network one could design and is used for providing the baseline comparison for our experiments. In the FI protocol neither interest identification nor semantic grouping is applied, since a node addresses a limited number of random friend node(s) at query time, thus performing a message-bounded flooding.

The FI+i Protocol. The FI+i protocol is similar to the FI protocol described above, but targets to demonstrate the usefulness of the semantic information in its simplest form. Under the FI+i protocol, when a node n connects to the $\mathcal{DS}^4$ network its interests are automatically derived by its local content as described in Section 3.2. Each node maintains a FI routing table containing the contact details mentioned above, together with the interest descriptions derived from the interest identification process executed locally by its social friends. In this way, at query time a node may address the friend(s) that are the most relevant to the query. The protocol is named after the combination of the FI with interests.

The SI+g Protocol. In the SI+g protocol, each node n maintains, in addition to the FI containing the contact details and interest descriptions of social friends, a Semantic Index (SI) routing table containing the contact details and interest descriptions of nodes sharing similar interests with node n (called semantic friends). The links in the SI form the semantic neighbourhood of the node, and will be refined accordingly by using the semantic grouping service described in Section 3.3 (by resorting *only to the SI* of other nodes). In this way, at query time the node(s) most relevant to the issued query are addressed. This protocol is designed to demonstrate the importance of semantic friendship in the content retrieval process, and is named after the combination of the SI with the grouping of semantic friends.

The FISI+g Protocol. The FISI+g protocol is similar to the SI+g protocol described above as the node maintains again two routing tables (SI and FI). However, under the FISI+g protocol the contents of SI will be modified during semantic grouping by resorting *both to the SI and FI* of other nodes. Similarly to the SI+g protocol, at query time the node(s) most relevant to the issued query are addressed. This protocol is designed to demonstrate the importance of the FI in the grouping of semantic friends, and show the potential of combining social and semantic friendship. The protocol is named after the combination of both FI and SI with the grouping of semantic friends.

3.2 Interest Identification

The interests of each node are automatically derived by its local content (meta-) data (e.g., user-defined tags, document titles, etc.) and naturally, a node may have more than one interests. The interests of a node are identified automatically, i.e., by applying a standard clustering algorithm like Hierarchical Agglomerative Clustering (HAC) [27] or K-means [27] on the node's local content repository. A node's interest is represented by the corresponding index terms (if an external reference system is used) or by the centroid vector (if content is categorised by clustering). Each node is then represented by the list of the centroid vectors of its interests.

3.3 $\mathcal{DS}^4$ Service Description

The main idea behind $\mathcal{DS}^4$ is to let nodes that are semantically, thematically, and socially close self-organise to facilitate the search mechanism. The services regulating node join, creation and maintenance of semantic friendship, query processing, and social interactions for all $\mathcal{DS}^4$ protocols mentioned above are discussed in the following sections.

Join Service. When a node connects to the $\mathcal{DS}^4$ network, it uses the join service which differs depending on the protocol followed. Since users are expected to connect also through mobile devices, we allow nodes to use dynamic IP addresses to enable a higher degree of decentralisation and dynamicity. Nodes are thus identified by a unique ID computed at the first time of connection and may connect, disconnect, or leave the system silently at any time.

According to the FI protocol, a node joining the $\mathcal{DS}^4$ network needs only to locate its social friends and update its FI with their contact details. Joining the network under the FI+i, SI+g, and FISI+g protocols involves a different procedure, since the node has to derive its interests as described in Section 3.2.

In the FI+i protocol, after calculating its interests, the node contacts its social friends to notify them about its interests and asks in turn for their interests. This information is then used to update its FI. In addition to the FI, in the SI+g and FISI+g protocols, each node maintains also a semantic index SI_{ik} containing the contact details and interest descriptions of nodes sharing similar interests (called *short-range links*) and of a small number of nodes with dissimilar interests (called *long-range links*. Long-range links are used to promote serendipitous discovery of information, and it has been shown that they are able to affect the effectiveness and efficiency of the retrieval [20]. This semantic index is maintained for each distinct interest I_{ik} of node n_i. A node may merge or split its semantic indices by merging or splitting its interests, depending on changes in its local content.

These (initially randomly selected) links form the semantic neighbourhood of the node; the short-range links contained in SI are refined accordingly by using the semantic grouping service (invoked either periodically or explicitly by the user) described below. Long-range links are updated by random walks in the network as in [20]. In the following, for simplicity of the presentation, we assume that each node n_i has only one interest I_i, and thus maintains only one SI_i. The same discussion applies for multiple interests, since nodes maintain one semantic index per interest and the semantic grouping service is applied independently for each one of them.

Semantic Grouping Service. The semantic grouping service is responsible (i) for updating social and semantic friends' interests both in FI and SI, and (ii) for reorganising the semantic neighbourhood (SI) of the node by establishing new connections and discarding old ones, forming groups of nodes with higher semantic friendship (i.e., more similar interests).

For this reason, the FI protocol requires no semantic grouping and incurs no extra message cost at the operation of the system, as it uses only the FI without node interests. Additionally, semantic grouping for the FI+i protocol refers only to the regular update of the node interests stored at the FI, and incurs a cost proportional to the number of social links maintained in the FI.

The most interesting facet of the semantic grouping service is utilised in the SI+g and FISI+g protocols. In these protocols, the network nodes use the service to form groups of nodes based on their likelihood to have similar interests (i.e., their semantic friendship). As already mentioned in the previous section, the SI of each node is used to maintain two types of semantic friendship links: short-range links that are links to other nodes with similar interests, and long-range links that links to nodes having different interests. Short-range links are used to *create groups* (clusters) of nodes with high semantic friendship, while long-range links are used to *maintain the connectivity* of remote semantic groups (clusters) in the system. This procedure for grouping semantic friends is executed locally by each node and aims at clustering nodes with similar content, so as to allow forwarding of queries to social friends and clusters of semantic friends that are similar to the issued query.

Semantic grouping resembles *emerging group creation* behaviours in social networks, where users with similar interests utilise word-of-mouth to cluster around a fan page or another user. This process in $\mathcal{DS}^4$ may occur in two facets: manually, by explicitly selecting to become friends with other users, and automatically through the semantic grouping service. This procedure, may also function as a *recommendation mechanism* for creating new social friendships.

Each node n_i may (either periodically or when explicitly invoked) initiate the semantic grouping service. The node computes its average Semantic Neighbourhood Similarity (SNS) to semantic friends as $SNS_i = \frac{1}{|c_i|} \cdot \sum_{\forall n_j \in c_i} sim(I_i, I_j)$, where $|c_i|$ is the number of n_i's connections (i.e., social friends and/or semantic links depending on the protocol) and $sim()$ can be any appropriate similarity function (e.g., the cosine similarity between the term vector representations). If the similarity computed is greater than a (user-defined) threshold θ then the node does not need to take any further action, since it is surrounded by nodes with similar interests. Otherwise, the node initiates a group refinement process by forwarding a message in the network with a (user-defined) time-to-live (TTL) to collect other nodes' interests. In particular, under both SI+g and FISI+g protocols, a node n_i issues a FINDNODES $= (id(n_i), I_i, P, t_g)$ message, where $id(n_i)$ is the identifier of n_i, P is an (initially empty) list, and t_g is the TTL of the message. The issued message is forwarded with equal probability to either m randomly chosen nodes or the m most similar nodes to the message initiator. The rationale of applying either of the forwarding strategies is that the message initiator should be able to reach similar nodes both directly (through other similar nodes), but also indirectly (through propagation of the FINDNODES() message through non-similar nodes). The only difference between the SI+g and FISI+g protocols lies in the utilisation of the routing tables: the SI+g protocol uses only the semantic connections of a node (i.e., only the entries in the SI routing table), while the FISI+g protocol uses both the semantic and social connections (i.e., entries in both the SI and FI routing tables) to compute neighbourhood similarity and propagate the message. In this way, we are able to study the effect of social friendship in the grouping of semantic friends.

Each node that receives the FINDNODES() message adds its contact details and its most similar interest in list P of the message, reduces TTL by one, and forwards the message in the same manner. When the TTL reaches zero, the message (containing the contact information and interests of all nodes that received it) is sent back to the initiator node, which uses the collected information to refine the (short-range) links contained in the *SI*. Additionally, to speed up semantic grouping, every intermediate node receiving the FINDNODES() message may utilise the message information to refine its semantic connections. The pseudocode providing a high-level description of semantic grouping in protocols SI+g and FISI+g is given in Figure 1(a).

Query Processing Service. Queries are issued as free text or keywords and are formulated as term vectors. Subsequently, the node issues a query message in the network with a (user-defined) TTL t_q using its social (FI) and/or semantic (SI) connections depending on the protocol implemented. All the nodes receiving the query message reduce t_q by one and apply the same forwarding technique; the query message is not forwarded further in the network when $t_q = 0$. Additionally

```
1: compute $SNS_i$
2: if $SNS_i < \theta$ then
3:   $P \leftarrow \{ \}$
4:   initiate message FindNodes = $(id(n_i), I_i, P, t_g)$
5:   send FindNodes to:
     - $m$ random neighbours or
     - the $m$ most similar neighbours to $n_i$
         ▷each neighbour $n_j \in SI$ for SI+g
         ▷each neighbour $n_j \in SI \cup FI$ for FISI+g
6:   $P \leftarrow P \cup \{(id(n_j), I_j)\}$
7: reduce message TTL $t_g$ by 1
8: do the same for the neighbours of $n_j$
9: repeat until message TTL $t_g = 0$
10: return list $P$ to $n_i$
```

(a)

```
   ▷compare with $n_i$'s interest
1: if $sim(q, I_i) > \theta$ then
2:   $R_i \leftarrow \{ \}$
     ▷compare with $n_i$'s local content
3:   if $sim(q, d) > \theta$ then
       ▷identify matching content
4:     $R_i \leftarrow R_i \cup (d, sim(q, d))$
5:   initiate message Query = $(id(n_i), q, t_q)$
6:   send Query to:
     - all short-range links of $n_i$ and
     - the $m$ friends of $n_i$ most similar to $q$
7: else
8:   send Query to
     the $m$ connections of $n_i$ most similar to $q$
9: reduce query TTL $t_q$ by 1
10: do the same for each visited node $n_j$
11: repeat until query TTL $t_q = 0$
12: return answer sets $R_j$ to $n_i$
13: rank results $R \leftarrow \cup R_j$ by similarity to $q$
```

(b)

Fig. 1: Pseudocode for (a) grouping of semantic friends and (b) query processing

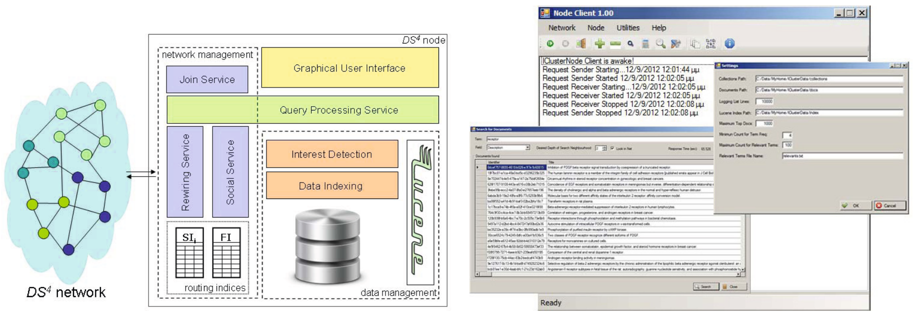
to the forwarding of the query message, each node executes the query locally, identifies matching content and returns appropriate pointers and metadata to the query initiator. Finally, the query initiator collects all responses, produces a list with the candidate answers ordered by similarity to the issued query, and presents the list to the user. Figure 1(b) summarises the steps of the query processing algorithm under the SI+g or FISI+g protocol.

Query forwarding depends on the implemented protocol, as not all protocols have the same routing indices. In the FI protocol a node forwards the query to m social friends found in the *FI*, without resorting to any form of semantic relatedness between the issued query and the contacted nodes. This is a baseline scenario used to demonstrate a distributed social network without explicitly designed content retrieval capabilities.

Contrary to the FI protocol, where the query q is forwarded in an uninformed way, in the FI+i protocol q is forwarded in an informed way: a node forwards a query message to its m social friends that have the highest similarity to q, i.e., to those nodes in its FI that have interests closer the issued query.

Finally, query forwarding in the SI+g and FISI+g protocols is the same and involves forwarding the query to (i) a number of semantic friends that are expected to have interests similar to the issued query and are contained in the SI of the node (semantic facet of the search), and (ii) a number of social friends contained in the FI of each node that are similar to the issued query (social facet of the search). Specifically, the message initiator compares the query against its interest(s) and, if similar, the query is forwarded to all of its short-range links and similar friends, i.e., the message is *broadcasted* to the node's neighbourhood (*query explosion*)[4]. Otherwise, the query is forwarded to the nodes (found in either the *SI* or the *FI*) that have the highest similarity to the query (*fixed*

[4] Notice that the query explosion is bound by a short TTL (typically 1 or 2) and the number of *SI* links.



!ClusterNode Client is awake!
Request Sender Starting...12/9/2012 12:01:44 μμ
Request Sender Started 12/9/2012 12:02:05 μμ
Request Receiver Starting...12/9/2012 12:02:05 μμ
Request Receiver Started 12/9/2012 12:02:05 μμ
Request Receiver Stopped 12/9/2012 12:02:08 μμ
Request Sender Stopped 12/9/2012 12:02:08 μμ

Collections Path: C:/Data/MyHome/!ClusterData/collections
Documents Path: C:/Data/MyHome/!ClusterData/docs
Logging List Lines: 10000
Lucene Index Path: C:/Data/MyHome/!ClusterData/index
Maximum Top Docs: 1000
Minimum Count for Term Freq: 4
Maximum Count for Relevant Terms: 100
Relevant Terms File Name: relevants.txt

(a) (b)

Fig. 2: (a) $\mathcal{DS}^4$ node architecture and (b) GUI with results and settings screen

forwarding). The query routing strategy induced by both protocols is referred to in the literature as the *fireworks* technique [1,2] and combines low message traffic with high efficiency in clustered distributed environments.

Since many end-users are expected to connect through mobile devices with limited resources, replication and caching techniques (such as [28]) are necessary to maintain data availability. Notice however, that our design contains an inherent replication mechanism through friend-of-a-friend (foaf) links.

Social Networking Services. All social networking functionality is implemented by resorting on unicast or multicast messages to the explicitly declared friends in the FI. Each node may add (or remove) a friend from the FI, send a personal message to one of his friends, post an announcement to a subset of his friends, or follow a friend (i.e., subscribe to posts or content changes). To add or remove a friend requires only local changes in the FI. Additionally, to communicate with another friend a node n_i issues a $\textsc{FriendComm} = (id(n_i), msg, f)$ message containing the identification of n_i, a free text bulletin, and a flag indicating the type of message (post or follow). The node receiving a $\textsc{FriendComm}()$ message makes the message content available to the user.

3.4 Prototype Implementation

The $\mathcal{DS}^4$ prototype system[5] is build upon Microsoft .NET Framework v4.0 using C# and the Lucene v2.9.1.2 library for performing the content clustering and interest identification. Figure 2(a) shows a high-level view of a $\mathcal{DS}^4$ node and the different types of services implemented. A user in $\mathcal{DS}^4$ may utilise the node join service to connect to the social network and invoke interest creation to automatically cluster the content to be shared and identify one of more user interests. Additionally, the user may also manage his own collection in the local index store and add, remove, or modify the content or the metadata (e.g., tags). Interest creation may be invoked by the user when a significant amount of content in its local store has changed, or when the user wants to add/remove an interest. Apart from sharing the content with the rest of the community,

[5] http://www.uop.gr/~praftop/ds4/

the user may use the query processing service to issue Boolean, multi-keyword, and wildcard queries on the shared (meta-)data and discover new content. The content discovery process is automatic and returns (i) relevant results from the users' local store and (ii) content created by friends (social search) or nodes specialising on the query topic (semantic search). Finally, a user may refine/refresh its connections manually by invoking the semantic grouping service at any time. All actions are facilitated through a graphical user interface (Figure 2(b)).

4 Experimental Evaluation

The experiments in this section are designed to demonstrate the necessity and importance of semantic friendship in a distributed social network, and showcase the feasibility and qualitative benefits of our design. In the next sections, we show that content searching restricted only in social friends (protocol FI) is not effective in distributed social networks (returns only 10% of the relevant content), while small improvements are observed when we maintain the interests of social friends and use them at query time (protocol FI+i). We also show that by utilising the notion of semantic friendship and creating groups of nodes with similar interests (protocols SI+g and FISI+g) recall is significantly improved (we receive up to 80% of the relevant content), while keeping message traffic low. Finally, we demonstrate that our prototype implementation –apart from supporting fundamental social interactions– may be also used as a full-fledged content search engine in a distributed social network setting.

4.1 Simulation Measurements

Set-Up and Measures. The search efficiency and effectiveness of $\mathcal{DS}^4$ and the effect of semantic friendship have been tested on a data set derived from a home crawl of a real-life social network. For our simulations we used a crawl of the Delicious social bookmarking site performed in November 2011, comprised of about 700K users, 10M tags, and 21M bookmarks belonging to 170K categories. The social graph followed power-law distribution in social friendships, with the average number of social friends being 11. Notice that social friendships in Delicious are not symmetric, i.e. user A following (being a friend of) user B does not imply that B is also a friend of A. The average overlap between two social friends was 15% in friends, 17% in tags, and around 5% in data (bookmarks).

In each graph of this section, we randomly selected a part of 200K users from the user graph of our data set as nodes in the distributed social network, and averaged our results over 10 different network topologies and 100 queries per network. Experiments were performed considering bookmarks as resources and the corresponding tags as resource descriptions. The set of resource descriptions in a node was used as the node description. All descriptions were represented as term vectors, after applying case folding, stemming and stopword removal. The queries employed in the evaluation are tags that were strong representatives of bookmark categories. Cosine similarity over term vectors was used to calculate

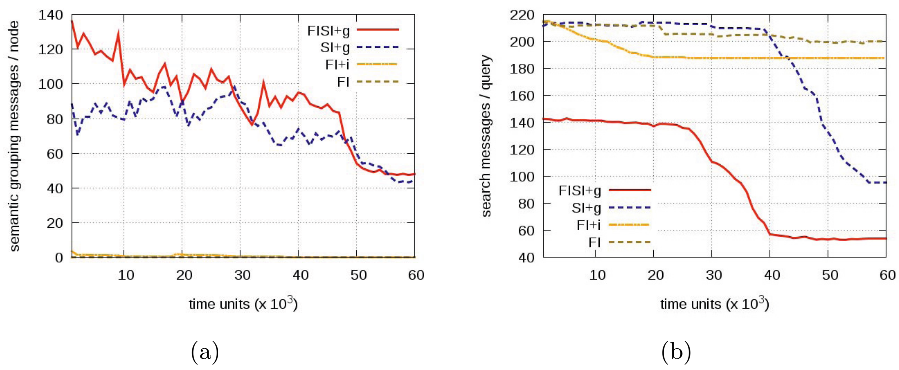


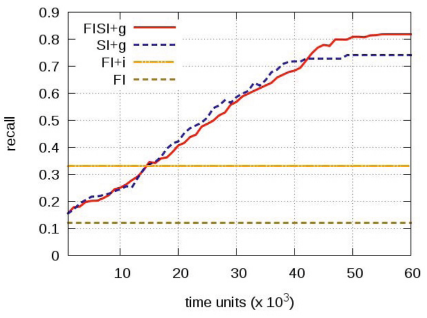
Fig. 3: (a) Semantic grouping messages/node and (b) search messages/query over time

the similarity among nodes and between a query and a node. Each node periodically tries to improve its semantic friends by initiating a semantic grouping procedure. The base unit for time used is the period t; the beginning of the semantic grouping procedure for each node is chosen uniformly at random from the time interval $[0, 4Kt]$, and its periodicity is selected in the same way from the time interval $[0, 2Kt]$ (and differs for each node). The simulation was run for all four protocols described in Section 3.1.

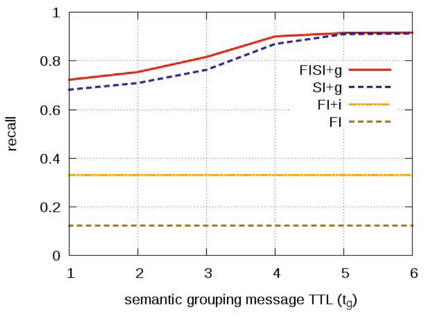
The performance of $\mathcal{DS}^4$ is evaluated in terms of the effectiveness of the retrieval process and the communication load incurred (both for the semantic grouping and the query processing). The accuracy of retrieval is evaluated using *recall* (i.e., percentage of qualifying answers retrieved with respect to the total number of qualifying answers in the network). Notice that, in our setting, precision is always 100% since only relevant content is retrieved. Additionally, the *network load* is measured by the total number of messages (requests and answers to requests) exchanged by the nodes during semantic grouping or querying.

The simulator was implemented in C/C++ and all experiments were run on Linux. The baseline parameter values used are $|SI| = 20,\ \theta = 0.5,\ t_g = 3,\ t_q = 5,\ m = 3$; a parameter setup is better than another if it achieves better recall for less communication load. In the following, we present how the most important of the parameters affect both the effectiveness and efficiency of the retrieval; the rest of the parameter setup experiments are briefly discussed at the end of this section due to space reasons.

Communication Load. Figure 3(a) presents the number of semantic grouping messages per node over time. The plots presented in the figure correspond to the four protocols discussed earlier in the paper. As expected, the FISI+g and the SI+g protocols impose a need for node organisation, which in turn leads to message traffic at node grouping time. Compared to the protocols that do not reorganise node connections (FI+i and FI), FISI+g and SI+g load the network with messages to make nodes discover semantically similar friends and get organised in semantic groups. Because of this, the network initially presents a high message overhead indicating that nodes are still randomly connected and seek the network for semantic friends. Algorithm FISI+g imposes more traffic load at



(a)



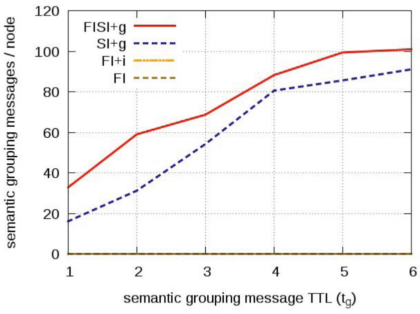
(b)

Fig. 4: Recall (a) over time and (b) versus t_g

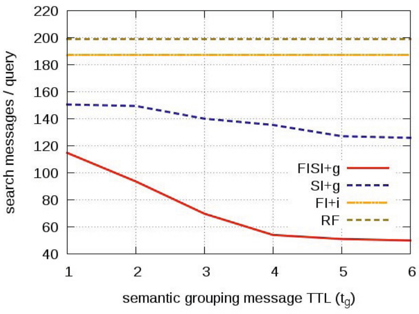
the network compared to SI+g, since nodes use both semantic and social connections to decide the initiation of semantic grouping. However in both cases, as nodes self-organise into semantic groups, message traffic is greatly reduced (over 55%). Notice also, that the FI+i protocol imposes the network with some minor message overhead (subject to the number of social friends of nodes) in order to update the nodes' FI with the interests of social friends.

Figure 3(b) shows the number of messages per query over time for the four different protocols. Initially, a high number of search messages is needed to retrieve the available data relevant to a query in all cases. However, this message overhead is decreased when one of the FISI+g or SI+g protocols is used (more than 55%) as nodes continuously get organised into groups with similar interests, which proves helpful at query time. The FISI+g protocol imposes the network with less communication load compared to the SI+g protocol, indicating that using both social and semantic connections helps (i) at grouping time towards discovering semantically similar nodes and create cohesive semantic neighbourhoods and (ii) at query time towards discovering the available data by exploring less node connections. Notice also, that the FI+i protocol shows better network performance compared to the FI protocol, indicating that using semantic information to forward the query to the social friends helps the query discover the available data.

Retrieval Effectiveness. Figure 4(a) illustrates the retrieval effectiveness of the network as a function of time for the different protocols. At the beginning, the values for recall are low (around 15%) for protocols FISI+g, SI+g, and FI, as nodes utilise only social friends (as no semantic grouping is yet initiated) to find the available data. After some time, when nodes start to organise into semantic groups with the FISI+g and SI+g protocols, we can observe the effect of the different strategies to retrieval effectiveness. The FI and FI+i protocols do not improve recall since network connections do not change over time, as they utilise only social friends. However, FI+i achieves 3 times better recall compared to FI, indicating that semantic information proves useful; even if search resorts only to social connections, using friends' descriptions to forward the query may improve



(a)



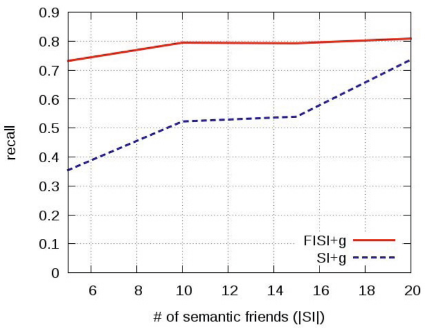
(b)

Fig. 5: (a) Grouping messages/node and (b) search messages/query when varying t_g

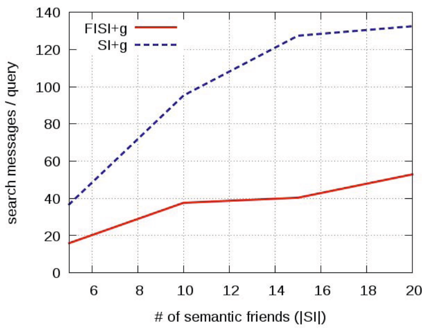
performance. The FISI+g and SI+g protocols demonstrate improved retrieval performance overall, achieving up to 2.5 (resp. 8) times better recall than FI+i (resp. FI). Besides, the FISI+g protocol outperforms SI+g achieving up to 10% better recall, indicating the usefulness of social friends in semantic grouping.

Varying t_g. Figure 4(b) investigates the retrieval effectiveness when social friends have been discovered (at time unit $50Kt$) for various values of t_g (i.e., the system parameter denoting how far the grouping message will be sent in the network). As expected, when either of the FI+i and FI protocol is used, recall is not affected by the semantic grouping message TTL ($t_g = 0$ for both protocols), since these protocols do not reorganise nodes' semantic connections. On the other hand, the FISI+g and SI+g protocols improve their retrieval performance (FISI+g by 26% and SI+g by 30%) when greater values for t_g are used, since nodes explore a larger part of the network, thus increasing the probability to discover nodes with similar interests and populate SI with more relevant semantic friends. Then, at query time, the issued queries have higher probability to find the available data by exploring the nodes' semantic connections. Because of this, the retrieval performance of the FISI+g and SI+g protocols converges for higher values of t_g, indicating that organising the network in cohesive semantic neighbourhoods may improve recall without resorting to social connections.

Figures 5(a) and 5(b) show the message overhead of the network (in terms of semantic grouping and search messages) when semantic friends have been discovered (at time unit $50Kt$) while varying the semantic grouping message TTL. As expected, communication load, when either the FI+i or the FI protocol is used, is not affected by t_g as nodes do not reorganise their semantic connections. In terms of grouping messages, the FISI+g and SI+g protocols cause more communication load in the network as t_g increases, since the message is sent further in the network to discover semantically similar friends. On the other hand, this further exploration of the network causes more cohesive semantic connections, which in turn improve (FISI+g by 2.3 times and SI+g by 1.2 times) communication load in terms of search messages. Recall that the FISI+g protocol uses more grouping messages (by using both semantic and social links) to discover



(a)



(b)

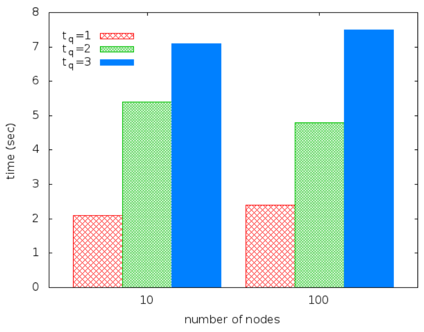
Fig. 6: (a) Retrieval effectiveness and (b) search messages/query when varying $|SI|$

semantic friends, and needs less search messages to find the available data as more cohesive semantic groups have been eventually formed.

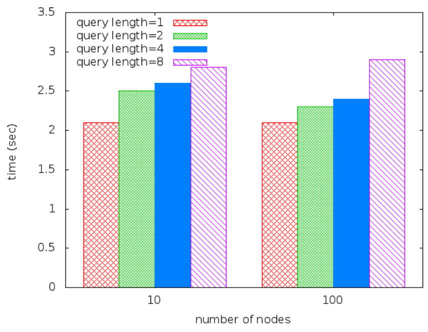
Varying $|SI|$. Figure 6(a) shows the retrieval effectiveness of the FISI+g and SI+g protocols when semantic friends have been discovered (at time unit $50Kt$) and how it is affected by the size of SI. Retrieval effectiveness is not demonstrated for the FI+i and FI protocols since they do not use semantic friends to find relevant data. The FISI+g protocol remains almost unaffected to changes in the number of semantic connections. This was expected since FISI+g uses both social and semantic connections to calculate SNS and forward the semantic grouping messages, explore a large part of the network, and achieve (an already) high recall. On the other hand, the SI+g protocol improves the retrieval performance (by 200%) when nodes store more semantic friends, since this protocol relies heavily on SI.

The number of search messages per query for the same setting is presented in Figure 6(b). Communication load at query time is affected by the number of semantic friends at each node for both the FISI+g and SI+g protocols; the network load increases when more semantic friends are stored, since queries are eventually broadcasted to the semantic neighbourhoods. Finally, the FISI+g protocol imposes the network with overall less communication load compared to the SI+g protocol as it manages to achieve a more cohesive semantic organisation.

Other Parameters. The way other system parameters affect the performance of $\mathcal{DS}^4$ is independent of the protocol used. For greater values of θ (similarity threshold) (i) semantic grouping messages are increased as the semantic grouping process is initiated more often, (ii) search messages are decreased as less nodes are considered similar to a query, and (iii) recall in increased as less data matches a query. Additionally, recall and search messages are linear to the increase in t_q in all cases, while semantic grouping messages remain unaffected to t_q variations. Moreover, recall and communication load are highly affected by increases in m (message fanout). Finally, node clustering affects both the retrieval effectiveness and the communication load (as also shown in [20]); these results carry over to our setting, but are omitted due to space considerations.



(a)



(b)

Fig. 7: Response time in $\mathcal{DS}^4$ prototype when varying (a) t_q and (b) query length

4.2 Prototype Measurements

Set-Up and Measures. Apart from the simulation results aiming at showing the effectiveness and efficiency of the $\mathcal{DS}^4$ protocols, we also conducted a series of experiments on the prototype implementation. We implemented $\mathcal{DS}^4$ upon Microsoft .NET Framework v4.0 using C# and used the Lucene v2.9.1.2 library for performing the content clustering and interest identification of the nodes. Our experiments were executed on a local-area network that consists of 10 workstations connected by a Gigabit ethernet connection running up to 10 nodes per machine, that is 100 nodes in total.

To obtain a content-rich set-up for the experimentation on the prototype, the $\mathcal{DS}^4$ nodes were populated with a subset of the OHSUMED TREC[6] collection containing 30,000 medical articles belonging to 10 different categories. Each node was randomly assigned documents from different categories so as to support *multiple interests* per node, and the queries were issued from random peers in the network. The network was allowed a small bootstrapping period to perform node grouping and was subsequently used for social networking and content retrieval. The response time shown in the graphs is wall-clock time between the time a user issues a query and the time he receives the final result ranking.

Response Times. Figure 7(a) shows the response time for different values of the t_q parameter and for two network setups (consisting of 10 and 100 nodes). As it was expected, the response times between networks of different size do not present a significant variation, as the dominant factor for introducing a delay is the increase of the query TTL. Increasing the query TTL results in reaching nodes several hops away from the query originator and also, in receiving more relevant results that have to be collected, ranked, and presented to the user.

In Figure 7(b) we observe the response time when varying the query length for the two network setups. Queries posed in this setting are comprised of multi-word terms under the VSM model that were matched against both the local document store of the node and the results received from other network nodes.

[6] `http://trec.nist.gov/data/t9_filtering.html`

Query length refers to the number of terms contained in an issued query. We observe that $\mathcal{DS}^4$ is not very sensitive to changes in the query length.

The extensive experimental evaluation of the prototype system also considered the message costs, achieved recall, and grouping quality and the findings were in trace with the simulation results. Our recall values were as high as 78%, while the search cost in terms of bandwidth was kept low (at all times less than 2MB/query for all the search messages issued for a specific query). Our network analysis showed that the cost of local search is 40-50% of the response time, the cost of the network communication was 30-40% of the time, while 10-20% of the time was the collection, ranking, and display of results. The full scale evaluation of the prototype is omitted due to space reasons.

Storage and Memory Requirements. In terms of storage and memory requirements, $\mathcal{DS}^4$ is kept lightweight. Its memory trace when idle is around 8.5MB (8756KB) while its storage costs are largely dominated from the size of the shared data. Only one instance of the Lucene library (with a memory trace of 556KB on average) is loaded in main memory and serves all nodes hosted in this machine. In our evaluations, we managed to run as many as 30 nodes in a single workstation; however the experiments were executed with up to 10 nodes per machine for avoiding bottlenecks.

5 Summary of Results and Outlook

We have presented a social networking system that introduces semantic friendship between users to facilitate content search on top of the default social interactions. Our experimentation shows that semantic friendship improves the observed recall of the system significantly, while search costs are reduced.

We plan to deploy and test $\mathcal{DS}^4$ in large-scale environments, study the applicability of foaf links to the replication problem, and extend the proposed protocols to cover also subscriptions over content/tags with aggregation.

References

1. Ng, C.H., Sia, K.C., Chang, C.H.: Advanced Peer Clustering and Firework Query Model in the Peer-to-Peer Network. In: WWW (2002)
2. Raftopoulou, P., Petrakis, E.G.M.: iCluster: A Self-organizing Overlay Network for P2P Information Retrieval. In: Macdonald, C., Ounis, I., Plachouras, V., Ruthven, I., White, R.W. (eds.) ECIR 2008. LNCS, vol. 4956, pp. 65–76. Springer, Heidelberg (2008)
3. Voulgaris, S., van Steen, M., Iwanicki, K.: Proactive Gossip-based Management of Semantic Overlay Networks. CCPE 19(17) (2007)
4. Loupasakis, A., Ntarmos, N., Triantafillou, P.: eXO: Decentralized Autonomous Scalable Social Networking. In: CIDR (2011)
5. Narendula, R., Papaioannou, T., Aberer, K.: My3: A highly-available P2P-based online social network. In: P2P (2011)
6. Sharma, R., Datta, A.: SuperNova: Super-peers based architecture for decentralized online social networks. In: COMSNETS (2012)

7. Mani, M., Nguyen, A.M., Crespi, N.: SCOPE: A prototype for spontaneous P2P social networking. In: PerCom (2010)
8. Mitchell-Wong, J., Goh, S., Chhetri, M., Kowalczyk, R., Vo, Q.: A Framework for Open, Distributed and Self-Managed Social Platforms. In: VECN (2008)
9. Abbas, S., Pouwelse, J., Epema, D., Sips, H.: A Gossip-Based Distributed Social Networking System. In: WETICE (2009)
10. Graffi, K., Gross, C., Mukherjee, P., Kovacevic, A., Steinmetz, R.: LifeSocial.KOM: A P2P-Based Platform for Secure Online Social Networks. In: P2P (2010)
11. Stuedi, P., Mohomed, I., Balakrishnan, M., Mao, Z.M., Ramasubramanian, V., Terry, D., Wobber, T.: Contrail: Enabling Decentralized Social Networks on Smartphones. In: Kon, F., Kermarrec, A.-M. (eds.) Middleware 2011. LNCS, vol. 7049, pp. 41–60. Springer, Heidelberg (2011)
12. de Spindler, A., Grossniklaus, M., Norrie, M.C.: Development Framework for Mobile Social Applications. In: van Eck, P., Gordijn, J., Wieringa, R. (eds.) CAiSE 2009. LNCS, vol. 5565, pp. 275–289. Springer, Heidelberg (2009)
13. Kabir, M.A., Han, J., Yu, J., Colman, A.: SCIMS: A Social Context Information Management System for Socially-Aware Applications. In: Ralyté, J., Franch, X., Brinkkemper, S., Wrycza, S. (eds.) CAiSE 2012. LNCS, vol. 7328, pp. 301–317. Springer, Heidelberg (2012)
14. Upadhyaya, B., Choi, E.: Social Overlay: P2P Infrastructure for Social Networks. In: NCM (2009)
15. Tegeler, F., Koll, D., Fu, X.: Gemstone: Empowering Decentralized Social Networking with High Data Availability. In: GLOBECOM (2011)
16. Liu, G., Shen, H., Ward, L.: An efficient and trustworthy P2P and social network integrated file sharing system. In: P2P (2012)
17. Han, L., Punceva, M., Nath, B., Muthukrishnan, S., Iftode, L.: SocialCDN: Caching techniques for distributed social networks. In: P2P (2012)
18. Mega, G., Montresor, A., Picco, G.: Efficient dissemination in decentralized social networks. In: P2P (2011)
19. Crespo, A., Garcia-Molina, H.: Routing indices for peer-to-peer systems. In: ICDCS (2002)
20. Raftopoulou, P., Petrakis, E., Tryfonopoulos, C.: Rewiring strategies for semantic overlay networks. In: DPD (2009)
21. Cutillo, L.A., Molva, R., Strufe, T.: Safebook: A privacy-preserving online social network leveraging on real-life trust. IEEE Communications Magazine (2009)
22. Gunther, F., Manulis, M., Strufe, T.: Key management in distributed online social networks. In: WOWMOM (2011)
23. Greschbach, B., Kreitz, G., Buchegger, S.: The devil is in the metadata - New privacy challenges in Decentralised Online Social Networks. In: PerCom (2012)
24. Xue, M., Carminati, B., Ferrari, E.: P3D - Privacy-Preserving Path Discovery in Decentralized Online Social Networks. In: COMPSAC (2011)
25. Backes, M., Maffei, M., Pecina, K.: A Security API for Distributed Social Networks. In: NDSS (2011)
26. Buchegger, S., Schioberg, D., Vu, L.H., Datta, A.: PeerSoN: P2P social networking: early experiences and insights. In: SNS (2009)
27. Manning, C.D., Raghavan, P., Schütze, H.: Introduction to Information Retrieval. Cambridge University Press (2008)
28. Sozio, M., Neumann, T., Weikum, G.: Near-optimal dynamic replication in unstructured peer-to-peer networks. In: PODS (2008)

Model-Driven Composition of Information Systems from Shared Components and Connectors

Stefania Leone[1,⋆], Alexandre de Spindler[2], and Dennis McLeod[1]

[1] Semantic Information Research Laboratory,
Computer Science Department, USC Los Angeles, CA, 90089-0781, USA
{stefania.leone,mcleod}@usc.edu
[2] School for Management and Law, ZHAW
CH-8400 Winterthur, Switzerland
alexandre.despindler@zhaw.ch

Abstract. We introduce CompIS, an approach, model and platform for model-driven component-based information system engineering. Our approach is based on the concept of shared components and connectors between them. To address the data-intensive nature of information systems, our components follow an extended model-view-control structure that also includes data. Component composition is based on configurable connectors, which define the collaboration logic between components and support component composition at the level of the component model, view, control and data. The CompIS UML profile allows to graphically define new components, connectors and compositions. The CompIS platform is a model-driven engineering environment, based on an extended object database that natively integrates the CompIS model. From graphical UML model definitions, the platform automatically generates application code that creates and initialises components and connectors. We present and validate our approach in the eCommerce domain.

Keywords: information system engineering, component model, model-driven engineering.

1 Introduction

Requirements towards an information system are subject to continuous change and evolution. For example, a company's initial version of their eCommerce solution may support standard online store functionality by means of product, customer and order management. Over time, they may desire to integrate support for electronic payment, extend the customer experience with support for product ratings and product recommendation, or may require more sophisticated product management functionality.

⋆ Stefania Leone's work is supported by the Swiss National Science Foundation (SNF) grant PBEZP2_140049.

While modern software engineering advocates a modular design and flexible development process to cater for such evolving requirements, information systems are still designed and developed in a rather monolithic and sequential way, where adaptation and evolution are not inherently supported. More recently, however, so-called community-driven development approaches [1] have become a popular way of providing more configurable and extensible information system platforms. In the eCommerce domain, popular open-source platforms, such as Magento Commerce[1] or osCommerce[2] follow this style of architecture. Their core, providing fully-fledged online-store functionality, can be modified, specialised and extended through extensions developed by and shared with the community. Popular Magento Commerce extensions offer support for payments via bank transfers[3] or integrate a blog[4]. While their extension mechanisms is flexible and powerful, extensions represent isolated units that only extend the core. There is no inherent mechanism that allows for flexible and modular composition scenarios, where one extension could be composed with another. For example, an extension offering advanced product management could not easily be combined with a product rating extension. The developer would rather have to get familiar with the code of both extensions and implement the composition code. In fact, these platforms typically do not even provide a systematic approach to the development of extensions and developers need to get familiar with the inner workings of the platform.

In this paper, we combine the approach taken by community-driven extensible information system platforms with ideas of component-based software engineering (CBSE). In contrast to CBSE, our component model is specifically targeted towards the data-intensive nature of information systems. At the same time, our approach overcomes the sequential nature of database design by introducing a modular and agile design process, where components may be composed at the model, view, control and data level, thus introducing modularity into the database. We build on a preliminary approach presented in [2], where we integrated support for component-based Web engineering into content management systems. The current work generalises our previous work, and introduces a model-driven approach to component-based information systems engineering.

The contributions of this paper consist of a refined, general component model for component-based information system engineering based on components and explicit connectors between them, a UML [3] extension that supports the model-driven engineering of components, connectors and compositions and our prototypical CompIS platform, which generates composed information systems from user-defined UML models and deploys it onto an extended object database that natively integrates the concept of components and connectors.

The paper is structured as follows. In Sect. 2, we present the background of our work. Section 3 introduces the general approach, followed by the design

[1] `www.magentocommerce.com`
[2] `www.oscommerce.com`
[3] `www.magentocommerce.com/magento-connect/bankpayment.html`
[4] `www.magentocommerce.com/magento-connect/blog-community-edition.html`

process in Sect. 4. In Sect. 5, we introduce the CompIS component model and in Sect. 6 we present the CompIS UML profile. The model-driven CompIS platform is presented in Sect. 7 and we give concluding remarks in Sect. 8.

2 Background

To facilitate the software design and evolution, CBSE [4, 5] postulates a modular and systematic construction of software from reusable components that can be extended, adapted and replaced. The cornerstone of CBSE is the underlying component model that defines how components are specified, constructed, assembled and deployed [6]. A component typically exhibits a set of functions and data through a well-defined interface and hides implementation details. Service-oriented architecture (SOA) [7] can be seen as a continuation of CBSE, explicitly addressing the requirements of loosely coupled, standards-based, and protocol-independent distributed computing. Services are reusable, self-contained, autonomous units with a well-defined interface, and are capable of communicating with each other via messages. Services are published to a repository by service providers and can be retrieved and consumed by service consumers.

While CBSE and SOA only offer composition at the message level, application composition may also take place at other levels. For example, a recent work in the area of web service composition proposed to also include the distributed orchestration of user interfaces (UI) [8]. They focus on extending web service standards such as WSDL [9] and BPEL [10] to integrate support for so-called UI components representing complete web applications. The extended orchestration supports the distributed synchronisation of UI components and services.

There have also been proposals for component-based database design, e.g. [11–13]. Thalheim [11] proposes the use of composable subschemas and other meta-structures to support the modelling and management of large and complex database schemas. Such components are database schemas that exhibit a similar structure and define import and export interfaces for connecting them. They also support incremental database development between versions of databases of different development phases through a mechanism that supports the importation of data from one version of a database into another, either as read-only or fully modifiable, in order to support incremental system design [14]. In our previous work [13], we showed, how structural composition can be supported through native constructs of the data modelling language.

Our current work combines composition approaches at various levels and introduces a modular and flexible approach to the development of information systems. We offer support for component-based information system engineering, where components can be composed at various levels, depending on the nature of the actual composition scenarios. We support, for example, the composition of a currency converter component with an order component at the level of the component view to integrated currency conversion into the user interface of the order component. At the same time, an electronic payment component can be composed with an order component at the message level to invoke the payment process upon order completion, while a product and rating component is structurally composed at the model level to allow products to be rated.

As stated in [15], one of the main challenges of modular system development lies in the fact that modular units may not be compatible for composition. As a consequence, the CompIS model is inspired by the Architecture Description Language (ADL) [16, 17], an approach to CBSE, where the component model consists of components and explicit connectors between them. Through the definition of explicit connectors between components, we circumvent the problem of component incompatibility. Connectors encapsulate the composition logic, exhibiting functionality ranging from simple message passing, to complex collaboration logic, such as data transformation operations, and, consequently, would allow for the composition of arbitrary components.

In line with model-driven engineering approaches for CBSE and SOA, e.g. [18–20], we have realised the CompIS model as a UML extension that refines the UML component and class models with a number of stereotypes representing the concepts defined by the CompIS model. Developers design new components and connectors using the CompIS stereotypes. We also provide a set of default connectors that can be reused and configured through model refinement and by means of Object Constraint Language (OCL) expressions [21] to realise a specific composition scenario. From the UML model definitions, our CompIS platform automatically generates application code that configures default connectors for particular compositions, and also generates and instantiates newly defined components and connectors. The generated code is deployed onto an extended object database that natively integrates our component model. By using an object database, components and connectors are defined and handled uniformly, as objects exhibiting data, structure and application logic.

Note that in contrast to model-driven Web engineering approaches, such as WebML [22], which prvide models for specifying the structure, navigation and presentation of web applications, we focus on component-based information systems engineering in general. We provide an approach to compose information systems in a model-driven way from components, representing small application units defining a model, view, control and data, and configurable connectors between them.

3 Approach

Our approach is motivated by the need to develop information systems in the context of complex and evolving requirements. For example, the development of an eCommerce platform may start with basic support for products that can be ordered by customers. In a second iteration, the system may be further extended with additional shipment and payment options, followed by advanced user experience support in the form of product ratings and recommendations.

Information system components targeting such individual requirements may already exist, either within the company's information system infrastructure or publicly available, for free or purchase. For example, a product supplier may share a product management component offering management of and access to their products. Therefore, it is of utmost importance to offer developers means and methods to construct their systems iteratively, ideally from shared components, to respond to complex and evolving requirements. Our approach targets such scenarios and supports the model-driven composition of information systems from reusable components and connectors between them. We will first give an overview of the CompIS model before we show, based on an eCommerce example, how components can be composed in a model-driven manner.

The CompIS component model is inspired by ADL, defining compositions by means of explicit connectors between components. Connectors enable composition and define the collaboration logic between components. To address the data-intensive nature of information systems, the CompIS component structure follows an extended MVC pattern that includes data. As a consequence, composition may take place at the level of the model, view, control or data. At all four levels, a component may expose so-called connection points used for compositions by the connectors. Figure 1 shows two example compositions in the

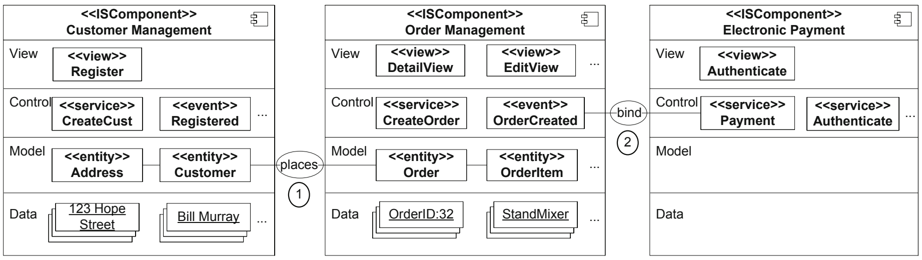


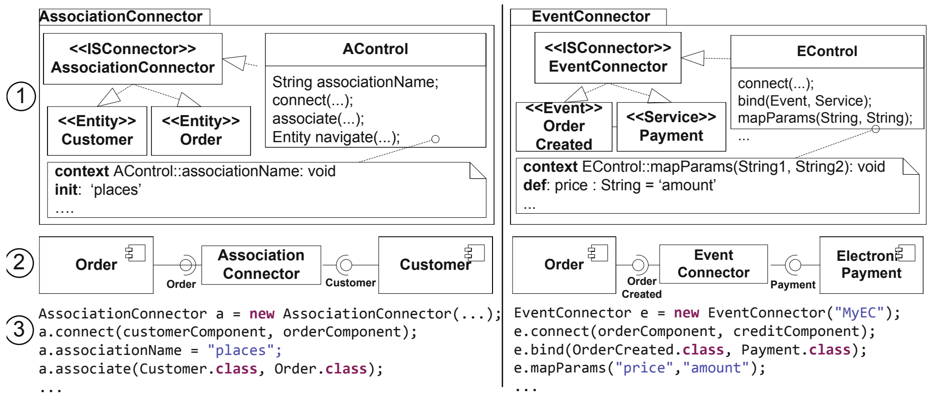
Fig. 1. Composition Example

eCommerce domain. Components are represented using the white box view of the UML component model, stereotyped with `<<ISComponent>>`. At each level, a component may expose connection points. For example, the customer management component on the left, exposes the `Register` view for composition at the view level, the service `CreateCust` and the event `Registered` for control level composition, the entities `Address` and `Customer` for composition at the model level, and the set of all `Customer` and all `Address` objects for composition at the data level. Similarly, the order management component, shown in the centre, offers a number of connection points at each level, while the electronic payment component, shown on the right, only offers connection points at the control and view level.

In (1), a composition at the model level is shown, where an association `places` is created between the `CustomerManagement.Customer` entity and the `OrderManagement.Order` entity. In (2), a composition at the control level is shown, binding the invocation of the `ElectronicPayment.Payment` service to the `OrderManagement.OrderCreated` event.

In Fig. 2, we show the steps involved in realising these compositions. On the left, the customer management and the order management components are composed through an association connector, and on the right, the event connector composes the order management and electronic payment components. Both connectors are default connectors provided as part of our model. They can be configured by the developer to define a specific composition. For the composition, we make use of refined UML component and class models.

The UML component model represents software systems by means of components and their relationships. A component's behaviour is specified in terms of provided and required interfaces. Composition is realised by connecting components over required and provided interfaces that match. Components are realised through explicit classes defined in the scope of an associated class model that defines a component's inner workings, such as the implementations of the provided interfaces. In analogy to this definition, CompIS components and



```
AssociationConnector a = new AssociationConnector(...);
a.connect(customerComponent, orderComponent);
a.associationName = "places";
a.associate(Customer.class, Order.class);
...
```

```
EventConnector e = new EventConnector("MyEC");
e.connect(orderComponent, creditComponent);
e.bind(OrderCreated.class, Payment.class);
e.mapParams("price","amount");
...
```

Fig. 2. Connector Configuration

connector are specified by means of a package where components and connectors are represented as UML components with an associated class model defining the realisation of the component. To configure a connector, the developer first refines its class model (1). If a connector composes components through *required* connection points, the developer associates the connector to those connection points via a `usage` association. In cases where a component is composed over a *provided* connection point, the developer has to realise the required interface as part of the connector definition, by means of a class that implements the methods and attributes defined by the connection point.

The composition itself is defined through an additional component model (2), modelling the composition by means of the two components and a connector between them. The composition is defined using the compact component view, where connection points are represented as interfaces. The UML socket notation stands for required and the lollipop notation for provided connection points. This implies that a component provides a connection point required by the connector and vice versa. Finally, from these two models, the code generator outputs initialisation code that configures the connector for the specific compositions (3).

The association connector, shown on the left in Fig. 2, creates an association between `Entity` connection points of two different components and, upon instantiation, allows to associate objects across components and to navigate along the association from one object to another. The association connector is represented by the `AssociationConnector` package. The `AControl` class realises the `AssociationConnector` defining its functionality. To define the composition, the association connector class model is refined. The `AssociationConnector` uses the two connection points, defined in the scope of their respective components, specified through a `usage` associations between the `AssociationConnector` and the `Customer` and `Order` entities. Note that the connection points are shown in the scope of the connector's package for the sake of this example, but are defined in the scope of their respective components' class models. `AControl` is further configured through user-defined OCL expressions. The OCL constraint shown in the example has been defined in the context of the `AControl.associate` method and consist of an attribute initialisation specifying the `name` of the association as 'places'. In a second step, the developer creates the component model, shown below the class model, which realises the composition. The association connector has been configured to require the `Customer` and `Order` connection points provided by the order management and customer management components. The code snippet, at the bottom left, defines the instantiation of the association connector and the invocation of the `associate` method with the defined association name and the two entity connection points passed as Java classes.

On the right of Fig. 2, the event connector configuration and instantiation is shown. The event connector handles the binding of an event of one component to a service of another component and takes care of the event registration and service invocation. The event connector class model is configured to require the connection points `OrderCreated` event from the order management component and `Payment` service from the electronic payment component, expressed by `usage` associations. The `EControl` class realising the `EventConnector` is further configured through OCL constraints. The OCL constraint associated with the `EControl.mapParams` method defines a variable definition `price = 'amount'` specifying the mappings between an event object attribute and a service parameter. Below, an excerpt of the initialisation code is illustrated, where the event connector is instantiated, and the `bind` method is invoked, passing the event and service as Java classes. Note that service classes define a method `invoke` which triggers the invocation of the service. The parameter mapping is based on OCL variables that map attributes to parameters by name. Upon service invocation, we make use of reflection to realise the actual mapping.

Note that developers are free to further extend the connectors with additional functionality. For example, the event connector may also define a user interface that redirects the user from the order to the payment process and back.

4 Component-Based Information System Engineering

Having introduced our approach, we will explain in more detail, how developers build their information systems in a model-driven and iterative manner from

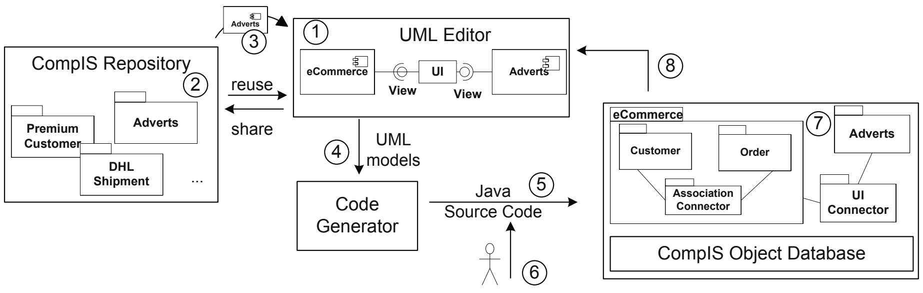


Fig. 3. CompIS Model-Driven Development Process

shared components and connectors. Figure 3 gives an overview of the CompIS development process and the involved components of the CompIS platform. A developer models and composes an information system by means of UML models using the UML editor provided by the CompIS platform (1). Our approach follows a community-based development paradigm. Components may either be designed from scratch by the developer, or imported from the community repository (2), where shared components and connectors are offered for reuse. During the design process, developers may browse the repository and import shared components and connectors into their local editor to compose them with other components. In the current example, we assume that the developer composes a local eCommerce component with a newly imported adverts component (3) using a UI connector.

Once the developer has modelled the application, the UML model descriptions are passed to the code generator (4). From the model description, the code generator generates Java source code realising the composed information system (5). Note that imported components and connectors consist of UML models and associated source code. Therefore, for the current composition, only the initialisation code of the imported adverts component and the UI connector has to be generated. However, if a developer designs new components and connectors from scratch, we generate the source code realising the component or connector structure, as well as initialisation code. Application logic not expressible via UML, such as method bodies, has to be manually added to the generated source code by the developer (6). To facilitate this task, the source code files are marked with annotations, where manual coding is needed.

The generated Java source code is deployed and runs on top of the CompIS object database (7). The CompIS object database is an extended object database that builds the basis for the deployed code providing native persistence for components, connectors and their data. The currently deployed application consists of a hierarchically composed eCommerce component that has been structurally composed from an order and customer component using an association connector. In the current iteration, the eCommerce component has in turn been composed with the advert component using a UI connector. The last composition results in an extended eCommerce main view that includes adverts.

As seen in the running example, such composition scenarios are typically the result of iterative development steps addressing new and evolving requirements. To cater for such evolving requirements and support continuous evolution, at any point in time, the developer may further extend or adapt the deployed information system by starting a new modelling iteration (8). In every iteration, the deployed application can be extended with new components or existing ones can be adapted or replaced. Also, developed components can be shared through the community repository. In this way, information systems are constructed modularly and collaboratively, from shared components and connectors.

5 Component Model

The CompIS component model supports the component-based design of information system. The model primitives have been designed in such a way that components represent fully functional (sub-) information systems and may be composed to build more complex information systems. Figure 4 gives an overview of our component model by means of a metamodel using UML notation. The part shaded in grey, highlights the basic component structure.

The component structure follows an extended MVC pattern that includes data. Each component may define a model, i.e. the data structure, a view defining the user interface, a control defining the application logic, and data structured according to the defined model, illustrated by the aggregation between `Component` and the corresponding parts. Note that components do not have to define all parts, but could only define a model and control, but no data and view.

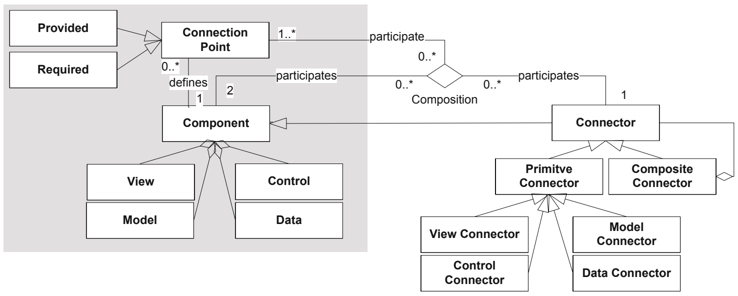


Fig. 4. Component Metamodel

To enable composition, each component defines connection points. In analogy to CBSE, we provide two types of connection points: required and provided connection points. A provided connection points represents a utility a component offers to the rest of the system. They can be seen as a signature of the component and there is no need to know about the inner workings of the component in order to make use of it. If a component needs to use utilities provided by another component in order to function, it defines required connection points. For example, an order management component may not include support for order shipment, but define a shipment interface connection point to be implemented by a third-party. A DHL shipment component may realise the required functionality and be composed using an appropriate connector that realises the collaboration logic. While components that only expose provided connection points are self-contained applications that run independently, components with required connection points need external help/support in order to function. It is therefore recommended to make sparse use of required connection points, or to opt for a different modularisation to reduce dependencies between components and thus, reduce complexity.

Figure 5 gives an overview of the currently supported connection points, grouped by level. Required connection points are shaded in grey. Note that required and provided connection points could be situated at any level. At the view level, `View` connection points represent composable view elements that expose a certain functionality or data through their user interface.

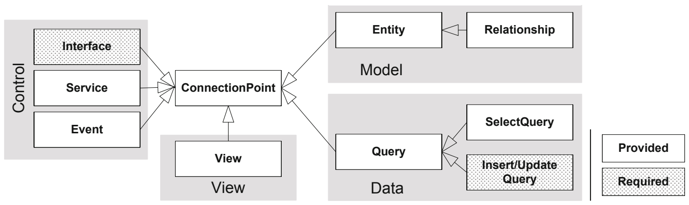


Fig. 5. Connection Point UML model

At the control level, we offer three types of connection points. `Event` connection points represent an event within a component, upon which another component can react. The event object gives access to the type of event and the involved (data) object(s). `Service` connection points define functionality offered by a component that can be invoked externally. `Interface` connection points are the counterparts of services and require an external component to provide the functionality defined by their signature of required methods and attributes.

At the model level, we rely on the ER model and its modelling constructs as connection points. `Entity` connection points give access to the structure of component data, i.e. the attributes and methods, and `Relationship` connection points represent relationships between entities. Note that a relationship is a specialisation of entity that defines a source and target attribute and methods allowing to navigate from one entity instance to a related one.

Finally, data connection points are represented by queries over the model. Such queries may be represented by SQL queries in a relational world, by XQuery expressions in an XML world, or code realising navigational data access in case of object databases. There are two types of query connection points: `Select Query` connection points give access to data, and thus are provided connection points, while `Insert/Update Query` connection points take external data as input, and thus are required connection points. To make use of a component's data in another component, a connector composes two components using a provided and required connection point, to export data from one component via select query and import it into the component using an insert/update query.

As illustrated in Figure 4, components may be composed through their connection points by explicit connectors, shown on the right. Depending on the connection points used for composition, a connector may be a data connector, a model connector, a control connector or a view connector. An association connector, for example, is a model connector, since it composes components at the model level by associating entity connection points. The connector, however, may define much more than a simple association between two entities at the model level. The connector may, for example, realise operations at the control level that allow to create, update and delete associations between entities as well as to navigate between entities along the associations. The connector may even define a view that allows such associations to be graphically created, and finally, also manage association data. Consequently, connectors may define a model, view, control and data, and are therefore defined as a specialisation of components, as shown by the specialisation association between `Component` and `Connector`.

We support composition of both components and connectors, since connectors are in turn components. The result of a composition is a new component (in fact, a connector), consisting of two sub-components plus newly defined component elements. For example, the eCommerce component has been composed from an order management and customer management component. The view of the eCommerce component is defined by the union of view elements defined by the respective components plus newly defined view elements. Generally, the composition interface of a composed component is built by the union of connection points defined by the sub-components plus newly defined ones.

Connectors may also be hierarchically composed, as illustrated by the composite pattern for connectors. For example, a composite connector between an eCommerce component and a recommendation component may consist of a data connector that composes the transformed eCommerce product and order data required by the recommendation component, and a UI connector which integrates recommendations into the eCommerce product overview view. Such composite components allow for arbitrary complex composition scenarios, where a composition may compose `n` sub-components.

To facilitate composition, we provide a set of default connectors that can be configured to adhere to a particular composition scenario. For each level, we provide default connectors tailored to compose components based on specific types of connection points. Such default connectors specify which types of connection points may be composed, and whether they are required or provided. Figure 6 gives an overview of the default connectors.

At the view level, we provide the composite connector that integrates two provided views into a composite view. With Java Swing, such views would

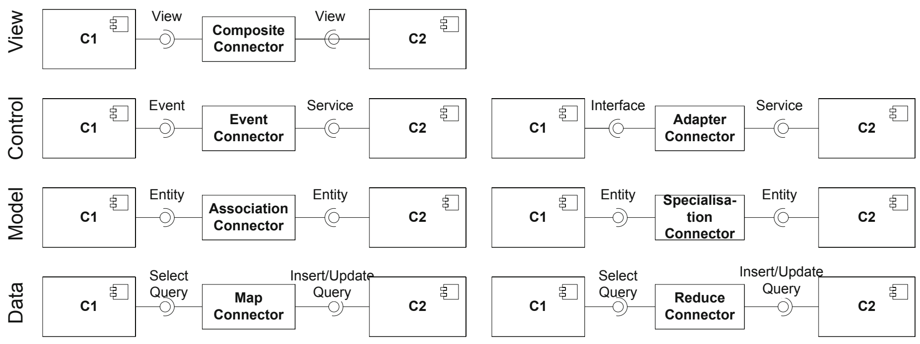


Fig. 6. Connectors

correspond to `JPane` instances that can be nested. In a web-based environment, a view may corresponds to a widget or `div` element in an HTML document.

At the control level, we provide the event connector, presented previously, that binds an event to a service, and invokes that service upon the event. The adapter connector is a realisation of the adapter design pattern. The connector connects components over an interface and a service connection point and forwards method invocation from one component to another. The connector translates internal calls to component A's interface into calls to component B's service, thus realising an implementation of interface A through service B.

At the model level, we rely on structural composition based on the underlying model's modelling concepts [13]. The association connector defines an association between two entity connection points and the specialisation connector defines an `isA` relationship between two entities. Since relationships are specialisations of entities, these two connectors may also act between relationships, thus allowing for n-ary relationships.

Finally, data connectors allow data from one component to be reused by another component. Data reuse may be defined by a mapping connector that maps the data structured according to one component's schema to the schema of another component, or by a reduce connector that transforms data from one component to a format specified by another component, thus enabling data reuse. Since connectors can be nested, data connectors can generally be defined as combinations of map and reduce functions.

As seen in Sect 3, default connectors are configured by extending and refining their class model. The connection points are specialised through usage or realisation associations from the connector class to the respective connection point classes, and through OCL constraints that further specify the composition.

6 CompIS UML Profile

UML 2.0 offers *UML Profiles* as a powerful extension mechanism that allows to extend, tailor or specialise UML to a specific domain [23]. Profiles are defined by

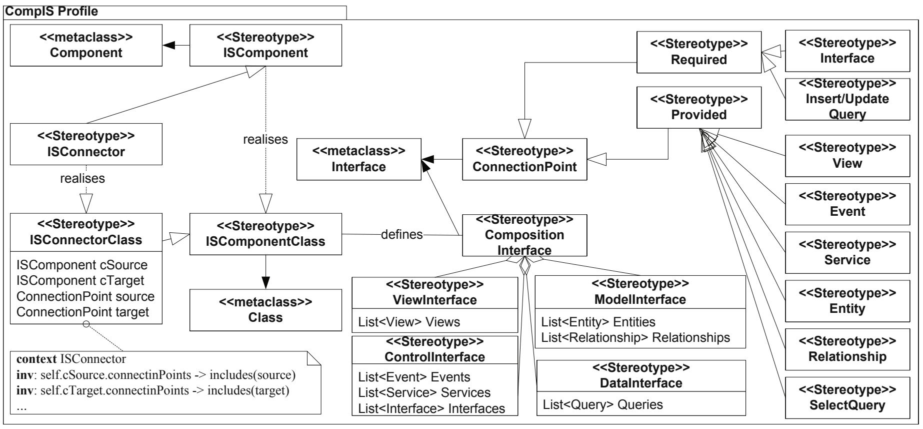

CompIS Profile
<<metaclass>> Component
<<Stereotype>> ISComponent
<<Stereotype>> ISConnector
realises
<<Stereotype>> ISConnectorClass
ISComponent cSource
ISComponent cTarget
ConnectionPoint source
ConnectionPoint target
context ISConnector
inv: self.cSource.connectinPoints -> includes(source)
inv: self.cTarget.connectinPoints -> includes(target)
...
<<Stereotype>> ISComponentClass
<<metaclass>> Class
defines
<<metaclass>> Interface
<<Stereotype>> ConnectionPoint
<<Stereotype>> Composition Interface
<<Stereotype>> ViewInterface
List<View> Views
<<Stereotype>> ControlInterface
List<Event> Events
List<Service> Services
List<Interface> Interfaces
<<Stereotype>> ModelInterface
List<Entity> Entities
List<Relationship> Relationships
<<Stereotype>> DataInterface
List<Query> Queries
<<Stereotype>> Required
<<Stereotype>> Provided
<<Stereotype>> Interface
<<Stereotype>> Insert/Update Query
<<Stereotype>> View
<<Stereotype>> Event
<<Stereotype>> Service
<<Stereotype>> Entity
<<Stereotype>> Relationship
<<Stereotype>> SelectQuery

Fig. 7. CompIS UML Profile

means of a UML model that specifies new model elements as so-called stereotypes that define properties and operations. Stereotypes can be used to introduce a new terminology, new syntax, new semantics and constraints. For example, a stereotype may refine and further constraint the definition of a UML type.

Figure 7 illustrates the CompIS UML profile. UML profiles are grouped as a package, in our case the CompIS Profile package. Our profile extends and refines concepts of the UML component model and class model. First, we introduced a new stereotype `ISComponent` that refines the UML `component` metaclass, as illustrated by the extension relationship between the two concepts.

`ISComponent` is realised through the `ISComponentClass`, which extends the UML `class` metaclass. Connection points are represented by the `Connection-Point` stereotype, which is a specialisation of the UML `Interface` metaclass, and further specialised into `Required` and `Provided` connection points. We define one stereotype for each type of connection point shown in Fig. 5. `ISComponentClass` defines a composition interface represented by the `CompositionInterface` stereotype, which is an extension of the UML `Interface` metaclass. The composition interface groups the required and provided connection points according to their composition levels (`ModelInterface`, `ViewInterface`, `ControlInterface` and `DataInterface`).

The UML component model introduces a connector concept linking components either as delegation or assembly. Since our connector concept is much more flexible, allowing to define complexe collaboration logic, we have introduced a separate concept `ISConnector` to define the collaboration logic between components. The `ISConnector` stereotype is a specialisation of `ISComponent`. It is realised through the `ISConnectorClass`, which in turn is a specialisation of the `ISComponentClass`. We make use of the standard UML connectors to link components and connectors using the lollipop notation to describe the composition through required and provided connection points.

While most of the newly introduced stereotypes define attributes and methods, which may also be constraint using OCL expression for further refinement, we generally omitted them in Fig. 7 for the sake of simplicity. Only the `ISConnectorClass` stereotype exemplifies attribute and constraint definitions. The class defines four attributes, referencing the composed components `cSource` and `cTarget` and the required and provided connection points used for composition. We have defined OCL constraints ensuring that the composed components define the specified types of source and target connection points required by the connector, since the UML editor would allow for arbitrary compositions.

7 Model-Driven Composition Platform

The UML Profile and default connectors have been realised as part of the CompIS platform for model-driven, component-based information system engineering. Figure 8 gives an overview of the platform architecture. Our platform

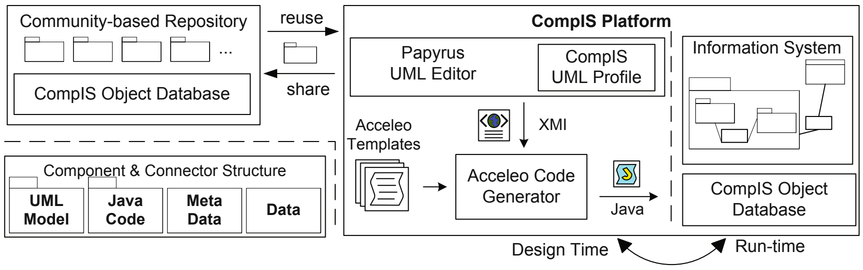


Fig. 8. CompIS Platform Architecture

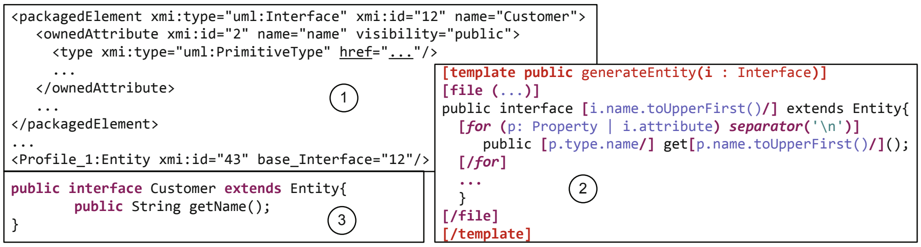
supports both the design and run-time of component-based information systems. At design-time, developers design their systems using a graphical UML editor and the CompIS UML Profile. We make use of the Papyrus Eclipse Plug-in[5], a graphical UML editor with support for profile definition, which we used to define the CompIS profile. A developer simply has to apply our profile to their model to get access to the defined stereotypes. This ensure that components, connectors and compositions follow our model description.

As part of the design process, developers may define their own components and connectors, or may reuse shared components and connectors from the community-based repository. Shared components can be imported into the developer's local platform and be used for composition. To make component and connectors shareable in a model-driven environment, their specification must include the component source code as well as the UML definition. Therefore, components and connectors are represented by the structure shown in the bottom left corner on Fig. 8, consisting of the UML model in the form of a XMI file, the source code

[5] `http://www.eclipse.org/papyrus/`

(Java files), metadata describing their purpose and connection points, and possibly also data. For example, a currency control component would be imported with actual currency conversion data that is regularly updated.

From the UML models, the code generator generates the composed information system. Code generation includes the generation and initialisation of newly defined components and connectors and the initialisation of imported components and connectors. For code generation, we rely on the Acceleo[6] plug-in, a model-to-text transformation tool following a template-based approach to code generation. Models, represented in the XMI format are transformed to text using user-defined templates.



```
<packagedElement xmi:type="uml:Interface" xmi:id="12" name="Customer">
   <ownedAttribute xmi:id="2" name="name" visibility="public">
     <type xmi:type="uml:PrimitiveType" href="..."/>
     ...
   </ownedAttribute>
   ...
</packagedElement>
...
<Profile_1:Entity xmi:id="43" base_Interface="12"/>
```

```
[template public generateEntity(i : Interface)]
[file (...)]
public interface [i.name.toUpperFirst()/] extends Entity{
  [for (p: Property | i.attribute) separator('\n')]
     public [p.type.name/] get[p.name.toUpperFirst()/]();
  [/for]
  ...
  }
[/file]
[/template]
```

```
public interface Customer extends Entity{
       public String getName();
}
```

Fig. 9. Template-based code generation

Figure 9 gives a simplified overview of the template-based code generation from XMI definitions. The XMI definition for a `Customer` entity connection point ① is of type `UML::Interface` and is linked to the `entity` stereotype via its XMI::ID (see last line of XMI file). In ②, an excerpt of the corresponding Acceleo code template is shown, defining the model to text transformation and the resulting Java code realising the connection point as Java interface extending the `Entity` interface ③. Note that the generated code may be extended by the developer. In fact, method bodies are left empty and marked with TODO annotations.

The code is generated for and deployed onto the CompIS platform. Note that there is no strict separation between run-time and design-time in the sense that we support continuous evolution of information systems. New components may be composed with ones that are already deployed as part of the information system, which allows for continuous evolution.

The CompIS platform is based on an extended object database which provides a metamodel extension mechanism introduced in [24]. This mechanism allows for extensions to be implemented as a native database facility rather than a layer on top of it. The database core provides the basic data management facilities used to model such extension modules. The core constructs are classes representing the concepts `Object`, `Class`, `Collection` and `Association`, extent collections for each one of these classes, associations among them and operators for the creation, retrieval, manipulation and deletion of class instances, depicted in Fig.10.

[6] `http://www.eclipse.org/acceleo/`

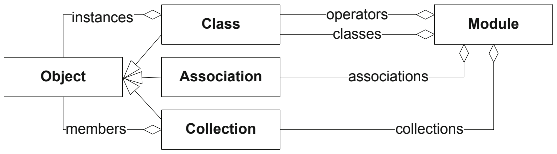


Fig. 10. Core Metamodel Concepts

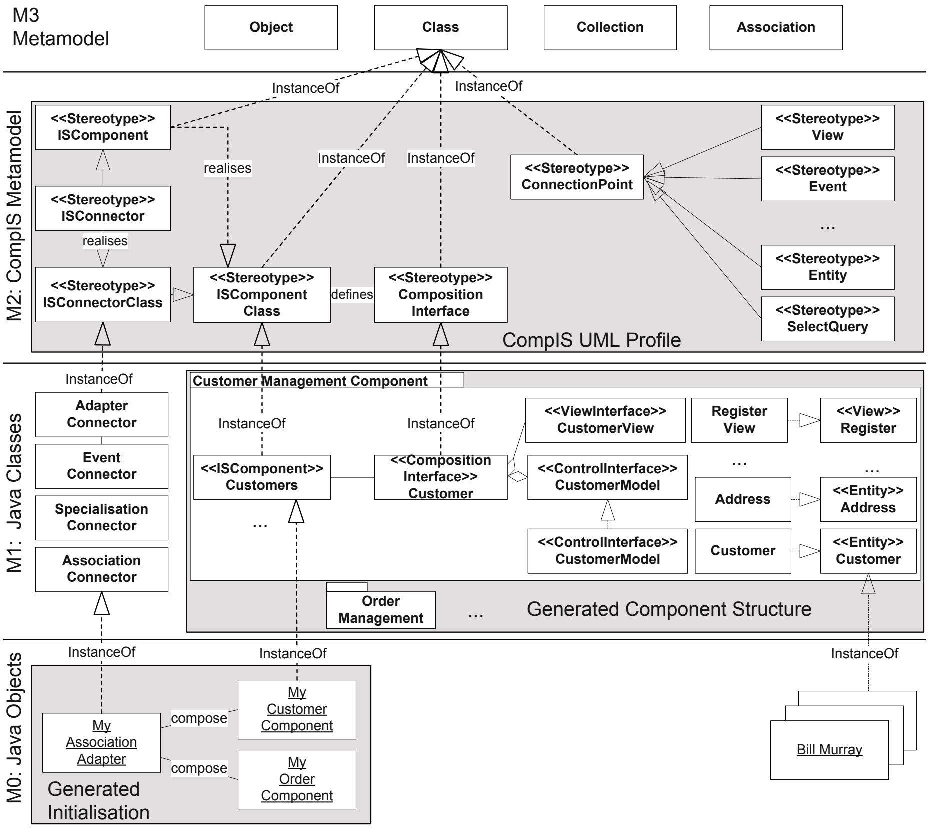


Fig. 11. MOF Levels

In order to support component definitions, compositions and deployment, we have defined a component module which implements the CompIS model by means of these core constructs. When the CompIS module is registered with the database, its classes, collections, associations and operators become part of the native database API. Consequently, we obtain a database supporting the definition, composition and deployment of components as described in Sect. 4.

Figure 11 is a summary of our approach and platform using the Meta Object Facility model (MOF) [25]. The M2 level represents the component metamodel described as UML profile. The metamodel has been realised as a metamodel extension module defined in terms of the object, class, collection and association concepts provided by the core module of the extended object database. Therefore, the core module of the object database establishes the M3 level.

The level M1 consist of the default connector and components represented by packages defining Java classes generated by our platform. The parts shaded in grey represent the source code generated from component models. In the example, the realisation of a customer component is shown. Finally, the M0 level defines the data managed by the components as well as component and connector instances configured for a particular composition through generated initialisation code.

8 Conclusion

We introduced an approach, model and platform for model-driven, component-based information system engineering, where information systems are designed from shared components and connectors. By encapsulating the collaboration logic into explicit connectors between components, we increase component compatibility and make our system resilient to component updates.

While we currently have focused on the definition of UML structure models, in the future, we plan to also look into extending interaction and behaviour models. This would support the model-driven definition of more complex collaboration logic, a more comprehensive system design and ultimately, also further reduce the amount of manual code development. We want to note that while the Papyrus UML editor is very powerful, its usability, especially when designing models based on user-defined profiles, is rather tedious. A tighter integration of the UML editor and user-defined profiles would greatly improve a developer's productivity. Finally, while we have presented our approach based on an eCommerce scenario, the approach is general and applicable to other information system domains. By doing so, the need for new connectors that for domain-specific compositions may arise, which would further complete our work.

References

1. Kazman, R., Chen, H.M.: The Metropolis Model a New Logic for Development of Crowdsourced Systems. Commun. ACM 52(7) (2009)
2. Leone, S., de Spindler, A., Norrie, M.C., McLeod, D.: Integrating Component-based Web Engineering into Content Management Systems. In: Daniel, F., Dolog, P., Li, Q. (eds.) ICWE 2013. LNCS, vol. 7977, pp. 37–51. Springer, Heidelberg (2013)
3. Rumbaugh, J., Jacobson, I., Booch, G.: Unified Modeling Language Reference Manual, 2nd edn. Addison-Wesley Professional (2010)
4. Heineman, G.T., Councill, W.T. (eds.): Component-based Software Engineering: Putting the Pieces Together. Addison-Wesley Longman Publishing Co., Inc. (2001)
5. Crnkovic, I., Sentilles, S., Vulgarakis, A., Chaudron, M.R.: A Classification Framework for Software Component Models. IEEE Transactions on Software Engineering 37(5), 593–615 (2011)

6. Lau, K.K., Wang, Z.: Software Component Models. IEEE Trans. Softw. Eng. 33, 709–724 (2007)
7. Erl, T.: Service-Oriented Architecture: Concepts, Technology, and Design. Prentice Hall PTR (2005)
8. Daniel, F., Soi, S., Tranquillini, S., Casati, F., Heng, C., Yan, L.: Distributed Orchestration of User Interfaces. Inf. Syst. 37(6), 539–556 (2012)
9. Christensen, E., Curbera, F., Meredith, G., Weerawarana, S.: Web Services Description Language (WSDL) 1.1. W3C Note (2001)
10. Organization for the Advancement of Structured Information Standards (OASIS): Web Services Business Process Execution Language (WS-BPEL) Version 2.0 (2007)
11. Thalheim, B.: Component Development and Construction for Database Design. Data Knowl. Eng. 54(1) (2005)
12. Casanova, M.A., Furtado, A.L., Tucherman, L.: A Software Tool for Modular Database Design. ACM Trans. Database Syst. 16(2) (1991)
13. Leone, S., Norrie, M.C.: Building eCommerce Systems from Shared Micro-Schemas. In: Meersman, R., et al. (eds.) OTM 2011, Part I. LNCS, vol. 7044, pp. 284–301. Springer, Heidelberg (2011)
14. Ma, H., Schewe, K.-D., Thalheim, B.: Modelling and Maintenance of Very Large Database Schemata Using Meta-structures. In: Yang, J., Ginige, A., Mayr, H.C., Kutsche, R.-D. (eds.) UNISCON 2009. LNBIP, vol. 20, pp. 17–28. Springer, Heidelberg (2009)
15. Shaw, M.: Modularity for the Modern World: Summary of Invited Keynote. In: AOSD 2011 (2011)
16. Medvidovic, N., Taylor, R.N.: A Classification and Comparison Framework for Software Architecture Description Languages. IEEE Trans. Softw. Eng. 26(1), 70–93 (2000)
17. Clements, P.C.: A Survey of Architecture Description Languages. In: IWSSD 1996 (1996)
18. Orriëns, B., Yang, J., Papazoglou, M.P.: Model Driven Service Composition. In: Orlowska, M.E., Weerawarana, S., Papazoglou, M.P., Yang, J. (eds.) ICSOC 2003. LNCS, vol. 2910, pp. 75–90. Springer, Heidelberg (2003)
19. Baïna, K., Benatallah, B., Casati, F., Toumani, F.: Model-driven web service development. In: Persson, A., Stirna, J. (eds.) CAiSE 2004. LNCS, vol. 3084, pp. 290–306. Springer, Heidelberg (2004)
20. Clemente, P.J., Hernández, J., Sánchez, F.: Extending Component Composition Using Model Driven and Aspect-Oriented Techniques. Journal of Software 3(1), 74–86 (2008)
21. Object Management Group (OMG): Object Constraint Language, OCL (2010), `http://www.omg.org/spec/OCL/2.2/`
22. Ceri, S., Fraternali, P., Bongio, A.: Web Modeling Language (WebML): A Modeling Language for Designing Web Sites. Computer Networks 33(1-6), 137–157 (2000)
23. Fuentes-Fernández, L., Vallecillo-Moreno, A.: An Introduction to UML Profiles. UPGRADE, European Journal for the Informatics Professional 5(2), 5–13 (2004)
24. Grossniklaus, M., Leone, S., de Spindler, A., Norrie, M.C.: Dynamic Metamodel Extension Modules to Support Adaptive Data Management. In: Pernici, B. (ed.) CAiSE 2010. LNCS, vol. 6051, pp. 363–377. Springer, Heidelberg (2010)
25. OMG: Meta Object Facility (MOF) Core Specification Version 2.0 (2006), `http://www.omg.org/cgi-bin/doc?formal/2006-01-01`

Collaborative Schema Matching Reconciliation

Hung Quoc Viet Nguyen[1], Xuan Hoai Luong[1], Zoltán Miklós[2], Tho Thanh Quan[3], and Karl Aberer[1]

[1] École Polytechnique Fédérale de Lausanne
{quocviethung.nguyen,xuan.luong,karl.aberer}@epfl.ch
[2] Université de Rennes 1
zoltan.miklos@univ-rennes1.fr
[3] Ho Chi Minh City University of Technology
qttho@cse.hcmut.edu.vn

Abstract. Schema matching is the process of establishing correspondences between the attributes of database schemas for data integration purpose. Although several schema matching tools have been developed, their results are often incomplete or erroneous. To obtain correct attribute correspondences, in practice, human experts edit the mapping results and fix the mapping problems. As the scale and complexity of data integration tasks have increased dramatically in recent years, the reconciliation phase becomes more and more a bottleneck. Moreover, one often needs to establish the correspondences in not only between two but a network of schemas simultaneously. In such reconciliation settings, it is desirable to involve several experts. In this paper, we propose a tool that supports a group of experts to collaboratively reconcile a set of matched correspondences. The experts might have conflicting views whether a given correspondence is correct or not. As one expects global consistency conditions in the network, the conflict resolution might require discussion and negotiation among the experts to resolve such disagreements. We have developed techniques and a tool that allow approaching this reconciliation phase in a systematic way. We represent the expert's views as arguments to enable formal reasoning on the assertions of the experts. We detect complex dependencies in their arguments, guide and present them the possible consequences of their decisions. These techniques thus can greatly help them to overlook the complex cases and work more effectively.

1 Introduction

Integrating data stored in autonomously-developed information systems is essential for a number of applications. Schema matching is the process of establishing correspondences between the attributes of two database schemas, which is crucial for data integration. There is a large body of work on schema matching techniques: a number of commercial and academic tools (so called *matchers*) have been developed [4, 40]. Although some matchers perform impressively on certain data sets, the heuristic nature of the matching algorithms prevent them from yielding completely correct results. In practice, data integration tasks often include a post-matching phase, called *schema matching reconciliation*, where human experts review, validate and correct the generated correspondences. Although this phase requires substantial efforts, it has received very little attention, with the notable exception of [13].

In real life scenarios, the data to be integrated is often stored in several distributed databases. The creation of a globally accepted schema or ontology is neither practical nor possible for certain cases and one has to construct a set of pairwise mappings between the involved schemas. Constructing pairwise mappings can bring a number of advantages, e.g. if new schemas are added or removed, one only needs to adjust the local mappings. One can also construct the mappings in a pay-as-you-go fashion, whenever they are needed. The pairwise mapping establishment has also disadvantages. In this case, the matching reconciliation task can be very challenging as we explain in the following motivating scenario. Our work proposes methods to minimize these difficulties.

1.1 Motivating Example

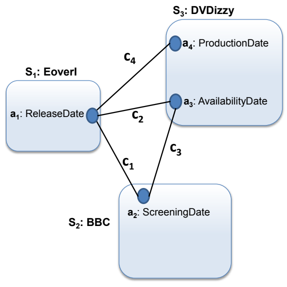
Let us consider the following scenario where three video content providers EoverI, BBC, and DVDizzy would like to create a shared website to publicize their offerings, which link back to the particular website for the purchases. The shared website needs information from the individual content providers (e.g. title, release date, main actors) so that consumers searching on the site can find the products they want. Although it would be conceivable to construct a global schema for the three providers, as more providers would join to this shared site, such a global schema could become impractical. We assume a scenario where the correspondences are established in a pairwise manner.

Figure 1a shows simplified schemas to illustrate this scenario. The three boxes represent the schemas of EoverI, BBC, and DVDizzy respectively. The figure shows four correspondences c_1, c_2, c_3, and c_4, generated by the matcher that we applied for each pair of schemas. As the names of the involved attributes are rather similar (ProductionDate, AvailabilityDate, ScreeningDate, ReleaseDate), typical matchers often fail to give the correct attribute mappings. Even human experts might disagree on which correspondences are correct. Moreover, as matchers only consider two schemas as input [4], they ignore natural expectations of the users with respect to the entire network. One can formulate these network-level consistency conditions as constraints:

- **One-to-one Constraint**. In some cases, one expects that each attribute of one of the schemas is matched to at most one attribute of any other schemas.
- **Cycle Constraint**. If the schemas are matched in a cycle, the matched attributes should form a closed cycle. This is a very natural requirement, if one would like to exchange data that is stored in the structure of the corresponding schemas.

Figure 1a shows some violations of these constraints, which are frequent in sets of automatically generated correspondences as most matchers do not take the constraints into account. For example, the attribute S_1.ReleaseDate is matched with two attributes S_3.ProductionDate and S_3.AvailabilityDate of schema S_3, thus violate the one-to-one constraint. The cycle constraint is violated also: S_3.AvailabilityDate is matched to S_2.ScreeningDate, which is then matched to S_1.ReleaseDate, and finishes at S_3.ProductionDate. Using such correspondences for data integration can lead to unwanted effects: AvailibilityDate and ProductionDate of DVDizzy would be mixed up.

If one would like to use the attribute correspondences for data integration or exchange, it is necessary to eliminate the errors from the set of correspondences.



(a)

	User Inputs	Arguments (derived from user inputs)
EoverI	approve c_1, c_2, c_3 disapprove c_4	$w_1^e = \langle\{c_1\}, c_1\rangle$, $w_2^e = \langle\{c_2\}, c_2\rangle$ $w_3^e = \langle\{c_3\}, c_3\rangle$, $w_4^e = \langle\{c_2, \neg c_2 \vee \neg c_4\}, \neg c_4\rangle$ $w_5^e = \langle\{\neg c_3 \vee \neg c_1 \vee \neg c_4, c_1, c_3\}, \neg c_4\rangle$ $w_6^e = \langle\{\neg c_4\}, \neg c_4\rangle$
BBC	approve c_3, c_4	$w_1^b = \langle\{c_3\}, c_3\rangle$, $w_2^b = \langle\{c_4\}, c_4\rangle$
DVDizzy	approve c_3, c_4	$w_1^d = \langle\{c_3\}, c_3\rangle$, $w_2^d = \langle\{c_4\}, c_4\rangle$

(b)

Fig. 1. The motivating example. (a) A network of schemas and correspondences generated by matchers. There are two violations: $\{c_2, c_4\}$ w.r.t. the *one-to-one* constraint, $\{c_1, c_3, c_4\}$ w.r.t. the *cycle* constraint. (b) An illustrated collaborative reconciliation between three video content providers: EoverI, BBC, and DVDizzy. The assertions (approvals/disapprovals) of BBC and DVDizzy are identical and different from those of EoverI.

This reconciliation process typically involves human input, which is a set of assertions that indicate whether a particular mapping, such as the correspondence between S$_3$.ProductionDate and S$_1$.ReleaseDate, should be approved or disapproved.

Until recently, this task was performed by a single expert. As the size of networks in data integration grows, the complex reconciliation tasks should be performed by not only one but several experts, to avoid the overload on a single expert and also to assign each expert the parts of the problem about which he is more familiar. In such cases, the experts might disagree about certain correspondences. If the application requires the network-level constraints, there could be rather complex dependencies among correspondences. For example, the choice of considering a correspondence *correct* can influence the possible choices for other correspondences. Moreover, the simple techniques for conflict resolution such as majority voting are not applicable, as the resulting set of correspondences would not comply to these constraints. To resolve these problems, the experts need to discuss and negotiate which correspondences to accept or reject. Because of complex dependencies in these networks, it is very challenging for the experts to overlook all possible consequences of their decisions. Thus on one hand it is highly desirable to split the reconciliation task, on the other hand combining individual results is very challenging. Our work addresses exactly this problem by proposing a number of services and a tool realizing those services to enable the collaborative process.

1.2 Contributions

In this paper, we leverage the theoretical advances and the negotiation nature of argumentation [5, 16] to support the work of multiple experts in the reconciliation task. This task, with the presence of consistency constraints, requires that the experts overlook a number of dependencies, which is very challenging without any support.

The specific contributions of our work are as follows. We model the schema matching network and the reconciliation process, where we relate the experts' assertions and the constraints of the matching network to an *argumentation framework* [16]. Our representation not only captures the experts' belief and their explanations, but also enables to reason about these captured inputs. On top of this representation, we develop support techniques for experts to detect conflicts in a set of their assertions. Then we guide the conflict resolution by offering two primitives: *conflict-structure interpretation* and *what-if analysis*. While the former presents meaningful interpretations for the conflicts and various heuristic metrics, the latter can greatly help the experts to understand the consequences of their own decisions as well as those of others. Last but not least, we implement an argumentation-based negotiation support tool for schema matching (ArgSM) [32], which realizes our methods to help the experts in the collaborative task.

Our paper is organized as follows. Section 2 consists the formulation of schema matching network reconciliation problem and the overview of our solution. Next, Section 3 discusses how to construct an argumentation framework for schema matching networks, while Section 4 deals with our guiding methods for conflict resolution. After that, Section 5 describes implementation details. Then, we present our tool, ArgSM, in Section 6. Section 7 presents the related work and Section 8 concludes the paper.

2 Model

In this section, we focus on modeling the collaborative reconciliation in schema matching. Firstly, we introduce the elements of a schema matching network, which is a network of schemas that are connected through pairwise mappings between them. Then we describe the process of collaborative reconciliation to validate the mappings of this network, which are generated by automatic matchers.

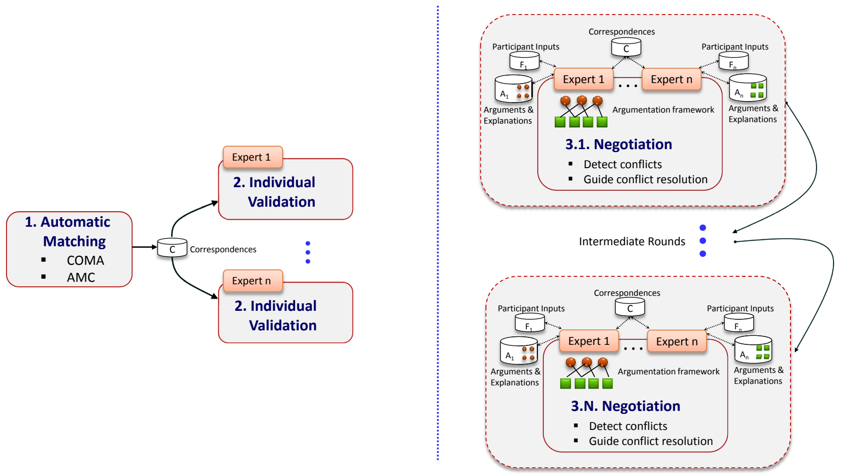
2.1 Schema Matching Network

We model a schema as a finite set of *attributes* $\{a_1, \cdots, a_n\}$. Let $\mathcal{S} = \{s_1, \cdots, s_n\}$ be the set of schemas of unique attributes ($s_i \cap s_j = \emptyset$ for all $1 \leq i, j \leq n$ and $i \neq j$) and $A_{\mathcal{S}}$ be the set of attributes in $\mathcal{S}$ ($A_{\mathcal{S}} = \bigcup_{i=1}^{n} s_i$). An *attribute correspondence* between a pair of schemas $s_1, s_2 \in \mathcal{S}$ is a pair of attributes (a, b) in which $a \in s_1$, $b \in s_2$. The set of all possible attribute correspondences is denoted by C. In addition, we have $\Gamma = \{\gamma_1, \ldots, \gamma_n\}$ as the set of integrity constraints, which expresses the intuitions observed from the schema matching problem. Combining the introduced notions, we define a *schema matching network* to be a triple $(\mathcal{S}, \Gamma, C)$.

In this paper, we consider a schema matching network $(\mathcal{S}, \Gamma, C)$, where C is typically the outcome of first-line schema matchers [20] such as COMA++ [1], AMC [37]. The schema matching reconciliation is the process of assessing the correspondences in C whether they are correct. The goal of the reconciliation process is to construct a maximal set of attribute correspondences ($M \subseteq C$) satisfying all constraints of Γ.

2.2 Collaborative Reconciliation

In practice, to obtain M, schema matching tasks often include a post-matching phase for correspondences to be reviewed and validated by experts. If the size of the schema matching network is large, the reconciliation task can be rather expensive. Moreover,



(a) Phase 1 - Individual validation (b) Phase 2 - Input combination

Fig. 2. The collaborative reconciliation process starts with a set of correspondences C generated by matchers. In Phase 1, each expert (user/participant) i is responsible for validating a particular set $C_i \subset C$. It is followed by Phase 2 that has multiple negotiation steps (3.1. to 3.N) to resolve conflicts in user inputs.

some experts might be more knowledgeable about some parts of the network, thus in these cases it is very natural to split the task among multiple experts.

The collaborative reconciliation as illustrated in Figure 2 is a two-phase process: *individual validation* and *input combination*. Let $\mathcal{E}$ denote the set of users $\mathcal{E} = \{E_1, \ldots, E_n\}$ who participate.

– *Individual validation:* In the first phase, each expert E_i is assigned to validate a subset C_i of C (usually of equal size). The assigned sets C_i usually overlap to a certain degree, thus there are correspondences assessed by several experts.
– *Input combination:* In the second phase, the individual inputs are combined. The goal of the collaborative reconciliation process is to construct a set of correspondences M that satisfies all constraints. If there are conflicting views about correspondences (for example, one expert considers correct while the other incorrect) then they need to come to a conclusion and chose which view to accept.

We leverage existing techniques from a large body of research [2, 27, 39] for the first phase. In this paper, we focus on the second phase. More precisely, we apply the theoretical advances of argumentation to detect conflicts in user inputs and guide them to resolve conflicts. Section 3, 4, and 5 will describe those functionalities in detail.

3 Argumentation for Schema Matching Networks

Let us consider a setting where several experts assess a set of attribute correspondences in a schema matching network. They might have different views whether a given correspondence should be correct or not. To complete the reconciliation task, they need to discuss and resolve these conflicts to obtain a globally consistent set of correspondences. The conflicts between the different user views can be rather complex in the presence of integrity constraints. We call a situation *direct* conflict if two experts disagree about a given correspondence (one of them thinks it is correct, while the other claims that it is incorrect). In the presence of integrity constraints, we can also talk about *indirect* conflicts. For example, in Figure 1, if we assume the one-to-one constraint between S_1 and S_3 and an expert considers c_4 correct, then c_2 must be incorrect (otherwise the constraint would be violated). We call a situation where a second expert thinks that c_2 is correct an indirect conflict.

3.1 Arguments for Schema Matching

In order to study the conflicts between the experts opinions, we will rely on argumentation techniques. Argumentation is a systematic study of techniques to reach conclusions from given premises [5]. We will use the standard representation of arguments [5] where an *argument* consists of a *claim* and a *support* that explains the respective claim. We represent arguments in the form $\langle\{support\}, claim\rangle$. In the case of logic-based argumentation, both the support and the claim are logical formulas, such that the support is a minimal set that is sufficient to prove the claim.

For the schema matching problem, given a set of correspondences C, we employ propositional logic to encode the user inputs (during the reconciliation process): if an expert asserts that a given correspondence c is *true*, then we can represent this by a propositional variable v_c (and the assignment $v_c = true$). To simplify our notation, sometimes we use c to denote the propositional variable (corresponding to a correspondence c). This representation also enables to represent the consistency constraints, for example $\neg(c_2 \wedge c_4)$ encodes the *one-to-one* constraint (the two correspondences c_2 and c_4 cannot be true at the same time).

If several experts assess a set of correspondences (or as it is more common, they work on a different but overlapping set of correspondences) then we can use this encoding to represent their individual input in the form of arguments. For example, in Figure 1, we can represent the input assertion of an expert by the argument $w_1^e = \langle\{c_1\}, c_1\rangle$, where the claim is that the c_1 is correct, that is based on the simple support that is the knowledge of the expert about the correctness of c_1. A more complex example is the argument (that is the claim of an expert, together with a support) $w_4^e = \langle\{c_2, \neg c_2 \vee \neg c_4\}, \neg c_4\rangle$. This can be interpreted as follows: the expert has approved c_2, he would like to avoid violating the *one-to-one* constraint ($\neg c_2 \vee \neg c_4$), he disapproves c_4. We give a more complete list of arguments of three experts in Figure 1b (with respect to one-to-one and cycle constraints). In our work, the claim of an argument is always a single propositional clause, while the support is a set of propositional formulae.

3.2 Understanding the Conflicting Arguments

This representation enables us to explain more precisely the direct and indirect conflicts. If the claim of two arguments w_1 and w_2 contradict each other (together they form an inconsistent set of formulae) then we say that the arguments w_1 and w_2 are in direct conflict. In argumentation terminology this is called *rebuttal*. If the claim of an argument

w_2 appears in a negated form in the support of w_1, we talk about indirect conflict. In argumentation terms, w_1 *undercuts* w_2. In the following, we will consider following attack relation: w_1 *defeats* w_2 if w_1 either undercuts or rebuts w_2. For example, w_1^b attacks w_3^e and the argument w_1^d rebuts the argument w_5^e (Figure 1b). A set of arguments and attacks between the arguments $\langle A, R\rangle$ is called an argumentation framework [16].

Constructing argumentation framework $\langle A, R\rangle$ for a reconciliation problem from the arguments of all the experts $A = \bigcup A_i$ with the above attack relation has several advantages. In particular, computing the attack relation we can detect each problem that might exists in the experts' inputs. The argumentation framework enables even more complex tasks that we explain in the following sections.

4 Guiding the Conflict Resolution

We have presented in Section 3 how to construct an argumentation framework for the schema matching problem, where multiple experts work on the reconciliation task. This not only enables to reason about the user input, but also to detect conflicts and determine the reasons for these problems. In this section we focus on techniques that exploit this information to guide the experts in resolving these conflicts.

In particular, we describe here two services that can largely help the collaborating experts. These are the (1) the *interpretation of conflict structures*, with which we can present meaningful interpretations for the conflicts together with some associated metrics that can largely support the negotiation of the experts and the (2) *what-if analysis*, with which we can compute (and in the tool visualize) the consequences of a particular potential decision. The details of these services are provided in what follows.

4.1 Interpretation of Conflict Structures

Given a potentially conflicting set of assertions from several experts who collaborate on the reconciliation task, we can analyse the structure of conflicts and compute (qualitative) metrics that explain the conflicts as well the potential ways to resolve the problems.

Extensions. An *extension* $\epsilon \subseteq A$ of an argumentation framework $\langle A, R\rangle$is an acceptable set of arguments ϵ; i.e., a set of arguments that can be accepted simultaneously, that is, they do not contain any conflicts. In fact, there are many possible ways to define an extension. The various strategies that can be used to construct such an exception are called *acceptability semantics* [16]. For instance, an extension following the *complete* semantics is a conflict-free set of arguments, defends all its arguments, and contains all arguments it defends. A set of arguments A' is conflict-free if there are no arguments $a_1, a_2 \in A'$ that a_1 attacks a_2. Meanwhile, a_1 defends a_2 if there exists a_3 that is attacked by a_1 and attacks a_2. Generally, there are more then one extension (for a given semantics). Hence, the experts need to agree about choosing which one. With argumentation, we can compute and present extensions for the experts, thus enables them to consider all possible options.

Witnesses. Given a set of conflicting arguments, we compute (also present and visualize) witnesses of the conflicts. Let A_c and $A_{\neg c}$ be the set of arguments having claim c

and $\neg c$ respectively, i.e. $A_c = \{(\Phi, \alpha) \in A \mid \alpha = c\}$ and $A_{\neg c} = \{(\Phi, \alpha) \in A \mid \alpha = \neg c\}$. A_c and $A_{\neg c}$ explain why we should approve/ disapprove c. Presenting the witnesses lets the experts understand problems arise during reconciliation.

Example 1. *In Figure 1, EoverI wants to approve c_1, c_2, c_3 and disapprove c_4. From the complete semantics, we obtain the extension $\{w^e_1, w^e_3, w^b_1, w^e_4, w^e_5, w^e_6\}$. Based on the explanations provided by this extension's arguments, EoverI can better argue for the other two participants to approve c_1 (explained by w^e_1), c_2 (explained by w^e_2), and disapprove c_4 (explained by w^e_4, w^e_5, and w^e_6). Furthermore, observing the witnesses for and against c_4 would make the decision to discard this correspondence more convincing. Indeed, from those witnesses ($A_{c_4} = \{w^b_2\}$ and $A_{\neg c_4} = \{w^e_4, w^e_5, w^e_6\}$), we realize that although the decision supporting c_4 is voted by more users, the opposite decision (to disapprove c_4) seems to be the one that is better explained. Moreover, disapproving c_4 is also justified by intuitive observations on the network: the approval of this correspondence would constitute not only a one-to-one constraint violation (with c_2) but also a cycle constraint violation (with c_1 and c_3).*

We cannot only compute the extensions and witnesses to facilitate the discussion among the experts, but also associate heuristic metrics to further support their work. Furthermore, we enable the users to rank decisions, based on the strengths of their explanations (arguments) or the decisions themselves.

Argument Strength. Computing extensions is a preliminary step to evaluate arguments. From the occurrences of an argument in the extensions, we compute the *argument strength*. Given the set of extensions $\mathcal{E}$ of an argumentation framework with respect to an acceptability semantics, the strength of an argument a is the number of its occurrences in $\mathcal{E}$ divided by the size of $\mathcal{E}$:

$$argument\_strength(a) = \frac{\sum_{\epsilon \in \mathcal{E}} \mathbb{1}_{a \in \epsilon}}{|\mathcal{E}|}$$

With *argument strength*, we have a more fine-grained metric to rank arguments and assist the users to make wiser decisions. Indeed, we were motivated by the notion of *argument acceptance* [12], which evaluates arguments based on their occurrences in all extensions of a given acceptability semantics. However, this is a rough metric, which does not take into account the difference between the number of occurrences of arguments in the extensions. This shortcoming prevents the users from having detailed looks on the credibility of arguments to compare them.

Decision Strength. Providing explanations might be overwhelming for the users, especially when there are too many arguments. Therefore, we associate each decision with a quantitative metric reflecting the *decision strength*, which is computed by applying aggregate operators (max, min, avg, etc.) on the set of supporting arguments. Based on this metric, the users can evaluate which decisions should be made given a specific circumstance. It is also important to note that the number of ambiguous correspondences is generally large. As a result, identifying the sequence of correspondences to negotiate is necessary. In practice, one may define such a sequence by taking pairs of decisions for and against a correspondence in the ascending order of the level of ambiguity, which can be measured by the difference between the strengths of associated decisions.

Example 2. *An example of argument and decision strength can be found in Figure 3. In that figure, we have on the left the decisions (circle shapes) supporting and opposing each correspondence in the network, as well as the associated arguments (square shapes). We follow the complete acceptability semantics to compute argument strengths and apply the* sum *operator to evaluate decision strengths. Those values are displayed right above the corresponding shapes. We can observe that the decision to disapprove c_4 possesses higher strength. Thus, disapproving c_4 would be the better decision to take. In fact, this outcome aligns with what the users should achieve using the qualitative metrics solely. Having the quantitative metrics (argument and decision strength), however, brings the users more details thus increases the confidence in making decisions.*

4.2 What-If Analysis

We have also developed another reasoning service, called *what-if analysis*, that exists in many decision support systems [25] and largely supports the collaborative work. This service can compute (and in our tool also visualize) the consequences of a possible decision. Using the what-if analysis, the experts can understand the consequences of any particular decision, resulting in a stronger feeling of trust throughout their work. Three questions for which we can compute the answers are:

- Q_1: *Which **arguments** will be added or deleted if a particular assertion is given?*
- Q_2: *Which **attacks** will be added or deleted if a particular assertion is given?*
- Q_3: *Which **extensions** will be modified if a particular assertion is given?*

By answering these questions, we can provide the experts with two important views. The (1) *Local view* reflects the relationships among inputs of a single participant. Each participating expert can check whether his new assertion conflicts with the previous inputs. Technically, we use the answers of Q_1 and Q_2 to construct this view. When an user gives a new assertion, his own arguments are maintained. If any attack between those arguments is found, the user is notified to adjust his inputs to avoid further inconsistencies. Besides, the (2) *Global view* reflects the connections between inputs of multiple users. All participants can observe the negotiation progress. To construct this view, we use the answers of Q_2 and Q_3. The number of attacks and extensions are maintained. In one hand, the users understand the current conflicts (attacks) among their arguments and the impact of those inconsistencies. In the other hand, keeping track of the extensions lets them know the current state of the system and when it reaches an agreement.

Example 3. *In Figure 1, three video content providers now attempt to change their assertions to reach an agreement. In the view-point of EoverI, he might change his disapproval of c_4 since the others both approve c_4. If EoverI approves c_4, two new arguments $w_7^e = \{\{c_4\}, c_4\}$, $w_8^e = \{\{c_4, \neg c_2 \vee \neg c_4\}, \neg c_2\}$ and two new attacks $w_7^e \leftrightarrow w_4^e$, $w_8^e \leftrightarrow w_2^e$ will be added. Through local view, EoverI can foresee these new arguments and attacks to realize the contradiction with himself. In the view-point of BBC and DVDizzy, they might change the approval of c_4 because of EoverI. If they disapprove c_4, two arguments w_2^d and w_2^b will be deleted; and hence, there is no attack between remaining arguments and only one extension remains. Through global view, they can foresee this consequence and feel more confident to make changes. In addition, they might also agree with EoverI on c_1 and c_2 since no further contradiction exists.*

5 Implementation

In the previous sections, we discussed how to support the collaborative reconciliation through detecting conflicts in the assertions of multiple experts and guiding the resolution of these conflicts. To accomplish these tasks, we need to realize the argumentation framework, which can only be achieved after generating arguments and computing the attack relation. This section serves a two-fold purpose. First, we present how to instantiate an argumentation framework using ASP-based tools. Second, we describe how to implement the proposed services on top of this argumentation framework.

5.1 Instantiate Argumentation Framework

We rely on a declarative language, Answer Set Programming (ASP) [7], to carry out the preliminary tasks before instantiating an argumentation framework (Figure 2). In our work, we utilize the ASP Solver DLV-Complex[8] to take advantages of its built-in data structures and functions. We invoke an ASP program called Π_{pre}, which is essentially the union of other ASP program, each of which is responsible for a specific task. In particular, $\Pi_{pre} = \Pi_{smn} \cup \Pi_{\Gamma} \cup \Pi_{inputs} \cup \Pi_{\Phi}$.

Encoding the Schema Matching Network (Π_{smn}). We extend the setting to keep track of the correspondences. Let I be the set of identities, a function $f : C \rightarrow I$ maps exactly one element in I to each in C. f is *injective* as an identity is assigned to at most one correspondence. For each schema $s_i \in \mathcal{S}$, we represent the attribute-schema relationship between s_i and each attribute $a \in A_{s_i}$ as a ground fact $attr(a, s_i)$. It is assumed that the attributes are globally unique. Meanwhile, the correspondence-attribute relationships are captured by $cor(c, a_i, a_j)$ for each $(a_i, a_j) \in C$ and $c = f((a_1, a_2))$. For instance, the network in Figure 1 has the following encoding of Π_{smn}:

$$\begin{aligned}\Pi_{smn} = \{&attr(a_1, S_1).attr(a_2, S_2).attr(a_3, S_3).attr(a_4, S_3).\\ &cor(c_1, a_1, a_2).cor(c_2, a_1, a_3).cor(c_3, a_2, a_3).cor(c_4, a_1, a_4).\}\end{aligned}$$

Encoding the Integrity Constraints (Π_{Γ}). We encode the integrity constraints as rules. In fact, we use rules to encode two different types of constraints. The first type is our basic assumptions, which are encoded in the program π_{ass} below:

$$\begin{aligned}\pi_{ass} = \{ &\leftarrow attr(a, s), attr(a, s'), s \neq s'.\\ &\leftarrow cor(c, a, a'), attr(a, s), attr(a', s'), s = s'.\}\end{aligned} \tag{1}$$

Second, we use rules to express the network-level integrity constraints. Atoms prefixed with # are built-in functions of DLV-Complex.

- *Cycle constraint*: a path of correspondences must not make any two attributes of a schema reachable. Each $rch(S, a, a')$ signifies the reachability between attributes a and a' via the correspondences in S. Below is the encoding of the program π_{cycle}:

$$\begin{aligned}\pi_{cycle} = \{&rch(S, a, a') \leftarrow cor(c, a, a'), \#set(c, S).\\ &rch(S, a, a') \leftarrow \#intersection(P, Q, R), \#card(R, 0),\\ &rch(P, a, b), rch(Q, b, a'), \#union(P, Q, S).\\ &violation(S) \leftarrow rch(S, a, b), a \neq b, attr(a, s), attr(b, s).\}\end{aligned} \tag{2}$$

- *One-to-one constraint*: any attribute is matched by at most one attribute in each of the other schemas. Each $pair(c, c\prime)$ captures a violation detected by π_{1-1} below:

$$\begin{aligned} \pi_{1-1} = \{&pair(c,c') \leftarrow cor(c,a,b), cor(c',a,b'), b \neq b', attr(b,s), attr(b',s). \\ &violation(S) \leftarrow pair(c,c'), \#set(c,c',S).\} \end{aligned} \tag{3}$$

To put it in a nutshell, Π_Γ is defined formally as $\Pi_\Gamma = \pi_{ass} \cup \pi_{cycle} \cup \pi_{1-1}$. For example, invoking Π_Γ for the network in Figure 1, we have π_{ass} check the validity of the schema matching network encoded by Π_{smn}, which is valid indeed. Besides, from π_{cycle} and π_{1-1}, we obtain two violations: $\{c_2, c_4\}$ and $\{c_1, c_3, c_4\}$.

Collecting User Inputs (Π_{inputs}). Indeed, user inputs are *assertions* on the correspondences. We represent user assertions (that may be *approvals* or *disapprovals* of correspondences) as ground atoms of the form $app(\cdot)$ and $dis(\cdot)$. For instance, in Figure 1b, the user EoverI approves c_1, c_2, c_3 and disapproves c_4, while BBC and DVDizzy only approve c_3 and c_4. Their inputs are encoded as follows:

$$\Pi_{inputs}^{EoverI} = \{app(c_1).\ app(c_2).\ app(c_3).\ dis(c_4).\}$$
$$\Pi_{inputs}^{BBC} = \Pi_{inputs}^{DVDizzy} = \{app(c_3).\ app(c_4).\}$$

Constructing the Set of Formulae (Π_Φ). We construct this set by extracting propositional formulae from either the assertions (through π_{simple}) or the detected violations (through $\pi_{extract}$). Formally, $\Pi_\Phi = \pi_{simple} \cup \pi_{extract}$. Each atom $kb(\cdot)$ captures a formula. For assertions, we consider each assertion $app(c)$ (or $(dis(c))$ as a simple formula c (or $\neg c$). This is capture by the program π_{simple}:

$$\begin{aligned} \pi_{simple} = \{&kb(c) \leftarrow app(c). \\ &kb(\mathsf{neg}(c)) \leftarrow dis(c).\} \end{aligned} \tag{4}$$

Things are more complex in the cases of the constraint violations (detected by the π_{cycle} and π_{1-1} of Π_Γ). From a violation $\{c_1, \cdots, c_n\}$, we can state that at least one of the assertions must be false. This is expressed formally by the formula $\neg c_1 \vee \cdots \vee \neg c_n$. Such formulae are extracted from the detected violations by the program $\pi_{extract}$, whose process is described below:

- Initially, violations are captured by ground atoms $violation(S)$ where S is a set of correspondences. We convert from set to list using the atom by $vioList(L)$, in which L is the list-based representation of S.
- From each list L, we create sublists starting from the first to the $(n-1)$th element $([c_1, \cdots, c_n], [c_2, \cdots, c_n], \cdots, [c_{n-1}, c_n])$.
- Based on the sublest, we form formulae in a bottom-up manner, starting from the shortest one $([c_{n-1}, c_n])$. The longer sublists have their forumlae composed recursively from those that are one element shorter. This is continued up to original list.

Example 4. *In Figure 1b, we have the collected the inputs of EoverI. Through π_{simple}, we obtain the formulae $kb(c_1)$, $kb(c_2)$, $kb(c_3)$, and $kb(\mathsf{neg}(c_4))$. Besides, we also detected the violations in the schema matching network, from which $\pi_{extract}$ form two formulae $kb(\mathsf{or}(\mathsf{neg}(c_2), \mathsf{neg}(c_4)))$ and $kb(\mathsf{or}(\mathsf{neg}(c_1), \mathsf{or}(\mathsf{neg}(c_3), \mathsf{neg}(c_4))))$. These ground atoms $kb(\cdot)$ compose the set of formulae of EoverI.*

With the set of formula, we proceed to first generate arguments then the attack relation, with the goal to compose an argumentation framework. One could consider formulae in the set we just obtained as candidates to be argument claims. This approach, however, would easily overwhelm the users due to the huge amount of generated arguments. The reason is that many formulae are syntactically different but semantically equivalent. To avoid this scenario, we limit the candidates for argument claims. In practice, users concern more with arguments claiming to approve or disapprove correspondences. We thus select the set of possible claims from the assertions. In the motivating example, the possible claims for EoverI is $cl(c_1)$, $cl(c_2)$, $cl(c_3)$, and $cl(\mathsf{neg}(c_4))$.

We take advantage of Vispartix [11], an ASP-based tool, to not only generate arguments but also to compute the attack relation. For argument generation, the tool considers only subsets of the set of formulae and the set of possible claims. It then looks for pairs which can be considered as arguments (Figure 1b presents an example of these generated arguments). Once the set of arguments is ready, we start to compute the attack relation. This is done by invoking the corresponding feature of Vispartix with the set of all arguments (the union of the arguments of each user) as the input. Visaprtix provides the users with several attack types [5], such as *defeat*, *undercut*, and *rebut*.

5.2 Realizing Services

In Section 3 and 4, we showed the elements of an argumentation framework as well as offered possible services (conflict detection, interpretation of conflict structures, what-if analysis) on top of this framework. In this subsection, we will describe how to realize these services, with the focus on technical aspects.

Conflict Detection. We detect conflicts based on the results of ASP-solver in section 5.1. In that section, we described how to encode the integrity constraints in the language of ASP. The solver DLV-Complex is responsible for detecting the violations based on our encodings. Based on the results of the solver, we have the atoms $vioList(L)$ as lists of violation, each of which contains a set of involved correspondences. Moreover, in our system, we show not only the violations but also the explanations for these violations. In doing so, we analyze the attack relations R of the argumentation framework. The user inputs are valid if this *vioList* is empty or R is empty.

Interpretation of Conflict Structures. To realize this service, we need to compute four elements: the extension, the witness, the argument strength, and the decision strength. In Section 5.1, we already generated a set of arguments. As previously defined, a witness of a claim is a set of arguments having this claim. By grouping arguments sharing the same claim, we obtain the witnesses for all possible claims. Then, we employ Vispatrix [11] to generate all possible extensions with different semantics. After obtaining all extensions, we compute argument and decision strength as mentioned in Section 4.1.

What-if Analysis. To realize this service, we need to recompute the argumentation framework and all possible extensions when user modifies an assertion. Then, we compare the differences between the new computed results and the current ones. Based on these differences, we can know what arguments, attacks, and extensions are added or deleted to answer three what-if questions in Section 4.2. This service is implemented with the support of Vispatrix [11], which allows efficient recomputation.

All above-mentioned services are integrated in our argumentation-based negotiation support tool, namely ArgSM. This tool not only implements these services but also provides graphical user interface. The details will be described in the next section.

6 Tool - ArgSM

ArgSM is an argumentation-based negotiation support tool for schema matching. It is developed by using the Java programming language and the JUNG[1] library for visualization purposes. We also integrate other supporting libraries, including Vispatrix [11] and DLV-Complex [8]. We have also made the source code to be publicly available at our website [2]. In this section, we discuss the user interface and the technical challenges.

6.1 User Interface

ArgSM can aggregate the assertions of multiple experts and visualize the concerned arguments. For *visualizing*, we provide users with a GUI (Figure 3), which is a unified view of the provided inputs and the argumentation framework. From the GUI, the users can compare their inputs via *views* and *view modes*. In particular, views give the users static pictures about the network, the decisions, and the explanations, while view modes provide the users with dynamic interaction during collaborative reconciliation.

Two views are supported in ArgSM: *Schema* and *Argumentation* view. They are displayed alongside each other in the GUI (Figure 3, from right to left). Together, they should help the users to review the inputs and make decisions effectively.

- **Schema View**. This view shows the schema matching network for the users who do not have deep understandings of argumentation, hence another name *User view*. In this view, correspondences are highlighted according on to their status. There are three possible status: (1) *all approved* and (2) *all disapproved* respectively for correspondences that are approved and disapproved by all users, and (3) *ambiguous* for those that are approved by some and disapproved by the others.
- **Argumentation View**. Also called the technical view, it is intended for those who have knowledge on argumentation. In this view, the numbers outside the shapes indicate the strengths of decisions (circles) or witnesses (squares) respectively. There are two perspectives supported:
  - *Decision-making perspective* shows all possible decisions (aggregated from the inputs) and the associated witnesses (arguments) that explain the reasons for making decisions.
  - *Abstract argumentation perspective* presents the argumentation framework in the form of a directed graph. The nodes are the arguments and the directed edges are elements of the attack relation.

Those views are further supported by three *view modes*. Apart from the *Normal mode*, which is set by default and has no interaction at all, the others allow users to interact with the network and the arguments:

- **Schema-Argumentation Mode**. Upon clicking on a correspondence in the *Schema view*, the user can see all generated decisions (circle shapes) and witnesses (square shapes) in the *Argumentation view*. For instance, in Figure 1, correspondence c_3 has

[1] JUNG - `http://jung.sourceforge.net`
[2] `https://code.google.com/p/argsm/wiki/ArgSM`

two arguments w_4^e and w_5^e for disapproving and w_2^b for approving. Therefore, the users will have two decisions at their disposal (c_4 and $\neg c_4$, presented in circles) and three witnesses (w_4^e, w_5^e, and w_2^b, presented in squares). Those circles and squares will be highlighted when the users click on c_4 in the *Schema view*.

– **Argumentation-Schema Mode**. There are two cases. First, when choosing a witness in the *Argumentation view*, the participants will see all the involved correspondences in the *Schema view*. Correspondences appearing in the support are showed differently from those in the claim of the witness. In the other case, once a decisions is clicked on, the relating correspondence is highlighted in the *Schema view*.

For a better understanding and stronger feelings of trust, ArgSM not only generates explanations but also provides the foreseeable effects of each decision. Technically, we keep the strength of arguments and the possible decisions up-to-date during negotiation.

6.2 Technical Challenges

When implementing ArgSM, we had to cope with a number of scalability issues. The schemas are usually too large, leading to high response time (i.e. computation time) for each human interaction and overwhelming control for the experts. To overcome such challenges, we apply the following techniques:

– **Partitioning**. We divide the correspondences into small disjoint and independent subsets such that any two correspondences in one subset share a common attribute.
– **Caching**. We apply the view maintenance technique [6] with a repository storing intermediate results along the process. The rationale behind is that collaborative reconciliation is incremental as a change (insertion or removal) only affects some arguments. Recomputing all arguments after each modification is unnecessary.
– **Filtering**. It is not useful to generate any and every argument. We only filter for arguments of predefined claims. Hence, not only does it reduce computation time but also avoid overwhelming the users. Every argumentation process should operate on the filtered set. That set may be refined by modifying the predefined claims.

To show the efficiency of the above techniques, we set up an experiment to measure the response time of computing arguments and attacks for a large network. With the help of automatic matchers, we obtain 472 correspondences in the network. Since the network is large, it is partitioned into 21 clusters, in which smallest and biggest ones contain

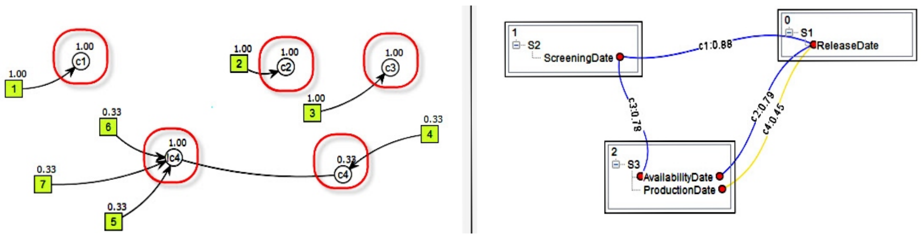


Fig. 3. The GUI of ArgSM, with *Argumentation view* (left) and *Schema view* (right)

6 and 59 correspondences respectively. Applying caching and filtering for each cluster, the response time varies from 0.38s (the smallest cluster) to 12.92s (the biggest cluster). In total, it takes about 61.16s to generate all arguments and attacks.

7 Related Work

In this section, we first review some techniques and applications related to schema matching. We then review salient work that supports negotiation in collaborative information systems as well as applications built on top of argumentation-based negotiation.

7.1 Schema Matching - Research and Applications

Database schema matching is an active research field. The developments of this area have been summarized in two surveys [4, 40]. Existing works on schema matching focused mainly on improving quality parameters of matchers, such as precision or recall of the generated matchings. Recently, however, one started to realize that the extent to what precision and recall can be improved may be limited for general-purpose matching algorithms. Instead of designing new algorithms, there has been a shift towards matching combination and tuning methods. These works include YAM [14], top-k matchings generation [23], reuse of existing matchings [22], systematic matching ensemble selection [21], automatic tuning of the matcher parameters [30], or validation of correspondences with the help of schemas belonging to the same domain [44]. While there is a large body of works on schema matching, the post-matching reconciliation proces, which is central to our work, has received little attention in the literature. Recently, there are some works [13, 27, 33, 34] using pay-as-you-go integration method to establish the initial matching and then incrementally improve matching quality. While [13, 27, 33] depend on one user only, the framework in [34] relies on the work of multiple participants of the crowdsourcing community, who -unlike in our setting- do not communicate. The authors [34] propose constraint-based aggregation techniques for conflicting user inputs. The major difference between our work and [34] is that we assume the participants are domain experts on the field.

Regarding applications, one of the major challenges posed by the Intertnet and collaborative business is enterprise interoperability (EI) in recent years is the ability of two or more systems to exchange information with each other. Several relevant initiatives have been undertaken and are ongoing today worldwide. Among them, schema matching is an important primitive to tackle strategic challenges such as Interoperability Service Utility [17], Web Technologies for Enterprise Interoperability [31], Knowledge-Oriented Collaboration [35], and Science Base for Enterprise Interoperability [10]. The roles of schema matching in enterprise systems are summarized in [47].

7.2 Argumentation in Collaborative Work

In the literature, many researchers have studied collaborative work which involves multi participants with different and possibly conflicting interests. In order to resolve conflicts and reach mutually acceptable agreements, participants negotiate with each other. There is a large body of mechanisms dedicated to support this negotiation process, including game-theoretical approach [29, 45], heuristic-based approach [19, 28], and argumentation-based approach [36, 46]. The work of this paper focuses on the argumentation-based approach, which provides not only explanations for each decision but also a language for communication between participants [42, 43].

The argumentation-based approach has been successfully applied to many practical applications. In e-commerce systems [3], argumentation is used for solving conflicts that may arise among distributed providers in large scale networks of web services and resources, thus improving the automation level of business processes. In collaborative & cooperative planning [18], argumentation can be combined with other techniques (e.g. machine learning) to help participants collaborate to solve problems by determining what policies are operating by each participant. In social-network platforms [24], arguments can be extracted from natural language and then argumentation is used to determined the social agreements among participants. In cloud-computing [26], argumentation can be used to help cloud providers, who manage computational resources in the platform, to reach an agreement on reacting against physical failures. In semantic web [41], argumentation has been modeled under Argument Interchange Format (AIF) ontology, which forms the foundation for a large-scale collection of interconnected arguments in the Web. In this paper, we apply argumentation in data integration domain, which is an active research field for more than ten years [4].

From the argumentation point of view, our work is related to two main directions in argumentation research: *abstract* and *logical* argumentation. This classification can be found in [38], along with a discussion on the development of argumentation research. In brief, abstract argumentation was proposed in [16]. In that paper, the author used an *argumentation framework* to describe a system of arguments and attacks, which are actually considered as abstract objects, hence the name *abstract argumentation*. Acceptability semantics of arguments were studied in [16], [15], and [9]. To make abstract argumentation more applicable, there are attempts to give concrete definitions to arguments and attacks. The most prominent proposal is *logical argumentation* [5], which relies on propositional logic. In particular, an argument is a pair of *support* and *claim*, while attacks are defined on the logical inconsistencies between the supports and/or claims of arguments.

8 Conclusion

We presented an argumentation-based tool to support collaborative reconciliation, where multiple users, with different sorts of opinions, cooperate to validate the outputs of automatic matchers. While splitting the reconciliation task is highly desirable, combining the individual results in the presence of consistency constraints is very challenging for the collaborating experts. Our tool and its services shall facilitate collaboration. In particular, we systematically detect conflicts, provide the experts with visual information to understand the causes of the problems. Moreover, we offer services to better understand decision consequences and make collaborative reconciliation more transparent.

Our work opens up some future research directions. First, we will design a negotiation protocol to enable negotiation within our tool. Second, we would like to extend the notion of proposed constraints and consider further integrity constraints that are relevant in the praxis (e.g., functional dependencies, domain-specific constraints). Third, we would like to apply our methods to other problems. While our work focuses on schema matching, our techniques, especially the argumentation-based reconciliation, could be applicable to other tasks such as entity resolution or business process matching.

Acknowledgment. This research has received funding from the NisB project - European Union's Seventh Framework Programme (grant agreement number 256955) and the PlanetData project - Network of Excellence (grant agreement number 257641). Many thanks to SAP [3] and MOM [4] for providing use cases and data sets as well as evaluating our tool.

References

[1] Aumueller, D., et al.: Schema and ontology matching with COMA++. In: SIGMOD, pp. 906–908 (2005)
[2] Belhajjame, K.: User feedback as a first class citizen in information integration systems. In: CIDR, pp. 175–183 (2011)
[3] Bentahar, J., et al.: Using argumentation to model and deploy agent-based B2B applications. In: KBS, pp. 677–692 (2010)
[4] Bernstein, P.A., Madhavan, J., Rahm, E.: Generic Schema Matching, Ten Years Later. PVLDB 4(11), 695–701 (2011)
[5] Besnard, P., Hunter, A.: Elements of Argumentation. The MIT Press (2008)
[6] Blakeley, J.A., Larson, P.-A., Tompa, F.W.: Efficiently updating materialized views. SIGMOD Rec., 61–71 (1986)
[7] Brewka, G., Eiter, T., Truszczyński, M.: Answer set programming at a glance. Commun. ACM, 92–103 (2011)
[8] Calimeri, F., Cozza, S., Ianni, G., Leone, N.: Computable Functions in ASP: Theory and Implementation. In: Garcia de la Banda, M., Pontelli, E. (eds.) ICLP 2008. LNCS, vol. 5366, pp. 407–424. Springer, Heidelberg (2008)
[9] Caminada, M.: Semi-Stable Semantics. In: COMMA, pp. 121–130 (2006)
[10] Charalabidis, Y., Gonalves, R.J., Popplewell, K.: Developing a Science Base for Enterprise Interoperability. In: Enterprise Interoperability IV, pp. 245–254 (2010)
[11] Charwat, G., Wallner, J.P., Woltran, S.: Utilizing ASP for Generating and Visualizing Argumentation Frameworks. In: ASPOCP, pp. 51–65 (2012)
[12] Doutre, S., Mengin, J.: On Sceptical Versus Credulous Acceptance for Abstract Argument Systems. In: Alferes, J.J., Leite, J. (eds.) JELIA 2004. LNCS (LNAI), vol. 3229, pp. 462–473. Springer, Heidelberg (2004)
[13] Duchateau, F., Bellahsene, Z., Coletta, R.: Matching and Alignment: What Is the Cost of User Post-Match Effort? In: Meersman, R., et al. (eds.) OTM 2011, Part I. LNCS, vol. 7044, pp. 421–428. Springer, Heidelberg (2011)
[14] Duchateau, F., et al.: (Not) yet another matcher. In: CIKM, pp. 1537–1540 (2009)
[15] Dung, P.M., Mancarella, P., Toni, F.: Computing ideal sceptical argumentation. Artif. Intell., 642–674 (2007)

[3] `http://www.sap.com`
[4] `http://www.momentumni.org`

[16] Dung, P.M.: On the Acceptability of Arguments and its Fundamental Role in Nonmonotonic Reasoning, Logic Programming and n-Person Games. Artif. Intell. 77(2), 321–358 (1995)

[17] Elvesaeter, B., et al.: Towards enterprise interoperability service utilities. In: ECOCW, pp. 224–229 (2008)

[18] Emele, C.D., Norman, T.J., Parsons, S.: Argumentation strategies for plan resourcing. In: AAMAS, pp. 913–920 (2011)

[19] Faratin, P., Sierra, C., Jennings, N.R.: Negotiation decision functions for autonomous agents. In: RAS, pp. 159–182 (1998)

[20] Gal, A.: Uncertain Schema Matching. Morgan & Claypool Publishers (2011)

[21] Gal, A., Sagi, T.: Tuning the ensemble selection process of schema matchers. In: JIS, pp. 845–859 (2010)

[22] Gal, A., et al.: Completeness and ambiguity of schema cover. In: Meersman, R., Panetto, H., Dillon, T., Eder, J., Bellahsene, Z., Ritter, N., De Leenheer, P., Dou, D. (eds.) OTM 2013. LNCS, vol. 8185, pp. 241–258. Springer, Heidelberg (2013)

[23] Gal, A., et al.: Making sense of top-k matchings: a unified match graph for schema matching. In: IIWeb, pp. 6:1–6:6 (2012)

[24] Grosse, K., Chesevar, C.I., Maguitman, A.G.: An Argument-based Approach to Mining Opinions from Twitter. In: AT, pp. 408–422 (2012)

[25] Haas, P., et al.: Data is Dead Without What-If Models. In: PVLDB, pp. 11–14 (2011)

[26] Heras, S., et al.: The Role of Argumentation on the Future Internet: Reaching agreements on Clouds. In: AT, pp. 393–407 (2012)

[27] Jeffery, S.R., Franklin, M.J., Halevy, A.Y.: Pay-as-you-go user feedback for dataspace systems. In: SIGMOD, pp. 847–860 (2008)

[28] Kowalczyk, R., Bui, V.: On Constraint-Based Reasoning in e-Negotiation Agents. In: Dignum, F.P.M., Cortés, U. (eds.) AMEC 2000. LNCS (LNAI), vol. 2003, pp. 31–46. Springer, Heidelberg (2001)

[29] Kraus, S., Sycara, K., Evenchik, A.: Reaching agreements through argumentation: a logical model and implementation. In: AI, pp. 1–69 (1998)

[30] Lee, Y., et al.: eTuner: tuning schema matching software using synthetic scenarios. JVLDB, 97–122 (2007)

[31] Nagarajan, M., et al.: Semantic interoperability of web services-challenges and experiences. In: ICWS, pp. 373–382 (2006)

[32] Nguyen, Q.V.H., et al.: An MAS Negotiation Support Tool for Schema Matching (Demonstration). In: AAMAS, pp. 1391–1392 (2013)

[33] Nguyen, Q.V.H., et al.: Minimizing Human Effort in Reconciling Match Networks. In: ER (2013)

[34] Nguyen, Q.V.H., et al.: On Leveraging Crowdsourcing Techniques for Schema Matching Networks. In: DASFAA, pp. 139–154 (2013)

[35] Ouksel, A.M., Sheth, A.: Semantic interoperability in global information systems. In: SIGMOD, pp. 5–12 (1999)

[36] Parsons, S., Sierra, C., Jennings, N.: Agents that Reason and Negotiate by Arguing. JLC, 261–292 (1998)

[37] Peukert, E., Eberius, J., Rahm, E.: AMC - A framework for modelling and comparing matching systems as matching processes. In: ICDE, pp. 1304–1307 (2011)

[38] Prakken, H.: Some Reflections on Two Current Trends in Formal Argumentation. In: Artikis, A., Craven, R., Kesim Çiçekli, N., Sadighi, B., Stathis, K. (eds.) Sergot Festschrift 2012. LNCS, vol. 7360, pp. 249–272. Springer, Heidelberg (2012)

[39] Qi, Y., Candan, K.S., Sapino, M.L.: FICSR: feedback-based inconsistency resolution and query processing on misaligned data sources. In: SIGMOD, pp. 151–162 (2007)

[40] Rahm, E., Bernstein, P.A.: A Survey of Approaches to Automatic Schema Matching. JVLDB, 334–350 (2001)

[41] Rahwan, I., Zablith, F., Reed, C.: Towards large scale argumentation support on the semantic web. In: AAAI, pp. 1446–1451 (2007)
[42] Rahwan, I., et al.: Argumentation-based negotiation. In: KER, pp. 343–375 (2003)
[43] Sá, S., Alcântara, J.: Cooperative dialogues with conditional arguments. In: AAMAS, pp. 501–508 (2012)
[44] Saleem, K., Bellahsene, Z.: Complex Schema Match Discovery and Validation through Collaboration. In: Meersman, R., Dillon, T., Herrero, P. (eds.) OTM 2009, Part I. LNCS, vol. 5870, pp. 406–413. Springer, Heidelberg (2009)
[45] Sandholm, T.: Algorithm for optimal winner determination in combinatorial auctions. In: AI, pp. 1–54 (2002)
[46] Sierra, C., Jennings, N.R., Noriega, P., Parsons, S.: A Framework for Argumentation-Based Negotiation. In: Singh, M.P., Rao, A., Wooldridge, M.J. (eds.) ATAL 1997. LNCS, vol. 1365, pp. 177–192. Springer, Heidelberg (1998)
[47] Smith, K.P., et al.: The Role of Schema Matching in Large Enterprises. In: CIDR (2009)

Completeness and Ambiguity of Schema Cover

Avigdor Gal[1], Michael Katz[1], Tomer Sagi[1], Matthias Weidlich[1], Karl Aberer[2], Hung Quoc Viet Nguyen[2], Zoltán Miklós[3], Eliezer Levy[4], and Victor Shafran[4]

[1] Technion – Israel Institute of Technology, Technion City, Haifa, 32000 Israel
[2] École Polytechnique Fédérale de Lausanne EPFL, CH-1015 Lausanne, Switzerland
[3] University of Rennes 1, France
[4] SAP Research Israel, Ra'anana, 43665, Israel

Abstract. Given a schema and a set of concepts, representative of entities in the domain of discourse, schema cover defines correspondences between concepts and parts of the schema. Schema cover aims at *interpreting* the schema in terms of concepts and thus, vastly simplifying the task of schema integration. In this work we investigate two properties of schema cover, namely *completeness* and *ambiguity*. The former measures the part of a schema that can be covered by a set of concepts and the latter examines the amount of overlap between concepts in a cover. To study the tradeoffs between completeness and ambiguity we define a cover model to which previous frameworks are special cases. We analyze the theoretical complexity of variations of the cover problem, some aim at maximizing completeness while others aim at minimizing ambiguity. We show that variants of the schema cover problem are hard problems in general and formulate an exhaustive search solution using integer linear programming. We then provide a thorough empirical analysis, using both real-world and simulated data sets, showing empirically that the integer linear programming solution scales well for large schemata. We also show that some instantiations of the general schema cover problem are more effective than others.

1 Introduction

Semantic interoperability among heterogeneous information systems is a critical problem for modern business networks. Data integration is considered one of the main challenges in establishing such interoperability, due to the need to provide correct interpretation of data [1,2,3]. In today's world of connected businesses, the idea of approaching the data integration challenge with methods based on reuse and collaboration becomes feasible. Such reuse can be based on a repository of information building blocks, referred to as *concepts*, [4] representative of entities in the domain of discourse (*e.g.*, a vendor concept in an eCommerce domain). Concepts establish a form of a conceptual middleware, providing a shared set of abstractions that facilitates interoperability.

Documents, modeled as schemata are mapped against a set of concepts in a process termed *schema cover*. The idea is to "cover" a schema and thereby interpret the schema in terms of known concepts. This way, the schema is integrated into an existing body of information and knowledge. For example, consider a network of enterprises that exchange business documents and cooperate to establish interoperability in order to conduct business and generate value from the network. The business documents do not

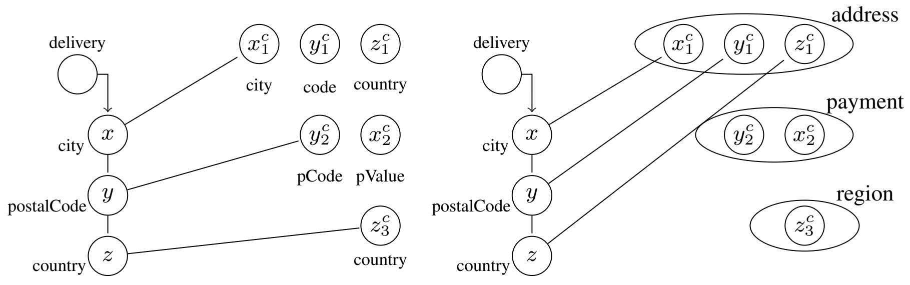


Fig. 1. Role of concepts illustrated

follow a common format or standard, although they come from the same domain, *e.g.*, sales and purchase orders. The aim of schema cover is to integrate different vocabulary and structural elements, representing similar real-world entities, by using concepts.

Schema cover matches parts of schemata (called subschemata) with concepts, using schema matching techniques [5] and then adds cover-level constraints. One such constraint is called *ambiguity* [4], intuitively representing the number of times an attribute is matched to attributes in several concepts. Another constraint, introduced in this work, is *completeness*, representing the part of a schema that is "covered" by any concept. Ambiguity and completeness constraints, to be formally defined in this work along with subschemata and concepts, are embedded as (either hard or soft) constraints in an optimization cover problem.

To illustrate the role of concepts in schema cover consider Figure 1. On the left there is a schema with three attributes. On the right, we introduce six attributes that are available for matching in some repository (not necessarily from the same schema). In the absence of concepts (Figure 1(left)), the matching task can select attributes without any restriction. However, in the presence of concepts (Figure 1(right), marked with ovals), we are guided to choose closely related attributes. Our hypothesis is that well-designed concepts improve the quality of the integration for exactly this argument.

Schema cover was first introduced by Saha et. al. [4] as a solution for schema mapping. However, in recent years new applications were introduced, to which schema cover can bring benefit [6,7]. We illustrate next two applications that motivate our investigation of the trade-off between ambiguity and completeness in schema cover.

Partner Identification: An enterprise may be interested in investigating a new market, seeking partners with sufficient common interests to its own while providing sufficient added value to its capabilities. The use of a cover here can ensure just the right amount of commonality between prospective business partners. Here, the main interest is in the amount of completeness of a cover, judging what part of one schema can be covered by concepts of the other schema and vice versa. Ambiguity takes a secondary role of ensuring the cover clarity.

Concept Filtering: Effective reuse requires the accumulation of many concepts from which useful schema covers can be created. To ensure efficient cover processing, concept repositories require a filtering process that ensures that only concepts of good quality that are also good candidates for a cover are retrieved. In this scenario, completeness is set as a constraint by a designer and ambiguity is a measure that needs to be optimized to avoid poor matching quality.

In this paper we investigate the trade-offs between completeness and ambiguity. The paper makes the following contributions:

- We introduce the trade-off of ambiguity and completeness through a general framework for schema cover, offering a spectrum of schema cover problems where the above applications can be supported, among other scenarios. In particular, we show that the framework of [4] is a special case of the generalized problem.
- We propose a new formulation of a cover problem and show it to be NP-Complete.
- We offer integer linear programming (ILP) formulation for optimally solving certain special variations of the cover problem in an exhaustive manner. These formulations are shown empirically to scale well to schemata with 1,000 attributes and cover repository with 8,000 concepts.
- We provide a thorough empirical analysis of the proposed algorithmic solutions to cover problems, using both real-world and simulated data sets. The empirical evaluation shows that the cover variation proposed in this work is more effective (in terms of completeness for similar levels of ambiguity) than the one presented by Saha et al. [4].

The rest of the paper is organized as follows. Preliminaries are given in Section 2 and Section 3 introduces the cover model. A set of cover problems are presented in Section 4, together with a theoretical analysis, followed by an ILP formulation in Section 5. Section 6 provides our empirical analysis. We conclude with a description of related work (Section 7) and directions for further research (Section 8).

2 Background

In this work, we focus on the task of covering a schema with a (possibly large) repository of concepts and analyze the trade-offs of completeness and ambiguity. Generally speaking, a preprocessing of the cover process involves decomposition and coupling. The former decomposes a schema into subschemata. Concepts and subschemata are then coupled using schema matching techniques, followed by a cover selection.

Schema decomposition is a process in which a schema is broken into subschemata. We set no particular restrictions on this process, and therefore attributes can belong to more than one schema, a subschema (modeled often as a graph) does not have to be a connected component, *etc.* Decomposition can be performed in one of two ways, namely *native* and by *concept-first*. Native decomposition is performed independently of the concept repository. For example, a method for decomposing an underlying ER model was proposed by An et al. [8]. Alternatively, decomposition by concept-first is guided by a set of concepts using schema matching techniques. Therefore, a schema matcher such as COMA++ [9] or OntoBuilder's Top-K algorithm [10] can determine the best matchings between a concept and the schema. All the schema attributes that participate in such a matching are considered part of a subschema.

To make the subsequent pairing and covering process more efficient, the concept repository can be filtered by selecting representatives of concept clusters. Filtering was advocated in [4] as a heuristic to confront the high complexity of the cover problem. In this work, we show first that the use of Integer Linear Programming with state-of-the-art solvers can do without filtering even for very large concept repositories. Secondly, we discuss a variant of the generalized cover description that assist in focusing the cover solutions on relevant concepts. Therefore, we can filter-in clusters for which a representative concept is identified as a good candidate for covering.

Coupling is a schema matching process in which the similarity between a concept and a subschema is assessed. If decomposition is performed by concept-first, then coupling trivially becomes the outcome of the matching discussed above. Otherwise, the same schema matching techniques are applied to construct a set of concept-subschema pairs for the cover process.

In the remainder of this section we provide, as a background, a model of schema matching based on the model presented by Gal [10], and put it in the context of the cover problem. We shall use a car rental domain example to demonstrate our model, where a set of concepts may include details about cars, customers, drivers, payments, pickup, return, and stations. The example attempts to match a schema of a car rental portal to these concepts.

2.1 Schema Matching

Let *schema* $s = \{a_1, a_2, ..., a_n\}$ be a finite set of *attributes*. Attributes can be both simple and compound and compound attributes should not necessarily be disjoint. An attribute in a schema of a car rental portal might be carType, firstName, *etc*. A compound attribute might be pickUpDate, combining two other attributes – eDay, and eMonth. In what follows, given a schema s with n attributes, we shall use the attribute indices $(1, 2, ..., n)$ to identify the attributes of s.

Let s and s' be schemata with n and n' attributes, respectively. Let $\mathcal{S} = s \times s'$ be the set of all possible *attribute correspondences* between s and s'. $\mathcal{S}$ is a set of attribute pairs (*e.g.*, (puDate, PickUpDate)). Let $m(s, s')$ be an $n \times n'$ *similarity matrix* over $\mathcal{S}$, where $m_{i,j}(s, s')$ represents a degree of similarity between the i-th attribute of s and the j-th attribute of s'. Whenever the schemata are known from the context we use $m_{i,j}$ instead of $m_{i,j}(s, s')$. The majority of works in the schema matching literature define $m_{i,j}$ to be a real number in $[0, 1]$. For example, Table 1 represents a simplified similarity matrix of the running case study. Schema s has the attributes group (representing car group), seat (referring to child's safety seat), xDriver (representing an extra driver), puDate (for pickup date) and rDate (for return date). For schema s' the attributes carType, pickUpDate and returnDate are self explanatory. The attribute options is a compound attribute with two sub-attributes chkBaby and chkExtraDriver, each a binary attribute. It is worth noting that the matcher, in this case, has identified chkBaby as a perfect match to seat and xDriver as a perfect match to chkExtraDriver, propagating these scores to the matching of seat and xDriver with options.

Similarity matrices are generated by *schema matchers*, instantiations of the schema matching process. They differ mainly in the measures of similarity they employ, which yield different similarity matrices. These measures can be arbitrarily complex, and may use various techniques. Some matchers assume similar attributes are more likely to have similar names [11,12]. Other matchers assume similar attributes share similar

Table 1. A Similarity Matrix Example

$s \longrightarrow$ / $\downarrow s'$	1 group	2 seat	3 xDriver	4 puDate	5 rDate
1 carType	0.843	0.323	0.323	0.317	0.302
2 options	0.290	1.000	1.000	0.326	0.303
3 pickUpDate	0.344	0.328	0.328	0.351	0.352
4 returnDate	0.312	0.310	0.310	0.359	0.356

domains [13,14]. Others yet take instance similarity as an indication of attribute similarity [15,16]. Finally, some researchers use the experience of previous matchings as indicators of attribute similarity [17,12].

Given two schemata, s and s', let the power-set $\Sigma_{s,s'} = 2^{\mathcal{S}}$ be the set of all possible *schema matches* between the schema pair (s, s'), where a schema match $\sigma(s, s') \in \Sigma_{s,s'}$ is a set of attribute correspondences (and thus $\sigma(s, s') \subseteq \mathcal{S}$). Whenever the schema pair is known from the context, we refer to a match simply as σ and to a power-set of matches as Σ. It is worth noting that σ does not necessarily contain all attributes in s or s'. Therefore, there may exist an attribute $a \in s$, such that for all $a' \in s'$, $(a, a') \notin \sigma$. We denote by $\bar{\sigma} = \{a \in s \,|\forall a' \in s', (a, a') \notin \sigma\} \cup \{a' \in s' \,|\forall a \in s, (a, a') \notin \sigma\}$ the set of all attributes that do not participate in a schema match.

2.2 Similarity and Constraints

A schema match is assigned a similarity measure that is aggregated from the similarity measures of its attribute correspondences. In the literature, such an aggregation took various forms, including among others, the aggregate functions of scaled summation (such as $average$), max, and min.

For the remainder of this work, f represents any of the common linear aggregate operators for schema matching. Given a non-empty schema match σ between two schemata s and s', we can define a schema match similarity measure $M_\sigma(s, s')$ to be:

$$M_\sigma(s, s') = f\left(\{m_{i,j}(s, s') \,|\, (a_i, a_j) \in \sigma\}\right). \tag{1}$$

For example, consider Table 1 and assume that the schema match σ involves matching

$$\{(1,1), (2,2), (3,3), (4,3), (5,4)\}.$$

Also, let $f = average$, then $M_\sigma(s, s') = 0.71$.

Let $\Gamma : \Sigma \to \{0, 1\}$ be a boolean constraint function that captures the application-specific constraints on schema matchings, *e.g.*, cardinality constraints and inter-attribute correspondence constraints. Γ partitions Σ into two sets, where the set of all *valid* schema matches in Σ is given by $\Sigma^\Gamma = \{\sigma \in \Sigma \mid \Gamma(\sigma) = 1\}$. Γ is a general constraint model, where $\Gamma(\sigma) = 1$ means that the match σ can be accepted by a designer. Γ has been modeled in the literature using matchers called *constraint enforcers* [18].

The input to the process of schema matching is given by two schemata s and s' and Γ. The output of the schema matching process is a *schema match* $\sigma \in \Sigma^\Gamma$.

3 Cover Model

We now introduce the model that serves in defining the different schema cover problems. The model includes subschemata, concepts (Section 3.1), and a cover (Section 3.2).

3.1 Subschemata and Concepts

Let $T_s = \{t_1, t_2, ..., t_m\}$ be a set of *subschemata* of s, $t_i \subseteq s$ for all $i = 1, 2, ..., m$. A subschema contains a subset of the attributes of s. For example, a subschema of a car rental portal may include the attributes carType, options and insurance, the first two already discussed above and the third referring to the type of insurance needed for this car group. In what follows, we keep the original indexes of s to represent the attributes from s that are present in t. For example, $t = \{a_1, a_5, a_8\}$ is a subschema of s that contains three of the attributes of s, namely a_1, a_5, and a_8.

Let $C = \{c_1, c_2, ..., c_p\}$ be a set of concepts, where a concept is a schema by itself. For example, in the car rental domain, a concept repository contains the following concepts: CarDetails, CustomerDetails, DriverDetails, PaymentDetails, PickUpDetails, ReturnDetails, and Station.

We note that schemata, subschemata, and concepts are all defined to consist of a set of attributes, however with different semantics. Concepts are schemata whose meaning is assumed to be known and well-defined in a given business context. Such concepts can be prepared by an enterprise for internal standardization purposes. Alternatively, it can be generated and maintained by organizations such as schema.org. A schema represents a new, unknown set of attributes, that is a candidate for a matching task. For clarity sake, we differentiate schema attributes from concept attributes and denote by a_j^c the j-th attribute of concept c.

3.2 Cover

We are now ready to introduce the notion of a cover, defined in this paper as any set of concepts that match parts of a given schema. Therefore, if a concept interprets part of a schema, a cover interprets a schema as a whole. We provide a formal definition of a cover, demonstrate it using our case study example, and explain the roles of completeness and ambiguity as quality measures for a cover.

Given a set of subschemata T_s of s, a set C of concepts, and a constraint function Γ for each $t \in T_s, c \in C$, we define a set of valid matchings between subschemata and concepts: $\mathcal{E}(T_s, C) = \left\{\sigma(t, c) \middle| t \in T_s, c \in C, \sigma(t, c) \in \Sigma_{t,c}^{\Gamma}\right\}$.

Definition 1. *A* cover *of s by C, $v_{s,C} \subseteq \mathcal{E}(T_s, C)$ is a subset of valid matchings between T_s and C.*

A cover is a set of pairs, where each pair in the set is a matching between a subschema and a concept. Let $\sigma = \sigma(t, c) \in v_{s,C}$ be an element of a cover. We define the presence vector $\bar{\lambda}_\sigma = (\lambda_{\sigma,1}, \lambda_{\sigma,2}, ...\lambda_{\sigma,n})$ as follows:

$$\lambda_{\sigma,i} = \begin{cases} 0 & a_i \in \bar{\sigma} \\ 1 & \text{otherwise} \end{cases} \tag{2}$$

Table 2. A Cover Example

Concept	Matching subschema
CarDetails: group, seat, insuranceTypeRequested...	carType, options, insurance...
CustomerDetails: address, addressCity, addressCountry...	address,city,country...
DriverDetails: ageGroup, email, license...	
PaymentDetails: CCV, card, cardExpiryMonth...	
PickUpDetails: date, time, stationID...	pickUpDate, pickUpTime, location...
ReturnDetails: date, time, stationID...	returnDate, returnTime, location...
Station: ID, name, location...	

Each element of the vector $\bar{\lambda}_\sigma$ is an indicator, set to 1 if attribute a_i is part of the matching σ and 0 otherwise (recall that $\bar{\sigma}$ represents all the attributes that are **not** in σ).

To illustrate, a partial sample cover is given in Table 2. Four concepts are used to cover a schema of a car rental portal where pickup and return location are always the same.[1] Among the schema attributes in the table, the attribute location is present twice in the cover, matched by the concepts PickUpDetails and ReturnDetalis, and therefore the relevant entry of the attribute location in $\bar{\lambda}_\sigma$ for each of the two concepts is assigned with the value of 1.

Given a cover $v = v_{s,C}$ between a schema s and a set of concepts C, we define the presence vector of v, $\bar{\lambda}_v = (\lambda_{v,1}, \lambda_{v,2}, ...\lambda_{v,n})$, to be the vector summation of its matchings presence vectors. Therefore,

$$\bar{\lambda}_v = \sum_{\sigma \in v} \bar{\lambda}_\sigma. \tag{3}$$

Using the values in a presence vector, two quality measures can be derived from it. First, *ambiguity* was introduced in [4] as a phenomenon where several concepts may give a different semantic interpretation to an attribute in a schema. We define the ambiguity of a cover to be the sum of duplicate appearances of an attribute in a cover:

$$A_v(s) = \sum_i \max(0, \lambda_{v,i} - 1) \tag{4}$$

As another quality measure we offer *completeness*, checking to what extent schema attributes are present in a cover:

$$C_v(s) = \frac{\sum_i \min(1, \lambda_{v,i})}{|s|} \tag{5}$$

$\lambda_{v,i} = 0$ means that attribute a_i is not matched by any of the concepts that participate in v, hence reducing completeness (C_v). $\lambda_{v,i} = 1$ means that attribute a_i is matched by exactly one pair and $\lambda_{v,i} > 1$ means that attribute a_i is present in more than one pair in the cover, hence increasing ambiguity (A_v). For example, the cover in Table 2 is represented by a presence vector where the relevant entry of the attribute carType is assigned the value of 1 and the relevant entry of the attribute location is assigned with 2, reflecting the attribute ambiguity in the cover.

[1] This has been the case for many years when bidding for car rentals in `priceline.com`

To generalize schema matching similarity to a cover we observe that Eq. 1 can be adopted to reflect the similarity of a subschema-concept pair. Let $M_\sigma(t,c)$ denote the similarity measure of a matching σ between a subschema t and a concept c. Then,

$$M_v(s,C) = f(\{M_\sigma(t,c) \mid \sigma \in v\}) . \tag{6}$$

For example, for $f = sum$, the schema matching similarity of a cover v is

$$M_v(s,C) = \sum_{\sigma \in v} M_\sigma(t,c) . \tag{7}$$

In what follows, we shall also use a measure of dissimilarity. Therefore, given a cover v, $dM_v(s,C)$ is defined similarly

$$dM_v(s,C) = f(\{dM_\sigma(t,c) \mid \sigma \in v\}) \tag{8}$$

where $dM_\sigma(t,c)$ is defined to be $1 - M_\sigma(t,c)$.

4 Problem Definitions

Equipped with the formal definition of a cover, we can devise an array of optimization problems, all aiming at optimizing some quality aspect of a cover. To illustrate the various optimization problems, we now provide a simplified example.

Example 1. Let $s = \{a_1, a_2, a_3\}$ be a schema and $T_s = \{\{a_1\}, \{a_1, a_2\}, \{a_2, a_3\}\}$ be a set of subschemata of s. Also, let $C = \{c_1, c_2, c_3\}$ be a set of concepts so that $\mathcal{E}(T_s, C) = \{(\{a_1\}, c_1), (\{a_1, a_2\}, c_2), (\{a_2, a_3\}, c_3)\}$ is the set of valid matchings between T_s and C. The attribute correspondences are illustrated in Figure 2. The similarity values of the elements of $\mathcal{E}$ are given as follows:

σ	$M_\sigma(t,c)$	σ	$M_\sigma(t,c)$	σ	$M_\sigma(t,c)$
$(\{a_1\}, c_1)$	0.45	$(\{a_1, a_2\}, c_2)$	0.95	$(\{a_2, a_3\}, c_3)$	0.45

□

4.1 Singly-Bounded Maximization Cover

The following problem was specified in [4] as the schema covering problem.

Problem 1 (Singly-Bounded Maximization Cover). Given a set of valid matchings $\mathcal{E}(T_s, C)$, the *singly-bounded maximization cover* problem (SBMC) is defined to be:

$$\max_{v \subseteq \mathcal{E}(T_s,C)} M_v \ s.t.\ \bar{\lambda}_v \leq \bar{H},$$

where $\bar{H}$ is a vector of integers.

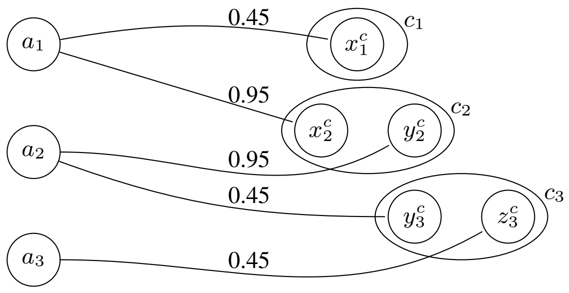


Fig. 2. Illustration of a Cover

For example, let $\bar{H} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$. Such a vector ensures no ambiguity ($A_v(s) = 0$) but does not guarantee completeness ($C_v(s) \leq 1$). Two possible covers that satisfy the presence constraint are $\{(\{a_1\}, c_1), (\{a_2, a_3\}, c_3)\}$ and $\{(\{a_1, a_2\}, c_2)\}$. Using *sum* for M_v, as in [4], we get $M_v = 0.9$ for the first cover and $M_v = 0.95$ for the second cover. Therefore, the cover $\{(\{a_1, a_2\}, c_2)\}$ is the solution to Problem 1.

Problem 1 carries two characteristics we would like to tune. First, note that a solution to the optimization problem guarantees ambiguity to be within a certain limit but lacks control over completeness. As a result, there may be attributes in the schema that are not covered by any concept. This is evident in the example above, where covering a_1 and a_2 is preferred over covering all attributes. Our second observation is that Problem 1 is "greedy" in the sense that concepts may be added although they offer no true contribution to the cover, simply because the presence "budget" was not fully spent yet. This feature negatively affects ambiguity uneccessarily, without any added value to completeness. To demonstrate this phenomenon, we change $\bar{H}$ to be $\bar{H} = \begin{bmatrix} 2 \\ 2 \\ 1 \end{bmatrix}$. The cover $\{(\{a_1\}, c_1), (\{a_1, a_2\}, c_2), (\{a_2, a_3\}, c_3)\}$ becomes the solution to Problem 1 with $M_v = 1.85$, although both $\{(\{a_1\}, c_1), (\{a_2, a_3\}, c_3)\}$ (with $M_v = 0.9$) and $\{(\{a_1, a_2\}, c_2), (\{a_2, a_3\}, c_3)\}$ (with $M_v = 1.4$) suffice to achieve maximal completeness.

4.2 Doubly-Bounded Minimization Cover

We are now ready to introduce an alternative cover problem, the *doubly-bounded minimization* cover problem. This problem seeks covers with different properties than Problem 1. As our empirical analysis shows later, the doubly-bounded minimization cover problem offers a more attractive alternative in terms of matching effectiveness.

Problem 2 (Doubly-Bounded Minimization Cover). Given a set of valid matchings $\mathcal{E}(T_s, C)$, the doubly-bounded minimization cover problem (DBMC) is defined to be:

$$\min_{v \subseteq \mathcal{E}(T_s, C)} dM_v \ s.t. \ \bar{H}_l \leq \bar{\lambda}_v \leq \bar{H}_u,$$

where $\bar{H}_l$ and $\bar{H}_u$ are vectors of integers.

In Problem 2 we set a minimal bound on presence. If $\bar{H}_l = \bar{0}$ then we are back to the presence constraint that was set in Problem 1. Setting $\bar{H}_l = \bar{1}$ ensures full completeness. Higher values of elements of $\bar{H}_l$ generalize the problem. We also note that Problem 2 aims at minimizing dissimilarity. Therefore, any additional concept added to the cover may either leave its dissimilarity measure unchanged or increase it. This change mitigates the "greediness" of Problem 1. Concepts are added to maintain the lower presence bound and the overall presence is also bounded from above as before.

dM_v is a representative of a quality measure assigned with a specific cover and $\bar{H}_l$ and $\bar{H}_u$ are constraints associated with each attribute in the original schema. As such, Problem 2 (as well as Problem 1) are instantiations of a general formulation for a cover problem in which a quality measure of a cover is optimized subject to a set of constraints on individual attributes.

It is worth noting that problems 1 and 2 both treat $\bar{\lambda}$ as a hard constraint while optimizing a measure of similarity (or dissimilarity). Optimizing two quality measures of a cover, presence and similarity, may be alternatively viewed as a bi-objective optimization problem. Therefore, one can also envision an ambiguity minimization problem, where the role of ambiguity and similarity is exchanged. We refrain from providing a formal presentation of such a problem due to space consideration and defer it to an extended version of this work. Intuitively, such a cover problem specification allows the designer to specify the importance she assigns with various attributes, putting more emphasis on the similarity of attributes that are of greater importance. Therefore, such specification becomes handy when filtering the concept base.

The need for introducing the generalized cover problem and several of its instantiations goes beyond intellectual curiosity. If there are various ways to use concepts to cover schemata, can we evaluate whether one way is better than the other? One way of doing so can evaluate which of the instantiations yields a more effective matching. Indeed, in Section 6 we answer this question empirically, showing that solutions to the new formulation offered in this work (Problem 2) outperform solutions to Problem 1 (as introduced by Saha et al.), reaching higher completeness for the same ambiguity level.

4.3 Complexity Analysis

Having introduced two variations of the generalized cover problem we now turn our attention to analyzing the complexity of the problem. it was shown in [4] that the *Singly-Bounded Maximization Cover* problem is NP-complete. Theorem 4 (which proof is omitted due to space considerations) asserts the same complexity result for the *Doubly-Bounded Minimization Cover* problem.

Theorem 1. *The decision of the DBMC problem is NP-complete.*

5 ILP Formulation for Cover Problems

We present now an Integer Linear Programming formulation to the DBMC problem. ILP problems are known to be NP-complete [19], and therefore no polynomial time algorithm exists (unless P=NP). However, contemporary efficient solvers solve many instances of ILP within a reasonable time frame. In Section 6 we present an extensive empirical evaluation using MOSEK solver [20], showing its ability to solve the DBMC problem efficiently, even for large concept bases. We have also implemented the ILP formulation for SBMC, the cover problem presented in [4], showing the efficiency (albeit with lower effectiveness then DBMC) of the solution.

We start by noting that in the DBMC problem, the optimization is performed over subsets of the set $\mathcal{E}(T_s, C)$, and thus we associate a binary variable X_σ with each $\sigma \in \mathcal{E}(T_s, C)$. There is a natural $1 : 1$ correspondence between the assignments to these variables and the subsets of $\mathcal{E}(T_s, C)$. Given that, the linear constraints are defined according to Eq. 3 as follows.

$$\bar{H}_l \leq \sum_{\sigma \in \mathcal{E}(T_s, C)} X_\sigma \cdot \bar{\lambda}_\sigma \leq \bar{H}_u. \tag{9}$$

Using Eq. 8, dM_v is defined as a linear aggregation function over the elements of v, *i.e.*

$$dM_v = f(\{dM_\sigma \mid \sigma \in v\}), \tag{10}$$

where each dM_σ depends only on σ. Then, the (linear) objective function of our ILP can be defined as in Eq. 10. For example, the summation aggregation function yields

$$\min \sum_{\sigma \in \mathcal{E}(T_s, C)} dM_\sigma \cdot X_\sigma. \tag{11}$$

6 Empirical Evaluation

In this section we describe and discuss our empirical evaluation of the cover problem. We first detail the datasets and the experiment setup (Section 6.1) followed by a description and analysis of experiment results.

6.1 Datasets and Experiment Setup

Datasets. We used two types of datasets, namely real-world and synthetic. Features of both datasets are summarized in Table 3.

The OntoBuilder dataset consists of 27 Web forms from two domains, car reservations and aviation. We extracted a schema from each Web form using OntoBuilder.[2] The schemata vary in size, from 21 to 88 attributes. The Vendor and Business Partner datasets consists of 3 schemata each, containing information about the vendor and business partner business objects, derived from three different SAP systems. The IBM dataset describes the features of the dataset that was used in [4]. We were unable to experiment with this dataset but we have experimented with datasets (both real-world and synthetic) with similar features. It is worth noting that we have reimplemented the algorithm of [4] using our general ILP formulation and experimented with it, comparing it to our proposed algorithm.

[2] All ontologies and exact matchings are available for download from the OntoBuilder Web site, `http://ie.technion.ac.il/OntoBuilder`.

Table 3. Data and Concept Sets

Dataset	# of schemata	# of domains	schema size (# of attributes)	Relevant concept sets
OntoBuilder	27	2	21-88	WF
Vendor	3	1	304	UBL, NativeVendor
Business Partners	3	1	~100	UBL, NativeBP
IBM	5	1	144-456	IBM
Synthetic			700-1000	Synthetic
Concept set	**# of concepts**	**# of domains**	**concept size (# of attributes)**	**Relevant datasets**
WF	15	2	3-12	OntoBuilder
NativeVendor	10-12	1	10	Vendor, Business Partner
UBL	62	1	500-700	Vendor, Business Partner
IBM	292	4	~15	IBM
Synthetic	350-8000		25-40	Synthetic
NativeBP	10	1	3-10	Vendor, Business Partner

The Web form (WF) concept set is constructed by human effort for each domain. The NativeVendor concept set is constructed by performing schema decomposition to each of the three schemata in the Vendor dataset. The UBL concept set was formed from schemata of the Universal Business Language (UBL) version 2.1. UBL is a library of standard electronic XML business documents such as purchase orders and invoices that is developed by OASIS.[3] The IBM concept set represents the features of the concept set that was used in [4].

The synthetic dataset is used for run-time evaluation. Therefore, we generate additional schemata and concepts using multiple copies of the real-world schemata with minor variations of schema element names. The number of copies depend on the specific scale that is required per experiment.

Experimental Setup. We implemented an experimental environment to test the different solutions. In each experiment a single schema (from the pool of real-world or synthetic datasets) is introduced to a concept repository. We vary the size and type of concepts in the concept repository. The decomposition phase is performed using concepts from a relevant domain and the number of subschemata also vary among experiments. Jointly, the number of concepts and the number of subschemata determine a range for the number of pairs in each experiment.

We have varied the low and high presence constraints. For the high presence constraint we use a k-reduction value that ranges from 0 to 4. A k-reduction value of 0 sets the high presence constraint for each attribute to be the number of concepts that are matched with this attribute. Higher k-reduction values put an increasing constraint on the allowed presence. Table 4 provides the six control parameters, for each stating the range of values we use in our experiments and a baseline value (whenever there is one), kept fixed unless otherwise described.
We report on the following metrics:

– Ambiguity: For a cover v we measure $A_v(s)$ normalized by schema size, for better comparison among schemata.

[3] `http://www.oasis-open.org/committees/ubl/`

Table 4. Controlled Parameters

Parameter	Parameter Name	Range	Baseline
$\|C\|$	# of concepts	4-3000	
$\|T\|$	# of subschemata	2-500	
$\|a\|$	# of attributes in a subschema	2-100	
$\|a^c\|$	# of attributes in a concept	2-100	
H_l	low ambiguity constraint	0-2	1
k	reduction of high ambiguity constraint	0-4	0

– Completeness: For a cover v we measure $C_v(s)$.
– Runtime: the execution time of each algorithm.

The experiments were conducted on a Intel(R) Core(TM)2 Quad CPU Q8200 @ 2.33GHz. The algorithms were implemented in Java, using JDK version 1.6.0. The JVM was initiated with a heap-memory size of $8.00GB$.

For decomposition, we use Auto-Mapping Core (AMC), a tool developed by SAP Research that provides an infrastructure and a set of algorithms to establish correspondences between two business schemata. The algorithms are designed to explore various features and dependencies that exist in business schemata. Based on the extracted features the algorithm may suggest correspondences between nodes in two different schemata. Different algorithms may suggest different correspondences and the overall result is integrated based on quality measures as reported by the various algorithms.

6.2 Ambiguity-Completeness Tradeoff

In the first set of experiments we solve the DBMC problem starting with $\bar{H}_l = \bar{1}$ and then relaxing completeness by assigning a 0 value to some elements of the $\bar{H}_l$ vector.

Figure 3 provides the tradeoff between the two parameters for one schema pair. Other pairs show similar behavior. As expected, there is a tradeoff between ambiguity and completeness, where an increased completeness is necessarily accompanied with an increased ambiguity. Specifying both $\bar{H}_l$ and $\bar{H}_u$ allows a designer the flexibility to choose where on the curve to seek a solution.

In the next set of experiments we have taken real-world schemata of varying sizes (ranging from 30 to 40 attributes), and tested the impact of various presence constraints. As a baseline, we have applied a $k = 0$ reduction of presence, allowing maximum ambiguity per attribute and setting $\bar{H}_u$ accordingly. Even at this level some attributes may not find a correspondence, due to a threshold constraint that is applied by AMC, in which case their lower bound is set to 0. Then, we started reducing the values of $\bar{H}_u$ by increasing k and checking for completeness.

Table 5 illustrates the impact of the presence constraint on SBMC and DBMC and the tradeoff between ambiguity and completeness. Each table's row represents the average of completeness for schemata of a different size. Each column represents the k-reduction in ambiguity for SBMC and DBMC. Each entry is the average completeness of schemata of a given size with a different k-reduction value and one of SBMC and DBMC.

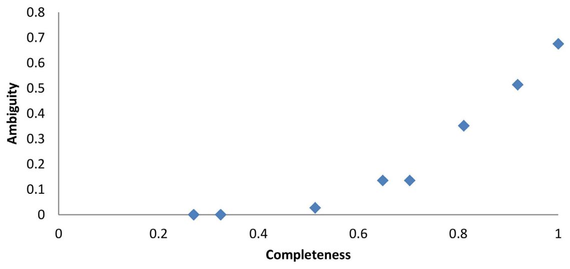


Fig. 3. Ambiguity vs. Completeness

Table 5. Impact of Presence Constraint

k-reduce $\rightarrow$	0		1		2		3		4	
$\downarrow$ Schema size	SBMC	DBMC	SBMC	DBMC	SBMC	DBMC	SBMC	DBMC	SBMC	DBMC
30	0.63	0.63	0.6		0.57		0.57		0.57	
31	0.58	0.58	0.48		0.48		0.48		0.48	
32	0.72	0.72	0.72	0.72	0.69		0.69		0.63	
33	0.61	0.61	0.55		0.52		0.52		0.52	
34	0.56	0.56	0.56	0.56	0.56	0.56	0.56	0.56	0.56	0.56
35	0.43	0.43	0.43	0.43	0.43	0.43	0.43	0.43	0.43	0.43
36	0.36	0.36	0.25	0.36	0.28	0.36	0.28	0.36	0.28	0.36
37	0.76	0.76	0.68		0.49		0.49		0.46	
38	0.53	0.53	0.37		0.37		0.32		0.29	
39	0.51	0.51	0.41		0.41		0.41		0.36	
40	0.7	0.7	0.68	0.7	0.58	0.7	0.58	0.7	0.53	

For DBMC, completeness remains fixed, since $\bar{H}_l$ does not change with increased k. Therefore, whenever the upper limit of presence does not allow certain attributes to be mapped, DBMC returns no solution, marked with a blank cell in the table.

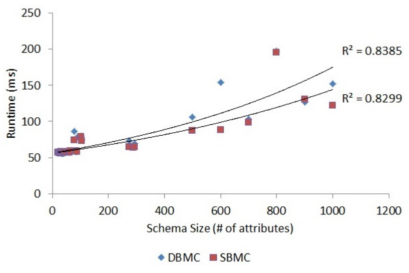
We observe that whenever a full-cover solution (baseline completeness) is possible, DBMC by definition covers all possible attributes while SBMC does not necessarily do so. For example, for a schema of size 40, completeness reduces with an increased k in solutions for SBMC, while up to $k = 3$, a solution that cover all possible attributes is still found with DBMC.

An interesting anomaly can be seen for a schema of size 36. Here, when k increases from 1 to 2, meaning that a tighter presence constraint is applied, the completeness for SBMC actually increases (from 9 to 10). This may indicate some instability in the performance of SBMC.

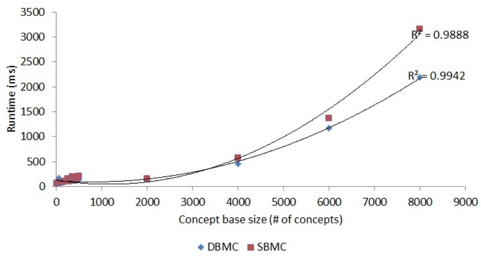
6.3 Runtime

Figure 4a demonstrates the increase of runtime with schema size, using both real-world and synthetic datasets. As expected, the runtime of the ILP-based solutions demonstrate an exponential runtime trend,[4] yet even for large schemata (of size up to 1,000 attributes) cover is performed in less than 200ms.

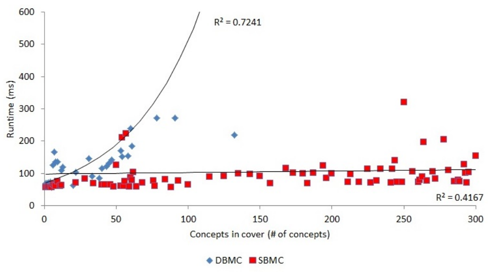
[4] Trendlines were generated using MS-Excel.



(a) A function of schema size



(b) A function of concept base size



(c) A function of number of concepts in a cover

Fig. 4. Runtime analysis

Figures 4b illustrates the impact of concept base size on runtime. Runtime increases with concept base size, albeit with a quadratic runtime trend, handling concept bases of 8,000 concepts in a few seconds.

Figure 4c illustrates the impact of the number of concepts in a cover on the execution time. The correlations is less conclusive, with low R^2 values. DBMC exhibits an exponential correlation. SBMC finds covers with more concepts than DBMC due to its greedy approach, while maintaining a lower execution time.

6.4 Discussion

Our empirical analysis covers tradeoff and run-time analysis of cover solutions. When it comes to run-time analysis we show that while the general cover problem and its two instantiations are NP-Complete, encoding cover problems using ILP generates an efficient solution that can handle large schemata and big concept bases. Using presence constraints with DBMC, when given in moderation, help in identifying better interpretation with higher completeness.

7 Related Work

A definition of a schema cover was first introduced by Saha et al. [4]. In this work we extend the cover definition to a general linear constraint optimization, and offer a new algorithm to solve cover problems. We show that the ILP formulation of the problem is efficient even for large schemas and large concept repositories. We also show that the proposed cover algorithm outperforms the one proposed in [4], in its ability to provide higher completeness for a given level of ambiguity.

Our work on schema covering builds on schema matching techniques. We provide next a brief overview of major achievements in schema matching modeling, although it is not the focus of this work. The body of research on the topic of schema matching is vast and we do not attempt to cover all of it here. Schema matching research has been going on for more than 25 years now (see surveys [1,21,22,23], books [24,10], and on-line lists, *e.g.*, OntologyMatching[5] and Ziegler[6]) first as part of schema integration and then as a standalone research. Due to its cognitive complexity, schema matching has been traditionally considered to be AI-complete, performed by human experts [25,26]. Semi-automatic schema matching has been justified in the literature using arguments of scalability (especially for matching large schemata [27], where schema covering can be specifically useful) and by the need to speed-up the matching process. Fully-automatic (that is, unsupervised) schema matching was suggested in settings where a human expert is absent from the decision process, *e.g.*, machine-understandable Web resources [28].

Over the years, a significant body of work was devoted to the identification of *schema matchers*, heuristics for schema matching. Examples include COMA [9], Cupid [13], OntoBuilder [14], Autoplex [15], Similarity Flooding [29], Clio [30], Glue [31], to name just a few. The main objective of schema matchers is to provide schema matchings that will be effective from the user point of view, yet computationally efficient (or at least not disastrously expensive). Such research has evolved in different research communities, including databases, information retrieval, information sciences, data semantics and the semantic Web, and others.

8 Conclusions

We have presented a general framework for schema cover. In this formulation, a cover problem aims at optimizing a quality measure subject to a set of constraints on individual attributes. We focus on minimizing a dissimilarity measure subject to bounds on ambiguity and also show that this general framework extends previous works in which similarity was maximized subject to an upper bound on ambiguity.

The empirical analysis shows the effectiveness of the proposed cover formulation (DBMC) over previous proposal (SBMC) in terms of completeness for a given ambiguity. It also shows the efficiency of ILP encoding of cover problems, by running experiments on schemata with up to 1,000 attributes and a concept repository with up to 8,000 concepts.

As part of an ongoing work, due to the complexity of the ILP solution we also created a heuristic framework that follows a simple scheme: we first order the set $\mathcal{E}\ (T_s, C)$ using a simple scoring function and then process this ordered list, one by one. At each step, we maintain a set of candidate subschema-concept pairs: if at least one attribute of the actual pair brings us closer to satisfying the lower bound and the pair also satisfies

[5] `http://www.ontologymatching.org/`

[6] `http://www.ifi.unizh.ch/~pziegler/IntegrationProjects.html`

the upper bound, we add this element to the set of candidates. We also maintain temporary presence values for the remaining problem that we update each time we add a new pair to the candidate set. We continue with this process until either we find a valid cover or we processed the entire list. We intend to report on this framework and on our empirical evaluation with it in an extended version of this work.

We believe that cover formulation is an essential component in the toolkit of data integration. It is implemented as part of the NisB platform,[7] serving as a discovery as well as a matching tool. In terms of future work, we intend to investigate formulations of cover problems that aim at minimizing ambiguity subject to individual constrains on dissimilarity. Improving the quality of concept bases is also part of the future work agenda, possibly through techniques of clustering and natural evolution.

Acknowledgement. This research has received funding from the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreement number 256955. This research is partially supported by JAPAN TECHNION SOCIETY RESEARCH FUND.

References

1. Batini, C., Lenzerini, M., Navathe, S.: A comparative analysis of methodologies for database schema integration. ACM Computing Surveys 18(4), 323–364 (1986)
2. Lenzerini, M.: Data integration: A theoretical perspective. In: Proc. 21st ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems, pp. 233–246 (2002)
3. Bernstein, P., Melnik, S.: Meta data management. In: Proc. 20th Int. Conf. on Data Engineering, tutorial Presentation (2004)
4. Saha, B., Stanoi, I., Clarkson, K.: Schema covering: a step towards enabling reuse in information integration. In: Proc. 26th Int. Conf. on Data Engineering, pp. 285–296 (2010)
5. Melnik, S.: Generic Model Management: Concepts and Algorithms. Springer (2004)
6. Lee, M., Yang, L., Hsu, W., Yang, X.: XCLUST: Clustering XML schemas for effective integration. In: Proceedings of the International Conference on Information and Knowledge Management (CIKM), pp. 292–299. ACM Press, McLean (2002)
7. Smith, K., Morse, M., Mork, P., Li, M., Rosenthal, A., Allen, D., Seligman, L.: The role of schema matching in large enterprises. In: CIDR 2009, Fourth Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA (January 2009)
8. An, Y., Borgida, A., Miller, R., Mylopoulos, J.: A semantic approach to discovering schema mapping expressions. In: Proceedings of the IEEE CS International Conference on Data Engineering, pp. 206–215 (2007)
9. Do, H., Rahm, E.: COMA - a system for flexible combination of schema matching approaches. In: Proc. 28th Int. Conf. on Very Large Data Bases, pp. 610–621 (2002)
10. Gal, A.: Uncertain Schema Matching. Synthesis Lectures on Data Management. Morgan & Claypool Publishers (2011)
11. He, B., Chang, K.C.-C.: Statistical schema matching across Web query interfaces. In: Proc. ACM SIGMOD Int. Conf. on Management of Data, pp. 217–228. ACM Press, San Diego (2003)
12. Su, W., Wang, J., Lochovsky, F.H.: Holistic schema matching for Web query interfaces. In: Ioannidis, Y., et al. (eds.) EDBT 2006. LNCS, vol. 3896, pp. 77–94. Springer, Heidelberg (2006)

[7] `http://www.nisb-project.eu/`

13. Madhavan, J., Bernstein, P., Rahm, E.: Generic schema matching with Cupid. In: Proc. 27th Int. Conf. on Very Large Data Bases, Rome, Italy, pp. 49–58 (September 2001)
14. Gal, A., Modica, G., Jamil, H., Eyal, A.: Automatic ontology matching using application semantics. AI Magazine 26(1), 21–32 (2005)
15. Berlin, J., Motro, A.: Autoplex: Automated discovery of content for virtual databases. In: Batini, C., Giunchiglia, F., Giorgini, P., Mecella, M. (eds.) CoopIS 2001. LNCS, vol. 2172, pp. 108–122. Springer, Heidelberg (2001)
16. Doan, A., Domingos, P., Halevy, A.: Reconciling schemas of disparate data sources: A machine-learning approach. In: Aref, W.G. (ed.) Proc. ACM SIGMOD Int. Conf. on Management of Data, pp. 509–520. ACM Press, Santa Barbara (2001)
17. Madhavan, J., Bernstein, P., Doan, A., Halevy, A.: Corpus-based schema matching. In: Proc. 21st Int. Conf. on Data Engineering, pp. 57–68. IEEE Computer Society, Los Alamitos (2005)
18. Lee, Y., Sayyadian, M., Doan, A., Rosenthal, A.: eTuner: tuning schema matching software using synthetic scenarios. VLDB J. 16(1), 97–122 (2007)
19. Karp, R.: Reducibility among combinatorial problems. In: Miller, R., Thatcher, J. (eds.) Complexity of Computer Computations, pp. 85–103. Plenum Press (1972)
20. MOSEK, The MOSEK Optimization Tools Version 6.0 (revision 61) (2009), `http://www.mosek.com`
21. Sheth, A., Larson, J.: Federated database systems for managing distributed, heterogeneous, and autonomous databases. ACM Comput. Surv. 22(3), 183–236 (1990)
22. Rahm, E., Bernstein, P.: A survey of approaches to automatic schema matching. VLDB J. 10(4), 334–350 (2001)
23. Shvaiko, P., Euzenat, J.: A survey of schema-based matching approaches. In: Spaccapietra, S. (ed.) Journal on Data Semantics IV. LNCS, vol. 3730, pp. 146–171. Springer, Heidelberg (2005)
24. Bellahsene, Z., Bonifati, A., Rahm, E. (eds.): Schema Matching and Mapping. Springer (2011)
25. Convent, B.: Unsolvable problems related to the view integration approach. In: Atzeni, P., Ausiello, G. (eds.) ICDT 1986. LNCS, vol. 243, pp. 141–156. Springer, Heidelberg (1986)
26. Hull, R.: Managing semantic heterogeneity in databases: A theoretical perspective. In: Proceedings of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS), pp. 51–61. ACM Press (1997)
27. He, B., Chang, K.-C.: Making holistic schema matching robust: an ensemble approach. In: Proc. 11th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, pp. 429–438 (2005)
28. Srivastava, B., Koehler, J.: Web service composition - Current solutions and open problems. In: Workshop on Planning for Web Services (ICAPS 2003), Trento, Italy (2003)
29. Melnik, S., Rahm, E., Bernstein, P.: Rondo: A programming platform for generic model management. In: Proc. ACM SIGMOD Int. Conf. on Management of Data, pp. 193–204. ACM Press, San Diego (2003)
30. Miller, R., Hernàndez, M., Haas, L., Yan, L.-L., Ho, C., Fagin, R., Popa, L.: The Clio project: Managing heterogeneity. SIGMOD Record 30(1), 78–83 (2001)
31. Doan, A., Madhavan, J., Domingos, P., Halevy, A.: Learning to map between ontologies on the semantic web. In: Proc. 11th Int. World Wide Web Conf., pp. 662–673. ACM Press (2002)

Adaptive Consistency and Awareness Support for Distributed Software Development*
(Short Paper)

André Pessoa Negrão, Miguel Mateus, Paulo Ferreira, and Luís Veiga

INESC–ID/Técnico Lisboa
Rua Alves Redol 9, Lisboa, Portugal
{andre.pessoa,mcm}@ist.utl.pt,
{paulo.ferreira,luis.veiga}@inesc-id.pt

Abstract. We present ARCADE, a consistency and awareness model for Distributed Software Development. In ARCADE, updates to elements of the software project considered important to a programmer are sent to him promptly. As the importance of an element decreases, the frequency with which the programmer is notified about it also decreases. This way, the system provides a selective, continuous and focused level of awareness. As a result, the bandwidth required to propagate events is reduced and intrusion caused by unimportant notifications is minimized. In this paper we present the design of ARCADE, as well as an evaluation of its effectiveness.

Keywords: Distributed Software Development, Replicated Data Management, Continuous Consistency, Interest Awareness.

1 Introduction

The dominant approach to work synchronization in Distributed Software Development (DSD) is to use a Version Control System, such as CVS [11]. With this approach, programmers work in an isolated environment for most of the time, sporadically synchronizing their work with their colleagues. Despite its widespread adoption, this approach presents two important drawbacks. First, the concurrent work carried out in isolation by different developers may result in synchronization–time conflicts that take time and effort to resolve. Second, it is widely regarded that maintaining programmers *aware* of what others do greatly improves the development process [3].

The acknowledgement of the importance of awareness led to the design of solutions [9,6,2,4] in which the modifications performed by each programmer are propagated to the others in real–time. Despite the improved awareness of this

* This work was partially supported by national funds through FCT – Fundação para a Ciência e Tecnologia, under projects PTDC/EIA-EIA/113613/2009, PTDC/EIA-EIA/108963/2008, PTDC/EIA-EIA/113993/2009 and PEst-OE/EEI/LA0021/2013.

solution, blindly propagating all modifications to every participant also has its shortcomings. In large projects, a high number of developers constantly modify the source code. As a result, the rate of notifications presented to the user (most of which are not relevant to his current task) is exceedingly high, leading to a distracting work environment. In addition, the bandwidth required to instantly propagate every update may prevent programmers from working in network constrained devices (such as tablets and laptops), which are becoming pervasive in working environments.

We address these issues with a continuous consistency and awareness model called ARCADE (Adaptive Replication, Consistency and Awareness for Distributed Development Environments). ARCADE dynamically controls the frequency with which programmers are notified of remote modifications to the software project. To do so, it takes into account the impact that an update has on the current task of each developer. Then, based on impact, it assigns priorities to each update, such that i) updates with high priority are propagated frequently and ii) updates with lower priorities are postponed for a period of time that depends on the impact factor. As a result, developers are more frequently notified about updates that affect their work more. In addition, network resources are used more efficiently, as savings are achieved by merging and compacting the postponed updates.

This paper is structured as follows. In Section 2 we relate ARCADE to previous work. In Section 3 we describe the model and architecture of ARCADE. In Section 4 we discuss our implementation of ARCADE as a plugin to the Eclipse IDE. In Section 5 we present the evaluation of ARCADE. Finally, Section 6 draws some conclusions.

2 Related Work

The idea of improving awareness by providing notifications to programmers in real–time has already been explored [9,2,6,4]. These systems allow for early conflict detection, potentially saving a significant amount of time and effort. However, none of these systems provides a gradual decrease in awareness as the importance of changes diminishes. As a result, they end up feeding the programmer with information that is frequently irrelevant to his current task and, consequently, unnecessarily consume extra bandwidth.

Our solution is also closely related to the notion of *Divergence Bounding* [8]. The Divergence Bounding model allows replicas to diverge, but define the conditions under which replicas are forced to synchronize. Simple solutions include forcing replicas to synchronize periodically [1] or after a maximum number discarded updates [5]. A more sophisticated approach is provided by TACT [12], which limits divergence according to a multidimensional criteria. However, it does not support varying consistency levels based on interest and locality in the data space. VFC [10] and CoopSLA [7] unify the multicriteria approach of TACT with locality awareness techniques. Both systems provide a variable degree of consistency based on the distance between a user's observation point and the location of the remote updates. In VFC distance corresponds to the metric distance over a coordinate space in a multiplayer game; in CoopSLA it corresponds to the distance within the tree structure of a text document in a cooperative editor. While both approaches are suitable for their domain, they fail to capture the highly complex dependencies of a software project, as they only consider the distance between the objects of the system, ignoring their semantic relation.

3 ARCADE

In this section we describe ARCADE's model and architecture. Our design is applicable to various object-oriented programming languages. However, throughout the rest of the paper we assume a Java–based DSD project.

3.1 Location and Impact

An object oriented project is composed of numerous entities (e.g., classes and methods) with different types of relations (e.g., inheritance) that can be expressed through a *dependency graph*. At any moment during the development process, we can map the activity of a programmer to a specific line of code (e.g., the one in which the cursor lies). This line, in turn, is a part of a code block that corresponds to one entity of the dependency graph. Hence, we can map the current focus of a programmer to a node of the graph. In the context of our work, we call such a node a *location*. We consider two types of locations. *View location* corresponds to the user's *observation point*, i.e., the location in which he is more interested. *Input location* refers to the location in which he is making changes. Typically, the view and input locations of a single programmer coincide. Most importantly, however, our approach is concerned with the relation between a programmer's view location and the input locations of the other programmers.

The dependency graph of an application shows the nature and strength of the relation between any two locations. From the graph, for any update concerning input location B, we can infer its importance regarding a programmer with current view location A. We refer to this relative importance between locations A and B as *impact*. Based on impact, we can control the frequency with which programmers receive the updates to the different locations of the project.

To exemplify, consider that programmers P_1 and P_2 are editing classes C and D, respectively. If D is a sub-class of C, it is likely that P_2 is interested in receiving the updates performed by P_1. If, on the other hand, the two classes are not related, it is likely that the work performed by one programmer does not affect the other. However, they should still be loosely informed about each other's work, in order to maintain a global level of awareness.

3.2 Controlling Notification Frequency

To clearly define the frequency with which programmers are notified, we introduce the notion of a *priority scale*. A priority scale consists of a set of monotonically decreasing priority levels to which locations are assigned according to their impact factor. Each level of the priority scale defines under which conditions an update to a location assigned to that level is allowed to be postponed. High impact locations are assigned the highest priority level and, consequently, the programmer is more frequently notified about events referring to it. As the impact of the locations decreases, they are assigned to lower priority levels; as a result, updates are postponed for longer intervals and, if possible, merged or discarded.

In each priority level, the conditions for postponing updates to a location are specified by three parameters: *time* (θ) defines the maximum amount of time a programmer can stay without being informed of changes to a location; *sequence* (σ) limits the number of updates that can be postponed or discarded without notifying the programmer; and *value* (ν) limits the divergence between the local copy of a location and its most recent state, according to some application-dependent metric. When any of these constraints is violated, every postponed update to the corresponding location is sent to the developers concerned.

In the same project, there may be a single priority scale shared by every user or multiple priority scales for different user groups. Even with a single scale, it is important to note that the enforcement of such scale is performed independently for each programmer. As such, even if two programmers have a given location in the same priority level of their particular scales, the exact timing with which they are notified of modifications to that location may differ. The reason is that the notification timing of a location depends on i) when the location is assigned to each programmer's priority level and ii) the elapsed time since updates concerning the location were propagated to each programmer. For these same two reasons, the notification timing of two different locations placed in the same priority level of a single programmer may also differ.

3.3 Architecture

In ARCADE, a single server provides the consistency and awareness management service to a group of clients running at the machines of the programmers. Both the clients and the server maintain a full replica of the project. The server replica contains, at any moment, the authoritative (i.e., the most recent) version of every element of the project. On the other hand, the client replicas have an outdated view of the project, managed according to the priority scale of each programmer. Despite the logically centralized architecture, the deployment of the system can be fully distributed (e.g., by having one client acting as the server).

It is the server's job to hold information regarding the view locations of all users, conduct update propagation and consistency enforcement, and maintain a continuously up-to-date representation of the dependencies among project elements. As operations arrive, the server measures its impact over the work of each developer of the current session. An incoming update is immediately applied to the local replica of the server and then stored until the consistency constraints force it to be sent to a particular client.

The two main components of the ARCADE server are the *Dependency Manager* (DM) and the *Consistency Manager* (CM). The DM is responsible for

Input Location	Same method	Used method/ Same method in superclass		Same class		Superclass		Used class	Same package	Other
Modification Focus		Signature	Body	Declaration/ Field	Method	Declaration/ Field	Method	Declaration/ Field	Any element	Any element
Impact Factor	1	1	3	1	2	1	3	1	4	5

Fig. 1. Impact factors

Priority Level		1 (Critical)	2 (High priority)	3 (Medium priority)	4 (Low priority)	5 (Minimum priority)
Notification Requirements	θ	1 second	10 seconds	30 seconds	60 seconds	120 seconds
	σ	1 update	5 updates	20 updates	50 updates	100 updates
	v	-	10%	20%	30%	50%

Fig. 2. Priority scale

building and maintaining the dependency graph of the software project. The CM is in charge of, for each client, translating the dependency relations between the client's view location(s) and the project's elements into levels of impact, as well as ensuring the corresponding awareness and consistency requirements.

Clients are responsible for dispatching the operations performed locally by the programmer to the server, as soon as they occur. In addition, the client translates, through a *Notification Manager* component, the updates received from the server into notification messages that the final user is able to understand (see Sect. 4).

4 Implementation

We implemented our approach as a plugin to the Eclipse IDE called *ARClipse.* ARClipse was written in Java for the Galileo version of Eclipse. In this section we discuss the most relevant aspects of the implementation of the plugin.

The impact factors and the priority scale considered in ARClipse are shown in Figures 1 and 2, respectively. We considered five impact levels, with level 1 having the highest impact and level 5 the lowest. Our impact function is based on our own notion of relevance in a Java project. However, ARCADE is flexible enough to allow different implementations. Similarly, any other distribution of impact factors across the priority levels, as well as a different priority scale, is also possible.

It is worth noting that impact also depends on the type of update. In most cases, only changes to a method signature are actually relevant; modifications to its body are, typically, not relevant for programmers using or even extending it. However, those programmers may still benefit from receiving (less frequent) notifications about the latter, which ARClipse allows.

To ensure that the updates received do not leave the local project in a non-compilable state, ARClipse maintains a dual view of the project: the programmer is notified about and can see the remote updates in the IDE, but they are not considered when the project is compiled, unless the programmer explicitly accepts them. We provide two modes of presenting remote updates. In *real–time* mode, remote updates are immediately visible in the form of comments: to each line of code containing a remote modification, ARClipse adds the prefix ”//[R]”. In *user–time* mode, remote modifications are shown in a separate window.

ARClipse adds two types of visual notifications to Eclipse, pop–up dialogs and new file icons. Pop–ups are shown to actively notify the user of remote updates. To minimize intrusion, ARClipse provides two types of dialogs. When a high impact update is received, a *warning dialog* containing the identification of the updated location is shown in the IDE. In addition, the icon of the corresponding file is tagged with a red circle. A *summary dialog* is presented periodically to the programmer, showing a list with the updates received since the last summary. The icons of files with non-critical modifications are tagged with a yellow circle.

5 Evaluation

We conducted a series of experiments to measure: i) the network performance (in terms of messages exchanged and data transferred) and ii) the CPU and memory utilization of ARCADE. The experiments were conducted on two Dual Core machines connected through a Gigabit Ethernet LAN. One machine executed the server, while the other executed a variable number of bots simulating the behaviour of programmers.

Each simulation consisted in a 30 minutes session, in which we recorded the results obtained by ARCADE and inferred the results that would be obtained by a solution that propagates every update in real-time. This solution is representative of DSD systems that provide non–variable continuous awareness. The simulations were repeated, at least, five times; further executions were conducted, when necessary, to remove outliers. The results presented correspond to the averages of the several executions.

5.1 Network Usage

Figures 3 and 4 present the results obtained regarding the number of transferred messages and the overall network traffic, respectively. The figures clearly show that ARCADE is able to significantly reduce the usage of network resources when compared to the baseline system. On average, the savings obtained surpass 50%.

In addition, the results show that savings do not decrease as the number of users increases. These results are due to the fact that different bots edit, typically, in different modules of the project. This behaviour tries to mimic the real patterns of DSD, in which the programmers added to the development team are typically assigned new tasks that are mostly independent from the other programmers' work (e.g., develop a new module). As a result, the work developed by the new programmer does not introduce a significantly higher impact to other programmers.

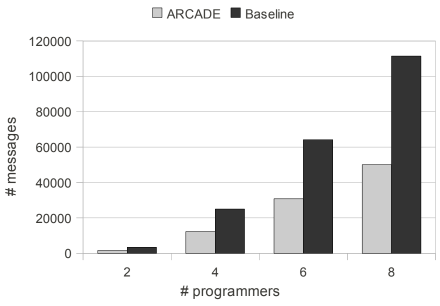


Fig. 3. Total propagated messages

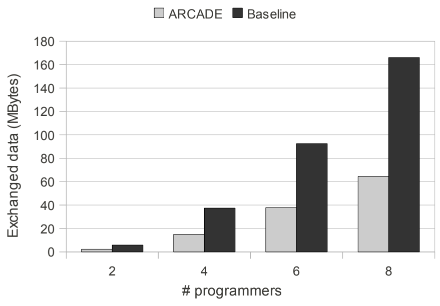


Fig. 4. Total data transferred

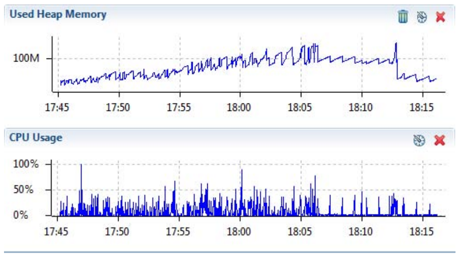


Fig. 5. Usage of client resources over time

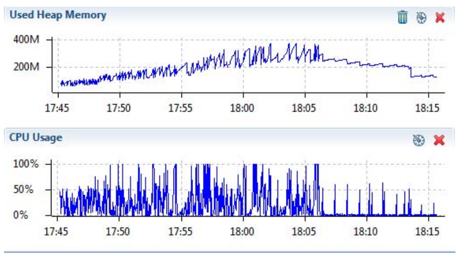


Fig. 6. Usage of server resources over time

5.2 System Resources

Figures 5 and 6 show the results measured at the client and the server, respectively, regarding CPU and memory utilization. Memory utilization at the client stays below 100 MB for most of the duration of the simulations. This is a low value that most likely will not have a significant overhead on the performance of the system. By comparison, the Chromium browser uses over 120MB to display a single Flash web page. At the server side, memory utilization is higher, as expected, due to the fact that the server retains a large number of updates that can only be safely removed when they are propagated or discarded. However, even at the highest load of 8 programmers, memory usage never surpasses 400MB.

Regarding CPU, the load at the clients averages 30% and very rarely exceeds 50%. Similar values are displayed by Chromium when presenting the page mentioned above. At the server, CPU load averages over 50%, with several spikes achieving close to 100%. These spikes are a direct consequence of the cyclic nature of ARCADE, in which a high number of previously retained updates needs to be periodically processed and sent to the programmers. These values are, nevertheless, acceptable for a dedicated server.

6 Conclusion

In this paper we described ARCADE, a new synchronization model for distributed software development. ARCADE assesses the impact that a remote update has on the task undertaken by a developer according to his current focus of work. Based on the measured impact, ARCADE determines if the update should be immediately sent to the developer or postponed for an interval that depends on the impact factor. As a result, ARCADE selectively increases the level of awareness provided to each developer by informing him more quickly about relevant changes. Furthermore, ARCADE is able to compress the log of postponed updates, thus reducing the overall network traffic. Our evaluation results show that ARCADE has great potential in terms of network savings, without requiring a significant increase in CPU and memory utilization.

References

1. Alonso, R., Barbara, D., Garcia-Molina, H.: Data caching issues in an information retrieval system. ACM Trans. Database Syst. 15(3), 359–384 (1990)
2. Cheng, L.T., de Souza, C.R., Hupfer, S., Patterson, J., Ross, S.: Building collaboration into ides. Queue 1(9), 40–50 (2003)
3. Dourish, P., Bellotti, V.: Awareness and coordination in shared workspaces. In: Proceedings of the 1992 ACM Conference on Computer-supported Cooperative Work, CSCW 1992, pp. 107–114. ACM, New York (1992)
4. Fitzpatrick, G., Marshall, P., Phillips, A.: Cvs integration with notification and chat: lightweight software team collaboration. In: Proceedings of the 2006 Conference on Computer Supported Cooperative Work, CSCW 2006, pp. 49–58. ACM, New York (2006)
5. Krishnakumar, N., Bernstein, A.J.: Bounded ignorance: a technique for increasing concurrency in a replicated system. ACM Trans. Database Syst. 19(4), 586–625 (1994)
6. Molli, P., Skaf-Molli, H., Bouthier, C.: State treemap: An awareness widget for multi-synchronous groupware. In: Proceedings of the 7th International Workshop on Groupware, pp. 106–114. IEEE Computer Society, Washington, DC (2001)
7. Negrão, A.P., Costa, J., Ferreira, P., Veiga, L.: Semantic and locality aware consistency for mobile cooperative editing. In: Meersman, R., et al. (eds.) OTM 2012, Part I. LNCS, vol. 7565, pp. 380–397. Springer, Heidelberg (2012)
8. Saito, Y., Shapiro, M.: Optimistic replication. ACM Comp. Surv. 37(1), 42–81 (2005)
9. Sarma, A., Noroozi, Z., van der Hoek, A.: Palantir: Raising awareness among configuration management workspaces. In: International Conference on Software Engineering, p. 444 (2003)
10. Veiga, L., Negrão, A.P., Santos, N., Ferreira, P.: Unifying divergence bounding and locality awareness in replicated systems with vector-field consistency. Journal of Internet Services and Applications 1(2), 95–115 (2010)
11. Vesperman, J.: Essential CVS. O'Reilly Media, Inc. (2006)
12. Yu, H., Vahdat, A.: Design and evaluation of a conit-based continuous consistency model for replicated services. ACM Trans. Comp. Syst. 20(3), 239–282 (2002)

Semantic Enrichment of Models to Assist Knowledge Management in a PLM Environment

Yongxin Liao[1,2], Mario Lezoche[1,2], Eduardo Loures[3], Hervé Panetto[1,2], and Nacer Boudjlida[4,5]

[1] Université de Lorraine, CRAN, UMR 7039, Boulevard des Aiguillettes B.P.70239, 54506Vandoeuvre-lès-Nancy, France
[2] CNRS, CRAN, UMR 7039, France
[3] Industrial and Systems Engineering, Pontifical Catholic University of Parana., ImaculadaConceicao 1155, Curitiba, Brazil
[4] Université de Lorraine, LORIA, UMR 7503, Boulevard des Aiguillettes B.P. 70239, 54506Vandoeuvre-lés-Nancy, France
[5] CNRS, INRIA, LORIA, UMR 7503, France
{yongxin.liao,mario.lezoche,herve.panetto}@univ-lorraine.fr, eduardo.loures@pucpr.br, nacer.boujlida@loria.fr

Abstract. Product Lifecycle Management (PLM) has been considered as an essential concept for improving the product competitive ability in manufacturing enterprises. The PLM solution aims at providing a shared platform for facilitating the management of the knowledge related to any product development process in or across enterprises. However, facing with different standards, enterprise systems and stakeholders, enterprises still need to deal with interoperability issues between those collaborative information systems, encompassing their capability to find the right information during the whole Product Life Cycle (PLC). The objective of this paper is to cope with the major issue of semantic interoperability, by proposing a formalization of semantic annotations and a prototype for facilitating a coherent, complete and contextualized interoperability of knowledge between all enterprise systems and related stakeholders. An example of the instantiation of our method within a real application scenario in manufacturing domain is presented to demonstrate its applicability and use, both at the engineering and the exploitation phases.

Keywords: Semantic Interoperability, Semantic Annotation, BPMN, PLM, Knowledge Management.

1 Introduction

Nowadays, the need for interoperability in a Product Lifecycle Management (PLM) environment has become increasingly imperative. A PLM solution aims to bring together different enterprise systems that deal with the product-related knowledge at each stage and intend to facilitate the communications among all stakeholders [1]. However most of the enterprises have implemented only a few of these systems without coping with any coherent integration of the entire information system. This has resulted in a kind of “tower of Babel”, where each application is considered as an island in the middle of the ocean of information. Of course, all of these stakeholders have their own background, unique knowledge, particular needs and specific practices. This “cloud of knowledge” is then over increasing the issue of interoperability, not only in the collaborations among the enterprise systems but also by the mutual understanding of the product-related knowledge between these stakeholders.

The objective of this paper is to cope with the issue of interoperability, especially the semantic interoperability, by proposing a formalization of semantic annotations and a prototype for facilitating a coherent, complete and contextualized interoperability of knowledge between all enterprise systems and related stakeholders. The remainder of this paper is organized as follows. Section 2 presents an overview of the background and identifies the semantic interoperability issues to limit the research scope and discusses the related works that made use of semantic annotations to meet particular needs. Section 3 illustrates eight formal definitions of semantic annotations, a conflict detection policy and a semantic annotation framework. A case study is presented in section 4 to demonstrate the applicability and the use of our solution. Section 5 concludes this paper and highlights future research directions.

2 Problems Statement and Related Works

The concept of the Product Life Cycle (PLC) has been introduced since the 1950s, it describes every phase a product of interest goes through, from the first initial requirement until it is retired and disposed [2]. In the early 1980s the problems of locating the required data and losing control of change process associated with these data became increasingly intense [3]. During the 1990s, the concept of Product Lifecycle Management (PLM) is proposed, which aims at providing a shared platform for facilitating the process of capturing, representing, organizing, retrieving and reusing the knowledge concerning the related product in or across enterprises. It should provide the integration strategies and technological supports to bring together all existing enterprise systems that deal with the product [4]. Therefore, product-related knowledge becomes one of the critical concepts in the PLM.

In the so-called DIKW Pyramid [5], referring to a hierarchical model for representing the structural relationships between Data, Information, Knowledge and Wisdom, knowledge is considered as the awareness of things that brings to its owner the capability of grasping the meaning from the information [6]. Knowledge is obtained through certain learning behaviors, in which, the external information from the real world is sublimated and becomes the awareness. In our research, we consider knowledge as a kind of invisible thing, which can only be captured by expressing it into multifarious forms of representations. In some of the literature about knowledge, researchers categorized it into two kinds: (1) tacit knowledge, which is highly personal, difficult to articulate and to formalize; (2) explicit knowledge which is easier to be expressed formally and systematically [7]. According to the main theses that Polany used to define the concept of knowledge, all knowledge is rooted in tacit knowledge

[8]. Therefore, we argue that knowledge is an internal awareness that is only explicit to its owner but remains tacit to external world.

In a PLM environment, we consider that all the relevant resources produced by different stakeholders through multifarious enterprise systems are Knowledge Representations (KRs). They act as the carriers of the stakeholders' knowledge to assist further collaboration activities. As the foundation of collaboration, interoperability signifies the ability of diverse entities being able to exchange and make use of those KRs, which is being categorized into five possible levels [9]: encoding level, lexical level, syntactic level, semantic level and semiotic level. While lexical and syntactic issues are now formally solved by many standards, enabling a seamless semantic interoperability remains a huge challenge [10]. The semantics that are contained in a KR is composed of two kinds: (1) explicit semantics, which is directly expressed in the KR; (2) implicit semantics, in opposite, which is hidden. Therefore enterprise systems and stakeholders, in order to cope with the semantic interoperability issue, need to overcome two important obstacles: (1) the implicit semantics that is necessary for understanding a KR is not being made explicit; (2) the lack of automatic semantic verification mechanism to guaranteed the correctness of explicit semantics in a KR.

Due to the essential of ontology, which is defined as a formal and shared understanding of some domains of interest, which specifies the concepts and the relationships that can exist for an agent or a community of agents [19], semantic annotation [11] is usually considered as a possible solution to deal with these two obstacles. Therefore, in our context, semantic enrichment is considered as the process of turning the implicit semantics explicit through the semantic annotations. Some questions are then emerging from the need for semantic enrichment in a PLM environment: What kinds of KR in a PLM environment need semantic enrichment? What kinds of ontology can be used to make the implicit semantics explicit from those KRs? What are the essential elements of a semantic annotation? What kind of policy is needed to detect the possible inconsistencies between semantic annotations and to identify possible mistakes among the annotated model elements?

The concept annotation is defined in the oxford dictionary as "a note by way of explanation or comment added to a text or diagram". Besides this basic meaning, a semantic annotation has two more important features: (1) the machines can read and process it; and (2) it contains a set of formal and shared terms for a certain domain [11]. Semantic annotations use formal knowledge, like ontologies, to capture annotator's knowledge and then they act as a knowledge carrier to enrich annotated object's semantics. Lots of efforts have been made to enrich the semantics of different KRs, such as, natural languages, images, web services and so on. Nevertheless, some shortcomings can be noted: (1) the formalization of semantic annotations is not the research focus in [14], [15], and [17], in which, it is only considered as a simple one-to-one link. Even if there is a specific formalization in [12], [13] and [16], it is difficult to be reused on other types of models but the studied ones; (2) some researches, such as [14] and [15], lack of mechanisms to perform inference to guarantee the consistency among semantic annotations; (3) the domain semantics of the annotated object is the only concern in [16] and [17], in which, the structure semantics (information that describes how the model should be built) is being ignored, or vice-versa. Based on the above literature analysis, in the context of the PLC, we focus our work on clearly identifying the essential of a semantic annotation by proposing a formalization that can be used to enrich both domain and structure semantics of different types of models, and to facilitate and to assist knowledge management in a PLM environment.

3 Solution Proposal

Based on the research context, we consider the semantic annotation as a way that uses one or several ontology-based knowledge representations to make explicit the semantics of various target knowledge representations. In our case the chosen domain is the PLM. The components of the semantic annotation can be divided into three parts: Target Knowledge Representation (TKR), Ontology-based Knowledge Representation (OKR) and Semantic Annotation Structure Model (SASM). The interoperation process between enterprise systems and stakeholders not only requires that the models can be exchanged and operated on, but it also demands the unambiguous understandings of the exchanged models. Therefore, the necessary and sufficient semantics in those models must be made explicit. The knowledge that is being captured and represented inside the Ontology-based Knowledge Representation can be categorized into two aspects: the domain semantics one and the structure semantics one. The domain semantics aspect of any OKRs describes the context and the meaning of some objects in a certain domain [18]. The structure semantics aspect of any OKRs describes how to build the specific models. In our research, The OKRs spread both domain and structure aspects of knowledge. Because of the objective of this paper, we assume that all the OKRs that we need for semantic annotations are already collected, formalized and pre-processed. The SASM presents into eight definitions the formalization of the semantic relations between TKRs and OKRs.

Definition 1. Semantic annotation SA is defined as: $SA := < E, P, SR, MME, MR >$ where: E is a set of elements from a TKR; P is a set of selected element sets from one or several OKRs, which makes explicit the domain semantics aspect of E. SR is a set of binary relations that describes the semantic relations between the elements in E and the elements in P. MME is a set of elements from one OKR, which makes explicit the structure semantics aspect of E. MR is a set of binary relations that describes the semantic relations between the elements in E and the elements in MME. Therefore, a semantic annotation $\mathrm{sa}_x \in \mathrm{SA}$ is defined as: $\mathrm{sa}_x = \{(e_i, p_j, sr_k, mme_l, mr_k) | e_i \in E, p_j \in P, sr_k \in SR, mme_l \in MME, mr_k \in MR\}$.

Definition 2. E is a set of elements from a TKR. Each element $e_i \in E$ has its own domain and structure semantics. The responsibility of the two sets: P and MME is to make explicit the domain semantics and the structure semantics of E.

Definition 3. Let $\mathrm{o_x}$ represents an ontology, which can be generally defined as a 4-tuple: $< \mathrm{C_{o_x}}, \mathrm{R_{o_x}}, \mathrm{I_{o_x}}, \mathrm{A_{o_x}} >$ where $\mathrm{C_{o_x}}$ is a set of concepts; $\mathrm{R_{o_x}}$ is a set of relationships; $\mathrm{I_{o_x}}$ is a set of instances; $\mathrm{A_{o_x}}$ is a set of axioms. Moreover, we assume that $\mathrm{oall_{o_x}}$ is a set that contains all elements (concept, relation, instance or axiom) of $\mathrm{o_x}$.

Definition 4. P is a set of selected element sets from one or several OKRs that makes explicit the domain semantics aspect of E. It can be considered as a subset of the power set of ontology elements. P is defined as follow:$P \subseteq P\left(\bigcup_{o_x \in O} oall_{o_x}\right)$, $\bigcup_{o_x \in O} oall_{o_x} = \{oe_{o_{xy}} | (\exists o_x)(o_x \in O \wedge oe_{o_{xy}} \in oall_{o_x})\}$.

Different from other semantic annotation methods, an e_i in a semantic annotation is not only annotated by one $oe_{o_{xy}}$, but also annotated by a set of selected mandatory ontology elements that are related to it. Given $p_j \in P$ and let use the defined p_j as the minimal set of ontology elements that is created by an annotator or a mechanism (such as semantic block identification [20]), who considers this shape of ontology elements contains the necessary and sufficient semantics for explaining the corresponding e_i.

Definition 5. SR is a set of binary relations that describes the domain semantic relations between $e_i \in E$ and $p_j \in P$. Given $sr_k \in SR$, $sr_k \subset E \times P$ is defined by the notation $e_i\, sr_k p_j$ meaning that $(e_i, p_j) \in sr_k$. Formally, sr_k is defined as: $sr_k = \{(e_i, p_j) | \text{the semantics of } e_i \text{ is in the relation } sr_k\ to\ the\ semantics\ of p_j\}$, where E is the domain of sr_k, denoted by $dom\, sr_k$, and P is the range of sr_k, denoted by $ran\, sr_k$: $dom\, sr_k = \{e_i | (\exists e_i)(\, e_i \in E \wedge (\, e_i, p_j) \in \ sr_k)\}$, $ran\, sr_k = \{p_j | (\exists p_j)(\, p_j \in P \wedge (\, e_i, p_j) \in \ sr_k)\}$. To be more specific, we define the binary relations as: "is equivalent to" ($sr_{\sim}$), "subsumes" ($sr_{\sqsupset}$), "is subsumed by" ($sr_{\sqsubset}$), "intersects" ($sr_{\sqcap}$) and "is disjoint with" ($sr_{\perp}$). SR is defined with the following notations: $SR = \{sr_{\sim},\ sr_{\sqsupset}, sr_{\sqsubset}, sr_{\sqcap}, sr_{\perp}\}$.

The "is equivalent to" relation on two sets E and P denotes the fact that the related two elements from the domain and the range are semantically equivalent. Relation $sr_{\sim} \subset E \times P$ is defined as: $sr_{\sim} = \{(e_i, p_j) | the\ semantics\ of\ e_i\ and \text{ the semantics of } p_j \text{ are equivalent}\}$.

The "subsumes" relation on two sets E and P denotes the fact that the semantics of the related element from the domain is more general than the one of the related element from the range. Relation $sr_{\sqsupset} \subset E \times P$ is defined as: $sr_{\sqsupset} = \{(e_i, p_j) | the\ semantics\ of\ e_i \text{ is more general than } the\ semantics\ of\ p_j\}$.

Conversely, the "is subsumed by" relation on two sets E and P denotes the fact that the semantics of related elements from the domain is less general than the semantics of the related elements from the range. Relation $sr_{\sqsubset} \subset E \times P$ is defined as: $sr_{\sqsubset} = \{(e_i, p_j) | the\ semantics\ of\ e_i\ is\ less\ general\ than\ the\ semantics\ of\ p_j\}$.

The "intersects" relation on two sets E and P denotes the fact that the related two elements from the domain and the range have only a part of common semantics. In our case, we specify that this relation doesn't cover the binary relations of $sr_{\sim}$, $sr_{\sqsupset}$ and $sr_{\sqsubset}$. Relation $sr_{\sqcap} \subset E \times P$ is defined as: $sr_{\sqcap} = \{(e_i, p_j) | \, e_i \text{ and } p_j \text{ have some common semantics}\}$, $sr_{\sqcap} \cap sr_{\sim} = \emptyset$, $sr_{\sqcap} \cap sr_{\sqsupset} = \emptyset$, $sr_{\sqcap} \cap sr_{\sqsubset} = \emptyset$.

The "is disjoint with" relation on two sets E and P denotes that the related two elements from the domain and the range have not common semantics. Relation $sr_{\perp} \subset E \times P$ is defined as: $sr_{\perp} = \{(e_i, p_j) | \, e_i \text{ and } p_j \text{ have not common semantics}\}$.

Definition 6. A meta-model is a model that specifies the concepts, relationships and rules to model a model [21]. Given a meta-model mm_x, it can be generally defined as a 3-tuple: $< C_{mm_x}, R_{mm_x}, RU_{mm_x} >$ where C_{mm_x} is a set of concepts; $R_{mm_x}$is a set of relationships; RU_{mm_x}a set of rules.

Definition 7. MME is a set of elements from one OKR, which makes explicit the structure semantics aspect of E. Each element $mme_l \in MME$ is defined as: $MME = \{mme_l | mme_l \in C_{mmo_x} \cup R_{mmo_x} \cup I_{mmo_x} \cup A_{mmo_x}\}$. An e_i in a semantic annotation is annotated by one mme_l.

Definition 8. MR is a set of binary relations that describes the structure semantic relations between $e_i \in E$ and $mme_l \in MME$. Given $mr_k \in MR$, $mr_k \subset E \times MME$ is defined by the notation $e_i mr_k mme_l$, meaning that $(e_i, mme_l) \in mr_k$. Formally, mr_k is defined as: $mr_k = \{(e_i, mme_l) | e_i \; is \; in \; the \; relation \; mr_k \; to \; mme_l\}$
where E is the domain of mr_k, denotes as $dom\, mr_k$, and MME is the range of mr_k, denotes as $ran\, mr_k$: $dom\, mr_k = \{e_i | (\exists e_i)(e_i \in E \wedge (e_i, mme_l) \in mr_k)\}$. $ran\, mr_k = \{mme_l | (\exists mme_l)(mme_l \in MME \wedge (e_i, mme_l) \in mr_k\}$.

In this paper, we defined this binary relation as: "is instance of" (mr_{io}). MR is defined as follow notation: $MR = \{mr_{io}\}$.

This formalization not only describes the semantic relations between TKR and OKRs, but also acts as the foundation for inconsistency detection and mistake identification. To cope with this objective, SAs are further classified into two types: (1) initial SA, which directly annotates an element in TKR by an annotator; (2) inferred SA, which is suggested to annotate an element in TKR through the inference of the related element's SA and certain rules. Based on the outcomes, three types of possible results can be identified: result (a) expresses that sa_x and sa_y are consistent with each other; result (b) expresses that there is a possible inconsistency between sa_x and sa_y; result (c) expresses that there is an inconsistency between sa_x and sa_y. These results not only detect the inconsistencies (or possible inconsistencies) between two SAs, but then can also be used to identify the possible mistakes among those annotated elements in TKR. In order to apply the above formalization of semantic annotation into a PLC context, a semantic annotation framework (SAF) is proposed. There are four main modules in the SAF: the Semantic Annotation and Processing Agent (SAPA), the OKR Creation and Management (CM) module, the Knowledge Cloud (KC) module and the Reasoning Engine (RE) module. Some processes, that describe a PLC, represent the use of various kinds of enterprise systems (ESs) to manage their own TKRs. KC is the core of the SAF, which is composed of a set of OKRs that are collected, formalized and generated by the CM module. RE is an external call pattern-matching search engine that is in charge of performing inferences to answer different reasoning requests according to some predefined rules. The SAPA acts as a mediator to support the communications between ESs in different processes of the PLC and three other modules in SAF.

4 Case Study

The proposed solution for semantic enrichment of models through semantic annotations is illustrated by a case study that deals with the semantic interoperability requirement of a process model. This PLC scenario is provided by an educational production site: AIPL[1], in which, MEGA[2] is used to design the initial manufacturing process models and to provide product engineers with a global view of the production phases in the PLC. A segment of the manufacturing processes in the AIPL product lifecycle is chosen as the context of this case study. There are five main processes: 1) The bar cutting process; 2) The base turning process; 3) The disc cutting process; 4) The part sticking process; 5) The product assembling process. The knowledge related to the above processes is formalized into the AIPL product ontology. Furthermore, two more OKRs are also integrated into the Knowledge Cloud: MSDL Ontology, which describes the manufacturing capability at the supplier level, process level and machine level of abstraction and the BPMN Ontology [22]

The prototype of the semantic annotation tool, SAP-KM (Semantic Annotation Plugin for Knowledge Management), has been developed as a plugin in the MEGA modeling application. It supports the semantic enrichment of process models and shows the possibility of applying this proposed solution into the reality. To be more specific we take the sub process “Usinage” with its input “Rondelle” and output “Palet” in this process model as an example. The “Usinage” is annotated by sa_1 that states that its domain semantics is more general than the semantics of p_1 and it is an instance of mme_1. The “Rondelle” is annotated by sa_2 that states that its domain semantics is more general than the semantics of p_2 and it is an instance of mme_2. After the reasoning based on these two initials semantic annotations and rules (SWRL Rules created in the preprocessing phase), the inference engine Pellet and predefined policies in the prototype suggest two inferred SAs. The “Rondelle” is suggested to be annotated with sa_3 that states that its domain semantics is more general than the semantics of p_3 and it is an instance of mme_3. The “Palet” is suggested to be annotated by sa_4 that states that its domain semantics is more general than the semantics of p_4 and it is an instance of mme_4. Because both sa_2 (initial) and sa_3 (inferred) are SAs of the “Rondelle”, based on the list of possible results of inconsistency detection between two SAs the semantics similarity comparison outcome $p_2 \perp p_3$ signifies a possible inconsistency between sa_2 and sa_3.

5 Conclusion

Because of the initial design objective of the modeling languages and the diversity of expressions of models, the inaccurate and inconsistent semantics among the model elements are difficult to be identified. With the supports of semantics enrichment solutions, this paper shows the possibility of using formal semantic annotations to enrich the enterprise models in a PLM environment for facilitating semantic interoperability issue during the knowledge management process. A semantic annotation framework is presented to describe and generate interactions among its four main modules. A prototype annotation tool, SAP-KM, is designed and implemented to instantiate the formal semantic annotations and to demonstrate its applicability and

[1] AIPL `http://www.aip-primeca.net/` - Atelier Inter-Établissements de Productique Lorrain.

[2] MEGA `http://www.mega.fr/en`

usability of our semantic enrichment solution. The further development of this prototype will be concentrated on the iterated suggestion and real-time verification.

References

1. Ameri, F., Dutta, D.: Product lifecycle management: closing the knowledge loops. Comput.-Aided Des. Appl. 2, 577–590 (2005)
2. Rink, D.R., Swan, J.E.: Product life cycle research: A literature review. J. Bus. Res. 7, 219–242 (1979)
3. CIMdata: PDM to PLM: Growth of an Industry (2003)
4. Ball, A., Ding, L., Patel, M.: An approach to accessing product data across system and software revisions. Adv. Eng. Informatics 22, 222–235 (2008)
5. Zeleny, M.: Management support systems: towards integrated knowledge management. Human Systems Management 7, 59–70 (1980)
6. Ackoff, R.L.: From Data to Wisdom. J. Applies Syst. Anal. 16, 3–9 (1989)
7. Ropo, A., Parviainen, J.: Leadership and bodily knowledge in expert organizations:: epistemological rethinking. Scand. J. Manag. 17, 1–18 (2001)
8. Polanyi, M.: The tacit dimension. Peter Smith, Gloucester (1983)
9. Euzenat, J.: Towards a principled approach to semantic interoperability. In: IJCAI 2001, Workshop on Ontology and Information Sharing, Seattle, USA, pp. 19–25 (2001)
10. Panetto, H.: Towards a classification framework for interoperability of enterprise applications. Int. J. Comput. Integr. Manuf. 20, 727–740 (2007)
11. Boudjlida, N., Panetto, H.: Annotation of enterprise models for interoperability purposes. In: IEEE International Workshop on Advanced Information Systems for Enterprises, IWAISE 2008, Constantine, Algeria, pp. 11–17 (2008)
12. Lin, Y.: Semantic annotation for process models: Facilitating process knowledge management via semantic interoperability, PhD Thesis (2008)
13. Di Francescomarino, C.: Semantic annotation of business process models, PhD Thesis (2011)
14. Attene, M., Robbiano, F., Spagnuolo, M., Falcidieno, B.: Characterization of 3D shape parts for semantic annotation. Comput.-Aided Des. 41, 756–763 (2009)
15. Li, C.: Ontology-Driven Semantic Annotations for Multiple Engineering Viewpoints in Computer Aided Design, PhD Thesis (2012)
16. Bergamaschi, S., Beneventano, D., Po, L., Sorrentino, S.: Automatic Normalization and Annotation for Discovering Semantic Mappings. In: Ceri, S., Brambilla, M. (eds.) Search Computing II. LNCS, vol. 6585, pp. 85–100. Springer, Heidelberg (2011)
17. Song, F., Zacharewicz, G., Chen, D.: An ontology-driven framework towards building enterprise semantic information layer. Adv. Eng. Informatics 27, 38–50 (2012)
18. Maedche, A., Staab, S.: Measuring Similarity between Ontologies. In: Gómez-Pérez, A., Benjamins, V.R. (eds.) EKAW 2002. LNCS (LNAI), vol. 2473, pp. 251–263. Springer, Heidelberg (2002)
19. Gruber, T.R.: A translation approach to portable ontology specifications. Knowl. Acquis. 5, 199–220 (1993)
20. Lezoche, M., Yahia, E., Aubry, A., Panetto, H., Zdravković, M.: Conceptualising and structuring semantics in cooperative enterprise information systems models. Comput. Ind. 63, 775–787 (2012)
21. OMG: Meta Object Facility Core Specification, `http://www.omg.org/spec/MOF/2.4.1/`
22. Ghidini, C., Rospocher, M., Serafini, L.: A formalisation of BPMN in description logics. Technical report (2008)

Discovering High-Level Performance Models for Ticket Resolution Processes
(Short Paper)

Francesco Folino, Massimo Guarascio, and Luigi Pontieri

Institute ICAR, National Research Council of Italy (CNR)
Via Pietro Bucci 41C, I87036 Rende (CS), Italy
{ffolino,guarascio,pontieri}@icar.cnr.it

Abstract. Predicting run-time performances is a hot issue in ticket resolution processes. Recent efforts to take account for the sequence of resolution steps, suggest that predictive Process Mining (PM) techniques could be applied in this field, if suitably adapted to the peculiarities of ticket systems. In particular, the performances of a ticket instance usually depend on which kinds of experts worked on it (more than on the mere sequence of resolution tasks), while relevant information about ticket cases is stored in the form of text fields, which are usually disregarded by PM approaches. Instead of relying on a-priori experts groups, we devise an ad-hoc method for clustering experts according to their real working patterns, based on log data. Regarding the discovered groups as abstractions for log events, we also perform a predictive clustering of ticket cases, while using context data as input attributes for splitting the tickets. In this way, different (context-dependent) execution scenarios are recognized for the process, and equipped with more accurate performance predictors. The approach was validated on a real application scenario, where it showed better results than state-of-the-art solutions.

1 Introduction

Ticket management systems (devoted to manage, maintain, and help resolve issues in an organization) are very popular in various collaboration environments (such as, e.g., CSCW, Social Computing, Bug Tracking systems), and they have been given increasing attention recently, especially as concerns the opportunity to exploit their log data to improve the handling of tickets. So far, most efforts have been spent to extract some kind of recommendation model for improving the allocation of (human) resources. To this end, some approaches just look at the text information (e.g., keywords, notes) stored for each ticket [3], while others also look at the sequence of worker groups each ticket has been progressively assigned to [7,10]. In general, an implicit goal of all the approaches above is to keep as short as possible the length and/or the duration of ticket resolution sequences, while possibly optimizing the usage of resources (i.e., experts or "workers"). Predicting process instances' performances is right the scope of an emerging research stream in the field of Process Mining [1]. The basic idea is to exploit historical log data, in order to induce a state-aware performance model for the analyzed process, based on some log abstraction function (see, e.g., [2,5]). A key point in the induction of predictive performance models (particularly when working with a ticket management process) is right the definition of a suitable log abstraction function. To this regard, most Process Mining approaches tend to rely on activity labels, assuming that they capture well the semantics of each processing step. However, most ticket systems just support generic activities, such as, e.g., open/close/lock/unlock a ticket, bounce/forward it, send a message, add comment, update a ticket's descriptor. Considering the intervention of workers can help gain insight into the resolution stage of a ticket, but looking at the very individuals involved is likely to yield unreadable and imprecise (overfitting) models. On the other hand, the idea of relying on pre-defined worker groups (as in [7,9,10]) may well be unsuitable for many real application contexts, where such groups do not exist or do not reflect the behavior of staff people.

In this work, we propose an approach to the prediction of performances for ticket resolution instances, which overcomes the limitations discussed above, while taking account for both context data and resolution steps. Specifically, in order to get a suitable abstraction level over the performed steps, we only focus on the sequence of workers involved in a ticket, abstracted into high-level groups. An ad-hoc clustering method is devised to discover such groups out of log data (while possibly reusing a-priori knowledge), so that they reflect real different resolution skills and performance profiles. These data-driven groups are the basis for regarding the sequence of events recorded in each ticket-resolution trace at a higher level of abstraction (i.e., as a sequence of worker groups), before inducing a predictive model for the process. Technically, this latter task can be accomplished by integrating a classic state-aware performance prediction method [2] into a predictive clustering scheme [4], as proposed in [5]. Such a clustering of ticket traces can help detect different (context-dependent) ticket handling cases, and equip them all with separate and precise performance predictor. Preliminary results on real-life data show that enhancing this latter approach with our clustering-based event abstraction method allows to build more effective models.

2 Preliminaries

Generally, since each ticket gives rise to a series of activities, it can be viewed as an instance of some (ticket handling) process. We then assume that a record (named *trace*) is kept for each ticket, which stores the sequence of *events* occurred during the corresponding process instance – where each event regards the execution of some activity. In the rest of the paper, we will only focus on manual activities performed by some *worker* (i.e., a member of the staff), including those (usually named "experts") who are in charge of finding a solution to the signaled issue. With little language abuse, we will use "trace", and "ticket" interchangeably, as the former is a direct consequence of the creation of a ticket.

Let $A = \{a_1, \ldots, a_{\#A}\}$ be the tasks composing the (ticket handling) process, $W = \{w_1, \ldots, w_{\#W}\}$ be the workers who can perform them, E be the universe of all events that can happen during the handling of tickets, and $\mathcal{T}$ be the universe of all possible ticket traces. Moreover, for any event $e \in E$, let $task(e)$, $executor(e)$, and $time(e)$ be the activity, executor and timestamp, resp., associated with e.

For each trace $\tau \in \mathcal{T}$, let $len(\tau)$ be the number of events stored in τ, let $\tau[i]$ be the i-th event of τ, for $i = 1 \,..\, len(\tau)$, and $\tau(i] \in \mathcal{T}$ be the *prefix* trace consisting only of the first i events in τ, for any $i = 0, \ldots, len(\tau)$. Finally, a *log* L (over $\mathcal{T}$) is a finite subset of $\mathcal{T}$, while the *prefix set* of L, denoted by $\mathcal{P}(L)$, is the set of all prefix traces that can be extracted from L.

In addition to resolution steps, ticket management systems usually keep several descriptive fields for each ticket instance. Most systems store, for each ticket, several kinds of free-text contents, ranging from notes to messages. Let us assume that the textual contents of any ticket τ is conveyed by a vector $text(\tau)$, encoding a bag-of-word representation of the concatenation of all τ's free-text fields – to this aim, classic vector-space model and TF-IDF weighting are used. Moreover, like in [5], we assume that a vector $context(\tau)$ is associated with any ticket trace τ, encoding all context data available for τ. Note that, besides "intrinsic" properties (explicitly stored for each ticket), a series of "extrinsic" context features (such as, e.g., workload indicators, temporal dimensions), can be derived for each trace, capturing the state of the ticket system when it began.

Performances: Measures and Models. Let us assume that $\hat{\mu} : \mathcal{T} \rightarrow \mathbb{R}$ is an (unknown) function assigning a performance value to any (possibly partial) process instance. Then, a (predictive) *Process Performance Model* (*PPM*) is a model that can forecast the performance value of any process instance, based on the events happened during its enactment. Such a model can be represented as a function $\mu : \mathcal{T} \rightarrow \mathbb{R}$, estimating $\hat{\mu}$ all over the trace universe. Learning a PPM is a special induction problem, where the training set is a log L, and $\hat{\mu}(\tau)$ is known for each (sub-)trace $\tau \in \mathcal{P}(L)$. Recent approaches to the induction of a PPM (e.g., [2, 5]) share the idea of regarding process logs at a suitable level of abstraction, with the help of functions like those described next.

An *event abstraction function* $\mathcal{E} : E \rightarrow \mathcal{A}^E$ is a function mapping each event $e \in E$ to an abstract representation $\mathcal{E}(e)$ in some proper space $\mathcal{A}^E$. Moreover, a *trace abstraction function* $abs_{\mathcal{E}} : \mathcal{T} \rightarrow \mathcal{A}^{\mathcal{T}}$, w.r.t. an event abstraction function $\mathcal{E}$, is a function mapping each trace $\tau \in \mathcal{T}$ to an abstract representation $abs_{\mathcal{E}}(\tau)$ in some space $\mathcal{A}^{\mathcal{T}}$. Let us consider the three trace abstraction functions (practically equivalent to those introduced in [2]) $list_{\mathcal{E}}^h(\tau)$, $bag_{\mathcal{E}}^h(\tau)$, and $set_{\mathcal{E}}^h(\tau)$, which map any trace τ to the *list*, *multiset*, or *set*, respectively, of the h latest events occurring in τ, while abstracting each of these events via function $\mathcal{E}$. Based on such trace abstractions, a PPM is derived in [2] in terms of an annotated finite state machine (named "A-FSM"), where: *(i)* each node corresponds to one abstract trace representation α (produced by $abs_{\mathcal{E}}$), and stores an estimate for the target measure (usually computed as the average over all the trace prefixes matching α), while *(ii)* each transition is labelled with an event abstraction (produced by $\mathcal{E}$). This learning approach was hybridized in [5] with a predictive clustering scheme [4], where context information are used as descriptive features while the targets are derived from performance measurements. As discussed before, the event abstraction functions used in [2, 5] (and in current literature generally) are pretty simple, for they just select some properties of an event e – typically, $executor(e)$, $task(e)$ or both – and risk being ineffective in the case of tickets.

Example 1. Let us introduce a toy example, inspired to the real-life logs considered in Section 5. Let τ and τ' be two traces storing the event sequences $\langle e_1, e_2, e_3, e_4, e_5\rangle$ and $\langle e'_1, e'_2, e'_3, e'_4, e'_5\rangle$, respectively, with: $executor(e_1){=}system$, $executor(e_2){=}executor(e_3){=}john$, $executor(e_4){=}mark$, $executor(e_5){=}system$; $executor(e'_1){=}system$, $executor(e'_2){=}mark$, $executor(e'_3){=}executor(e'_4){=}john$, $executor(e'_5){=}system$. Let also $\mu(\tau(1]) = 6$, $\mu(\tau(2]) = 4$, $\mu(\tau(3]) = 2$, $\mu(\tau(4]) = 1$, $\mu(\tau(5]) = 0$; $\mu(\tau'(1]) = 4$, $\mu(\tau'(2]) = 3$, $\mu(\tau'(3]) = 2$, $\mu(\tau'(4]) = 1$, $\mu(\tau'(5]) = 0$. Consider, as a straightforward event abstraction criterion, a function γ mapping each event e to $executor(e)$. It is easy to see that for the prefix $\tau(3]$ it is: $list^{\infty}_{\gamma}(\tau(2]) = \langle system, john, john\rangle$, $bag^{\infty}_{\gamma}(\tau(2]) = [system, john^2]$, $set^{\infty}_{\gamma}(\tau(3]) = \{system, john\}$. Conversely, when $h = 1$, all abstractions $list^{1}_{\gamma}(\tau(3])$, $bag^{1}_{\gamma}(\tau(3])$ and $set^{1}_{\gamma}(\tau(3])$ will just consist of the sole element $john$, which is indeed the last executor appearing in $\tau(3]$. The A-FSM model derived from the log $\{\tau, \tau'\}$ with $list^{2}_{\gamma}$ will contain the states $\langle system\rangle$, $\langle system, john\rangle$, $\langle john, john\rangle$, $\langle john, mark\rangle$, $\langle mark, system\rangle$, $\langle system, mark\rangle$, $\langle mark, john\rangle$, and $\langle john, system\rangle$, annotated with the performance estimations 5, 4, 3, 1, 0, 3, 2, and 0, respectively.

3 A Clustering Framework for the PPM Discovery

We next define the specific kind of PPM model we want to learn, extending the notion of "Context-Aware PPM" of [5], with the capability of gaining an effective abstraction level over ticket traces.

A *context-aware process performance model* (HO-PPM) for a given log L is a triple of the form $M = \langle abs, part, \langle \mu_1, \ldots, \mu_k\rangle\rangle$, such that: *(i)* abs is a trace abstraction function, *(ii)* $part$ is a trace partitioning function allowing to split log traces into a number, say k, of disjoint clusters, based on their context data, and *(iii)* μ_i is an A-FSM, for $i = 1, \ldots, k$, where the states are labelled with the abstract representations produced by abs. This model encodes a predictive clustering function estimating the unknown performance function $\hat{\mu}$ as follows: $\mu^M(\tau){=}\mu_j(abs(\tau))$ such that $j{=}part(context(\tau))$. In this way, a forecast for any new process instance τ is made by first assigning τ to a cluster (via function $part$, based on $context(\tau)$), and then providing the predictor of that cluster with the abstract representation of τ.

In order to make this model really effective, the event abstraction function must be defined carefully. To this end, we automatically partition workers into performance-relevant groups, through an ad-hoc clustering procedure, while exploiting log information, as a reflection of real workers' behaviors. These groups will then be regarded as high-level representations of log events, by technically defining an event abstraction function which maps each event e to the cluster of the worker associated with e, i.e., with $executor(e)$.

We next illustrate how a dissimilarity measure for workers can be defined, in order to eventually partition them into performance-relevant groups, with the help of whatever distance-based clustering method. Essentially, a rough performance indicator is defined, named *performance footprint*, to capture the impact of a worker on the performances of the tickets she participated to. For significance reasons, footprints are defined in an aggregated way, looking at the participation of workers to a given set of ticket classes, denoted by $C^{\mathcal{T}} = \{c_1^{\mathcal{T}}, \ldots, c_{\#C}^{\mathcal{T}}\}$.

For any ticket trace τ and any worker w appearing in it (as an executor), let $perf(w, \tau)$ be the performance value corresponding to the the last τ's event associated with w. Further, for any ticket class $c \in C^{\mathcal{T}}$ and any worker w, let $c(w)$ be the tickets of c including w. Then, the *performance footprint* of w w.r.t. c is defined as follows: $pf(w, c) = \texttt{NULL}$ if $|c(w)| < \texttt{MIN\_COUNT}$, or $\sum_{\tau \in c(w)} perf(w, \tau)/|c(w)|$, otherwise. Note that in the tests shown later on, MIN_COUNT (minimal nr. of samples for computing $pf(w, c)$) was set to 50 – but a wider range of values worked fine as well. As an instance, let c be a class gathering the two example traces τ and τ' of Example 1, while assuming that $\texttt{MIN\_COUNT} = 0$. It is easily seen that $perf(john, \tau) = \mu(\tau(3]) = 2$, $perf(john, \tau') = \mu(\tau'(4]) = 1$, whereas $perf(mark, \tau) = \mu(\tau(4]) = 1$, $perf(mark, \tau') = \mu(\tau'(2]) = 3$. Then, it is $pf(john, c) = (1 + 2)/2 = 1.5$ and $pf(mark, c) = (1 + 3)/2 = 2$.

Often, background knowledge is available concerning the relationships between workers and organization-oriented concepts (such as roles, teams, capabilities and skills). W.l.o.g., we assume that such information is encoded through a function org_O, mapping each worker w to the set $org_O(w)$ of organizational concepts she is associated with. Hereinafter, we will call $org_O(w)$ the (a-priori) *organizational profile* of worker w, and org_O an *organizational-profiling* function.

We can now define the overall distance measure for partitioning the executors of ticket management activities into disjoint clusters, each corresponding to some specific combination of capabilities and performance profiles.

Definition 1 (Workers' Distance). Let L be a ticket management log, $C^{\mathcal{T}}$ be a partitioning of all the tickets in L. Moreover, let $org_O : W \to 2^O$ be an organizational-profiling function (w.r.t. a given set O of organizational concepts). Then, the distance $dist^W(w_1, w_2)$ between two any workers $w_1, w_2 \in W$ (w.r.t. org_O, L and $C^{\mathcal{T}}$) is computed as a linear combination of three functions: $dist^W(w_1, w_2) = \beta \times dist_O(w_1, w_2)) + (1-\beta) \times [dist_A(w_1, w_2)) + dist_P(w_1, w_2)]/2$. Function $dist_O(w_1, w_2))$ (resp., $dist_A(w_1, w_2)))$ simply corresponds to the Jaccard distance between the sets of organizational concepts associated with (resp., of tasks executed by) w_1 and w_2, while the third (footprint-based) distance is defined as follows: $dist_P(w_1, w_2)) = \sqrt{\sum_{c \in C^{\mathcal{T}}} \delta\left(pf(w_1, c), pf(w_2, c)\right)^2 / |C^{\mathcal{T}}|}$, with $\delta(x, y) = \frac{|x-y|}{\max\{pf(w,c) | w \in W, c \in C^{\mathcal{T}}\}}$ if $\texttt{NULL} \notin \{x, y\}$, and $\delta(x, y) = 1$ otherwise; $\square$

4 Computation Approach: Algorithm HOPP

An algorithm, named HOPP (i.e., High-Order Performance Prediction), for extracting an HO-PPM from a ticket log, is shown in Figure 1. Besides historical log traces (and associated performance measurements), the algorithm takes as input a set of workers, along with their organizational profiles, and an initial partition

```
Input: A log $L$ over ticket (trace) universe $\mathcal{T}$, a target measure $\hat{\mu}$ known over $\mathcal{P}(L)$, an a-priori
       partition $C^{\mathcal{T}}$ of tickets in classes, a set $O$ of (organznl.) concepts, a set $W$ of workers,
       a function $org : W \rightarrow 2^O$, $mode \in \{\mathtt{LIST}, \mathtt{SET}, \mathtt{BAG}\}$, $h \in \mathbb{Z}$, and $\beta \in [0, 1]$.
Output: An HO-PPM model for $L$ (fully encoding $\hat{\mu}$ all over $\mathcal{T}$).
Method: Perform the following steps:
 1 Derive $context(\tau)$ for each $\tau \in L$, by computing $text(\tau)$, $data(\tau)$ and $env(\tau)$;
 2 Compute $pf(w, c)$ and $tasks_L(w)$ for each $w \in W$ and $c \in C^{\mathcal{T}}$;
 3 Compute a clustering $C^W$ of workers, by using the distance in Def. 1;
 4 Define an event abstraction function $\gamma : E \rightarrow C^W$ s. t. $\gamma(e)$ is the group including $executor(e)$;
 5 Let $abs$= $list^h_\gamma$ (resp., $set^h_\gamma$, $bag^h_\gamma$) if $mode$ = LIST (resp., SET, BAG);
 6 Let $AS = \{a_1, ..., a_k\}$ be the set of all trace abstractions produced by $abs$ on $L$
   – i.e., $AS = \{abs(\tau) \mid \tau \in \mathcal{P}(L)\}$
 7 Learn a PCT model $T$, using $context(\tau)$ (resp., $val(\tau, a_i)$, $i$=1..$k$) as descriptive
   (resp., target) features $\forall \tau \in L$;[a]
 8 Let $part_T$ be the partitioning function of $T$ and let $L[1], \ldots, L[q]$ be the clusters
   obtained by applying $part_T$ to $L$;
 9 for each $L[i]$ do Induce an A-FSM model $\mu_i$ (w.r.t. $abs$) out of $L[i]$ end;
10 return $\langle abs, part_T, \langle \mu_1, \ldots, \mu_k \rangle \rangle$
```

[a] $val(\tau, \alpha)$ = NULL if $matchs(\tau, \alpha) = \emptyset$, or $val(\tau, \alpha) = \hat{\mu}(\max(matchs(\tau, \alpha)))$, otherwise; where $matchs(\tau, \alpha)$ is the set of τ's prefixes matching a given trace abstraction α.

Fig. 1. **Algorithm** HOPP

$C^{\mathcal{T}}$ of tickets into classes (possibly capturing different kinds of ticket resolution problems). Parameters *mode* and h let the user choose the trace abstraction for building all A-FSM models, while β controls the relative weight of background knowledge in the clustering of workers.

Notice, in particular, that the (preliminary) partitioning of tickets allows to compute the (aggregated) performance footprints of workers, and to carry out a clustering over the domain of workers. To this end, our current implementation uses algorithm *X-means* [8] with the distance in Def. 1. If no a-priori ticket clustering is given, one can compute it with *X-means* – as done in our tests.

The discovered worker groups are then used to transform the original log, by replacing each log trace with a higher-level version of it (Step 5), where each log event e is mapped to the group of the worker who executed e. This abstracted version of the log is then exploited to extract a HO-PPM, using the context features of tickets as descriptive attribute, and performance values as target features. This task is carried out by following the approach in [5], where the discovery of such a model is accomplished in two phases: first a *Predictive Clustering Tree (PCT)* [4] is induced (Step 7) from a propositional view of the log; then the method in [2] is used to equip each of the discovered clusters with an *A-FSM* model (Step 9) – while using workers groups for event abstraction.

5 Case Study

This section illustrates the results of an experimental activity, performed to asses the validity of the proposed approach. This empirical analysis was carried out on a real-life scenario, concerning the handling of tickets in the IT department of an Italian enterprise. The typical lifecycle of a ticket, in the analyzed ticket

Table 1. Remaining steps' prediction: error reductions (%) of `HOPP` w.r.t. `CA-TP` and `FSM`, when using different trace abstraction functions (while fixing $\beta = 0.5$)

Trace abstraction function		*Δ*% **w.r.t. CA-TP [5]**			*Δ*% **w.r.t. FSM [2]**		
mode	*h*	*rmse*	*mae*	*mape*	*rmse*	*mae*	*mape*
`LIST`	2	-17.8%	-5.4%	-8.4%	-19.1%	-15.3%	-31.0%
	4	-31.2%	-33.4%	-26.4%	-34.1%	-45.8%	-48.4%
	8	-45.7%	-47.2%	-40.4%	-47.7%	-55.5%	-56.4%
	16	-48.2%	-49.5%	-43.5%	-50.0%	-57.2%	-58.9%
	Total	**−35.7%**	**−33.9%**	**−29.6%**	**−37.7%**	**−43.4%**	**−48.7%**
`BAG`	2	-15.5%	-7.5%	-8.8%	-17.1%	-14.6%	-21.4%
	4	-29.2%	-35.3%	-21.6%	-30.8%	-45.0%	-38.9%
	8	-42.2%	-45.8%	-39.2%	-44.0%	-52.2%	-52.0%
	16	-47.5%	-50.0%	-43.3%	-49.1%	-55.8%	-55.3%
	Total	**−33.6%**	**−34.7%**	**−28.2%**	**−35.2%**	**−41.9%**	**−41.9%**
Grand Total		**−34.7%**	**−34.3%**	**−28.9%**	**−36.5%**	**−42.7%**	**−45.3%**

management system, can be summarized as follows. As soon as a problem is reported, a ticket is created by the help desk with a short description of the problem signaled. The ticket is then sent via email to one expert, based on predefined (and partly implicit) allocation schemes. If the worker solves the problem, she closes the ticket; otherwise, she forwards it to another worker, while possibly making her informed about the handed-over case. Several kinds of data are stored for a ticket, including: the actions performed on it and their executors, a priority and a severity level, and general categories for the reported issue (e.g., logging-in problems, service outage, etc.). Moreover, as to free-text ticket contents, we considered the ticket summary (describing its nature and status) and the subjects and bodies of all messages exchanged between people working on it. These textual contents were summarized into a selection of keywords — precisely, the top 500 stemmed terms w.r.t. IDF scores, occurring in at least 10 tickets. The workers involved in the resolution of tickets were structured in 50 groups. In our tests, we focused on 2286 tickets handled in 2 consecutive months of year 2012. Since no a-priori partitioning $C^{\mathcal{T}}$ of the tickets was given, we computed it by applying *X-means* [8] on ticket data, while using an ad-hoc contents-oriented distance function for comparing them. Specifically, the distance between two tickets was computed by combining the *cosine* distance between the (vector space representations of) their respective text contents, with a mismatch-based distance between their respective data fields. As categorical attributes usually correspond to predefined categories (assigned by help-desk staff), we used the same factor β, as for workers' clustering, to control their influence on distance evaluation — the higher β, the higher the influence.

In order to quantify prediction accuracy, we used the same error metrics as in [2] — namely, *rmse*, *mae*, and *mape* (mean absolute percentage error) — measured via a (10 fold) cross-validation procedure. Table 1 reports the error reduction (in percent, denoted by *Δ*%) achieved by the algorithm `HOPP` when it is applied to the dataset described above to predict the resolution steps needed for a ticket w.r.t. two state-of-the-art approaches denoted by `CA-TP` [5] and `FSM` [2], respectively. The tests were performed by setting the value of parameter β (i.e., the weight given to background categories in the clustering of workers/tickets) to 0.5 – although the method seems to be quite robust to value of β taken out of $\{0.25, 0.5, 0.75\}$ – and different kinds of trace abstraction functions, specified through the parameters h and $mode \in \{\texttt{LIST}, \texttt{BAG}\}$. Even if the advantage of using our solution are already appreciable with $h = 2$, higher horizon values yield better performances. The effect of the abstraction mode is not very marked, seeing as very similar (good) results are found in both cases. In fact, whatever h, less than 2.5% error reduction is obtained (on all metrics) when moving from bags to lists, for all metrics but the *mape* – which shrinks of nearly 7%.

6 Conclusion

The paper presented an approach to the prediction of performances for ticket resolution instances. The approach features several innovative aspects, including: (i) the exploitation of textual contents (ignored by current process mining methods), and (ii) a data-driven (clustering-based) event abstraction procedure. As to future work, we plan to integrate the approach into a real ticket management system, and to try to make the clusterings of tickets and workers more synergistic – e.g., via some multi-way co-clustering scheme (like that in [6]).

References

1. van der Aalst, W.M.P., van Dongen, B.F., Herbst, J., Maruster, L., Schimm, G., Weijters, A.J.M.M.: Workflow mining: a survey of issues and approaches. Data & Knowledge Engineering 47(2), 237–267 (2003)
2. van der Aalst, W.M.P., Schonenberg, M.H., Song, M.: Time prediction based on process mining. Information Systems 36(2), 450–475 (2011)
3. Anvik, J., Hiew, L., Murphy, G.C.: Who should fix this bug? In: Proc. of 28th Int. Conf. on Software Engineering (ICSE 2006), pp. 361–370 (2006)
4. Blockeel, H., Raedt, L.D.: Top-down induction of first-order logical decision trees. Artificial Intelligence 101(1-2), 285–297 (1998)
5. Folino, F., Guarascio, M., Pontieri, L.: Discovering context-aware models for predicting business process performances. In: Meersman, R., et al. (eds.) OTM 2012, Part I. LNCS, vol. 7565, pp. 287–304. Springer, Heidelberg (2012)
6. Greco, G., Guzzo, A., Pontieri, L.: Co-clustering multiple heterogeneous domains: Linear combinations and agreements. IEEE Transactions on Knowledge and Data Engineering 22(12), 1649–1663 (2010)
7. Miao, G., Moser, L.E., Yan, X., Tao, S., Chen, Y., Anerousis, N.: Generative models for ticket resolution in expert networks. In: Proc. of 16th Int. Conf. on Knowledge Discovery and Data Mining (KDD 2010), pp. 733–742 (2010)
8. Pelleg, D., Moore, A.W.: X-means: Extending k-means with efficient estimation of the number of clusters. In: Proc. of 17th Int. Conf. on Machine Learning (ICML 2000), pp. 727–734 (2000)
9. Shao, Q., Chen, Y., Tao, S., Yan, X., Anerousis, N.: Efficient ticket routing by resolution sequence mining. In: Proc. of 14th Int. Conf. on Knowledge Discovery and Data Mining (KDD 2008), pp. 605–613 (2008)
10. Sun, P., Tao, S., Yan, X., Anerousis, N., Chen, Y.: Content-aware resolution sequence mining for ticket routing. In: Hull, R., Mendling, J., Tai, S. (eds.) BPM 2010. LNCS, vol. 6336, pp. 243–259. Springer, Heidelberg (2010)

Change Patterns for Supporting the Evolution of Event-Based Systems

Simon Tragatschnig, Huy Tran, and Uwe Zdun

Research Group Software Architecture
University of Vienna, Austria
{simon.tragatschnig,huy.tran,uwe.zdun}@univie.ac.at

Abstract. As event-driven architectures consist of highly decoupled components, they are a promising solution for facilitating high flexibility, scalability, and concurrency of distributed systems. However, the evolution of an event-based system is often challenging due to the intrinsic loose coupling of its components. This problem occurs, on the one hand, because of the absence of explicit information on the dependencies among the constituting components. On the other hand, assisting techniques for investigating and understanding the implications of changes are missing, hindering the implementation and maintenance of the changes in event-based architectures. Our approach presented in this paper aims at overcoming these challenges by introducing primitive change actions and higher-level change patterns, formalized using trace semantics, for representing the modification actions performed when evolving an event-based system. Our proof-of-concept implementation and quantitative evaluations show that our approach is applicable for realistic application scenarios.

1 Introduction

Event-driven architectures are a promising solution for developing distributed systems that facilitates high flexibility, scalability, and concurrency [6, 8]. An event-based system consists of a number of computational or data components that communicate with each other by emitting and receiving events [8]. Each component may independently perform a particular task, for instance, accessing a database, checking a credit card, interacting with users, or writing to a log file. The execution of a component can be triggered by some particular events, which are called the *input events*. In turn, a component can also emit one or many *output events*. The transfer of events among the components is performed through an event channel. Therefore, every component is totally unaware of the others. This way, the event-based communication style can support a high degree of flexibility. For instance, it enables replacing or altering any component (e.g., with a bug-fixed or upgraded version) or to change the execution order of the components (e.g., re-routing, skipping, or adding some components) whilst the system is running. There is a rich body of work in different research areas that investigate and exploit the prominent advantages of event-based communication styles such as middleware infrastructure [4], event-based coordination [2], active database systems [10], and service-oriented architectures [9], to name but a few.

Unfortunately, by introducing additional degrees of flexibility, the loose coupling in event-based systems also increases the difficulty and uncertainty in maintaining and evolving these systems. Implementing specific changes in an event-based system is challenging, because of the absence of explicit dependencies among constituent components makes understanding and analyzing the overall system composition difficult.

We present in this paper a novel approach for supporting the evolution of event-based systems. In particular, we introduce fundamental abstractions for describing primitive modifications that can be used to alter an event-based system. These primitive actions capture the low-level actions for modifying event-based systems, and therefore, can be efficiently leveraged by technical experts. On top of these primitive actions, we devise a number of high-level abstractions described as *change patterns* for event-based systems along with their formal descriptions based on trace semantics. These patterns encapsulate essential change actions that recur in many software systems. A certain evolution requirement can be implemented through low-level abstractions by applying an individual or a chain of primitive actions or at higher level of abstractions by using one or more change patterns, respectively. Our quantitative evaluations show the applicability of our approach for realistic application scenarios.

The paper is structured as following. In Section 2, we introduce some preliminary concepts and definitions in the context of event-based software systems and the trace semantics used to formalize our change patterns. Section 3 describes the fundamental concepts and abstractions of our approach for supporting the evolution of event-based systems. Section 4 presents evaluations of its productivity, based on our proof-of-concept implementation. The related literature is discussed in Section 5. We summarize the main contributions and discuss the planned future work in Section 6.

2 Preliminaries

Our approach aims at introducing techniques for implementing changes in event-based systems. Without loss of generality, we adopt the notion that a generic event-based system comprises a number of components performing computational or data tasks and communicating by exchanging events through event channels [8]. The inherent loosely coupled nature of the participating components of an event-based system makes it challenging to understand and implement changes. This issue is, on the one hand, due to the absence of explicit dependency information. On the other hand, software engineers have to deal with the complexity and every technical details of the underlying event-based systems because of the lack of appropriate abstractions for handling and managing changes. To aid the software engineers in better analyzing and applying changes for an event-based system, we make some basic assumptions that slightly reduce the system's non-determinism while still preserving a large degree of the flexibility and adaptability: (1) each component exposes an event-based interface that specifies a set of events that the component expects (aka the *input events*) and a set of events that the component will emit (aka *the output events*); (2) the execution of a component is triggered by its input events; and (3) after a component is executed, it will eventually emit its output events.

These requirements aim at specifying the semantics of the typical behavior of an event-based component and making some pragmatic assumptions that are slightly constraining the non-deterministic nature of event-based systems. Please note that the first requirement does not forbid the alteration of a component's input and output events but only enables us to be able to observe the input/output event information at a certain point in time. The major advantage of this perspective on event-based systems is that it supports extracting dependency information at any time without requiring access to the (currently deployed) source code. In addition, this requirement is pragmatic in case third-party components are used as they are often provided as black-boxes with documented interfaces.

Please also note that these requirements can be satisfied for most event-based components without change or with reasonable extra costs (e.g., for developing simple wrappers in case of using third-party libraries and components). Most of the existing event-based systems already support equivalent concepts [8]. For demonstration purpose, we leverage the Dynamic Event-driven Actor Runtime Architecture (DERA) framework [16] that provides basic concepts for modeling and developing event-based systems and supports the three requirements. The DERA concepts can easily be generalized to the concepts found in other event-based systems.

In DERA, a computational or data handling component is represented by an *event actor* (or *actor* for short). An *event* can be considered essentially as "any happening of interest that can be observed from within a computer" [8] (or a software system). An event might contain some attributes such as its unique identification, timing, data references, and so on [8]. DERA uses the notion of *event types* to represent a class of events that share a common set of attributes. To encapsulate a logical group of related actors (for instance, actors that perform the functionality of a certain department or organization), DERA provides the concept of *execution domains*. Two execution domains can be connected via a special kind of actor, namely, *event bridge*, which receives and forwards events from one domain to the other [16]. Well-defined actor interfaces support us in analyzing and performing runtime changes in event-based systems, such as substituting an event actor by another with a compatible port or changing the execution order of event actors by substituting an actor with another [16].

In this paper, we leverage trace semantics [3] for describing the observed behavior of event-based systems and the semantics of the proposed change patterns. One of the major advantages of trace semantics, which is very suitable for our approach, is that the underlying system can be treated as a black box and its behavior is described in terms of the states and actions that we observe from outside.

3 Approach

The implementation of a particular change in an event-based system involves defining the relevant actions (e.g., adding or removing components, enabling or disabling components, altering the components' inputs or outputs, or adjusting the execution order of components) and carrying out these actions while taking into account the consequences (as other components might be affected by these actions). That is, in order to enact a change in an event-based system, the software engineers have to deal with many technical details at different levels of abstraction, which is very tedious and error-prone.

Weber et al. identify a set of change patterns that recur in many of existing software systems [17]. These patterns are specific for process-aware information systems (PAIS) where the execution of the software system is bound to a process schema. Changes mostly can not be done during runtime [1, 13, 15]. These patterns are specific for process-aware information systems (PAIS) where the execution of the software system is bound to a process schema, a prescribed rigid description of the behavior flow, and therefore, mostly can not be changed during runtime or just slightly deviated from the initial schema [1, 13, 15]. As a result, these approaches are not readily applicable for event-based systems where components are highly decoupled and the dependencies between components are subject to change at any time, even during the execution of the systems. Nevertheless, the aforementioned patterns provide a basis for describing changes of the behavior in any information systems.

In our work, we investigate and adapt these patterns in the context of event-based software systems that are different to PAISs because there are no prescribed execution descriptions and the constituent elements of a system and their relationships can be arbitrarily changed at any time. In order to deal with the complexity and the large degree of flexibility of event-based systems, we aim at supporting system evolution at different levels of abstraction. We introduce low-level primitives for encapsulating the basic change actions, such as adding or removing an event or an actor, replacing an event or actor, and so forth.

The definitions of these primitives are given in Table 1. They describe simple, primitive low-level actions for populating and modifying event-based systems that conform to the definitions we provided in Section 2. Based on these primitives, in the following we present change patterns for event-based systems with which the software engineers can easier describe and apply desired changes at a higher level of abstraction.

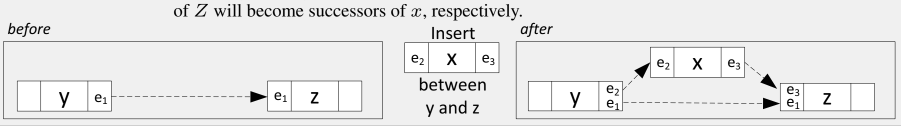
On top of the aforementioned primitives, we introduce change patterns for event-based systems. These patterns extend the change patterns that are frequently occurring and supported in most of today's information systems according to the survey of Weber et al. [17]. In this section, we present the change patterns along with their formal descriptions and abbreviated proofs for correctness. We describe these pattern based on the widely accepted intention of the developers as observed and documented in [17] and discuss potential variants and extensions of these pattern.

Due to space limitations, we opt to present the **INSERT** in detail. We also realized a set of other patterns in the same way as **INSERT**. An overview of our patterns as well as their evaluation can be found in Table 2.

Pattern INSERT. As an event-based system evolves, additional functionality is often added. We use a simple but realistic example of an online shopping system to illustrate the situation. In this system, to complete an order, the customer can use two addresses: one for billing and one for shipping. Let us consider the following scenario: In the first system deployment, the shipping address is also used for the billing address. During evolution of the system, a new component for adding an additional billing address should get inserted. Normally, in event-based systems the developers would have to deal with every technical details of implementing new components and exchanging events among the components. The **INSERT** pattern aims at encapsulating and hiding such details to help the developers to focus on defining the behaviors of the new components and specifying the desired inputs and outputs of the actors. The example visualization of the pattern below shows the change pattern to be applied in the middle. On the left side, the event-based system before the change is depicted, and on the right side it is depicted after the change.

Pattern: INSERT

Description INSERT(x, Y, Z) will add an actor x such that all actors of Y will become predecessors and those of Z will become successors of x, respectively.



An event-based system $\mathcal{S}$ is represented in DERA by a 2-tuple $(\mathcal{A}, \mathcal{E})$, where $\mathcal{A}$ is the set of event actors and $\mathcal{E}$ is the set of event types exchanged by these actors. Let $\bullet x$ (resp. $x\bullet$) be the set of input (resp. output) events of an actor x. The **INSERT** pattern (represented by the function p) that transforms an execution domain $\mathcal{S}(\mathcal{A}, \mathcal{E})$ into $\mathcal{S}'(\mathcal{A}', \mathcal{E}')$, i.e., $p : \mathcal{S} \xrightarrow{\text{INSERT}(x,Y,Z)} \mathcal{S}'$, can be defined as follows.

$$\begin{aligned}
&\mathcal{A}' = p(\mathcal{A}) = \mathcal{A} \cup x \\
&\mathcal{E}' = p(\mathcal{E}) = \mathcal{E} \cup x\bullet \cup \bullet x \\
&Y' = p(Y) : \forall y \in Y : y'\bullet = y\bullet \cup \bullet x, \text{where } y' = p(y) \\
&Z' = p(Z) : \forall z \in Y : \bullet z' = \bullet z \cup x\bullet, \text{where } z' = p(z)
\end{aligned} \tag{1}$$

The developers may want to use a variant of the **INSERT** pattern in which the transitions from the actors of Y to those of Z will be strictly redirected through x, i.e., $y' \rightarrow x' \rightarrow z'$. A formal description of the variant can be adapted from Equation (1) as:

$$\begin{aligned}
&\bullet y' = \bullet y \setminus \{e | e \in x\bullet \cap \bullet y \ \wedge \ e \notin a\bullet, \forall a \in \mathcal{A}\}, \text{where } y' = p(y) \\
&y'\bullet = y\bullet \setminus \{e | e \in y\bullet \cap \bullet z \ \wedge \ e \notin a\bullet, \forall a \in \mathcal{A}\}, \text{where } y' = p(y) \\
&\bullet x' = \bullet x \ \cup y'\bullet, \text{where } x' = p(x), y' = p(y) \\
&\bullet z' = x\bullet \ \cup \bullet z' \setminus \{e | e \in y\bullet \cap \bullet z \ \wedge \ e \notin a\bullet, \forall a \in \mathcal{A}\} \text{where } z' = p(z)
\end{aligned} \tag{2}$$

The specification of $\bullet y'$ in Equation (2) is to adjust any transition $x \rightarrow y$ that exists before changing. We alter the output of y', i.e., $y'\bullet$, and the inputs of x' and z', i.e., $\bullet x'$ and $\bullet z'$, respectively, so direct transitions $y \rightarrow z$ will be transformed to $y' \rightarrow x' \rightarrow z'$.

We devise the post-conditions for the basic case of the **INSERT** pattern according to the Equation (1) and assert that the changed system must satisfy these conditions. The conditions for the extended cases and their proofs can be achieved in the same manner.

Lemma 1. *The new state $\mathcal{S}'$ of the execution domain $\mathcal{S}$ achieved by applying the* **INSERT** *pattern, i.e., $\mathcal{S} \xrightarrow{\text{INSERT}(x,Y,Z)} \mathcal{S}'$, satisfies:*

$$\forall t \in \mathcal{T}_{S'}, \forall y \in Y : y \in t \Rightarrow y \prec x \tag{3}$$

$$\forall t \in \mathcal{T}_{S'}, \forall z \in Z : x \in t \Rightarrow x \prec z \tag{4}$$

We sketch a simple proof for Equation (3), which can be applied for Equation (4).

Proof. Let $\mathcal{S}'$ be the result of the application of the **INSERT**(x, Y, Z) pattern on $\mathcal{S}$. When an actor $y \in Y$ finishes its execution, y will emit all of its output events according to the prerequisite **R3** including the events that x is awaiting with respect to Equation 1. As a result, x will be triggered next due to **R2**. Thus, $y \prec x$. □

4 Evaluation

In the scope of our work, a proof-of-concept implementation of the primitive actions and change patterns has been developed and incorporated into the DERA framework [16]. In order to illustrate the increase of productivity using our approach, we estimate the necessary effort for manually implementing a change on an event-based system and compare these results to our change patterns. To quantify the required efforts for a change, we count the number of statements used for implementing a change. This is analogous to Line of Code metrics [5].

In Table 1, the number of statements encapsulated in the primitive change actions are presented. The first column contains a letter representing the corresponding primitive action shown in the second column. The third column, namely, E_s, depicts the number of statements needed to express the primitives. We note that only the major statements for the implementation are counted while the declarations (e.g. of packages, class body, methods or variables), comments, logging or failure handling are ignored. Also, the number of statements reflect the most simple case, handling only one predecessor and successor.

Table 1. Change Primitives for DERA-based systems

	Change Primitives	**Description**	E_s
INS	`add(Actor a)`	Add the actor a to the execution domain	9
DEL	`remove(Actor a)`	Remove the actor a from the execution domain	8
TAR	`setTarget(Actor a, ExecutionDomain d)`	Set the execution domain d for an actor a	3
PRT	`set(Actor a, Port p)`	Set a new port p for the actor a	8
DOM	`setDomain(Actor a, ExecutionDomain d)`	Set the execution domain d for actor a	2
ADD	`add(Port p, Event[] events)`	Add a set of $events$ to port p	14
REM	`remove(Port p, Event[] events)`	Remove a set of $events$ from port p	12
REPALL	`replace(Port p, Event[] events)`	Replace all events of port p with another set of $events$	7
REP	`replace(Port p, Event e1, Event e2)`	Replace event $e1$ of port p with event $e2$	11

Table 2. Change Patterns and Effort Reduction

Change Pattern	**Description**	**Primitive**	E_p	E_s	$ER\,(\%)$
INSERT	will add an actor x such that all actors of Y will become predecessors and those of Z will become successors of x, respectively	INS, 2*ADD	3	37	8.11
DELETE	will remove the actor x from the current execution domain $\mathcal{S}$	DEL	1	8	12.50
MOVE (x, y, z)	will move the actor x in a way that the actor y will become predecessor and the actor z will become successor of x, respectively	2*ADD, REPALL	3	35	8.57
REPLACE(x, y)	will substitute the actor x by the actor y	INS, DEL, 2*ADD	4	45	8.89
SWAP(x, y)	Given an actor x that precedes an actor y, this pattern will switch the execution order between x and y	4*REPALL	4	28	14.29
PARALLELIZE(x, y)	enables the concurrent execution of two actors x and y that are performed sequentially before	4*REPALL	4	28	14.29
MIGRATE$(x, \mathcal{S}_1, \mathcal{S}_2)$	will migrate an actor x from an execution domain $\mathcal{S}_1$ to another execution domain $\mathcal{S}_2$	INS, DEL, DOM, 4*ADD	7	75	9.33
Average					10.89

The result of the effort comparison between the number of statements for change primitives and the change patterns are shown in Table 2. The first column shows the name of the change pattern. The second column lists the used change primitives. The third column shows the number of change primitives E_p used by a change pattern. The third column shows the number of Java statements E_s used to implement these change primitives. The last column shows the effort ratio ER between the change primitives and the sum of Java statements, where $ER = E_p/E_s$. The ratio shows that describing changes using our change patterns is about 11 percent of the effort compared to the code needed to implement each change manually. Using the change patterns, there are roughly 9 times (i.e., $1/11\%$) less statements needed to be written in comparison to code each change individually.

5 Related Work

Weber et al. [14, 17] identified a large set of change patterns that are frequently occurring in and supported by the most of today's process-aware information systems, where a process is described by a number of activities and a control flow is defining their execution sequence. Since the process structure is defined at design time, changing it at runtime is very difficult. Several approaches try to relax the rigid structures of process descriptions to enable a certain degree of flexibility of process execution [7, 11, 12]. Event-based systems, like DERA, provide a high flexibility for runtime changes, since only virtual relationships among actors exist. The change patterns observed by Weber et al. are designed to target PAIS in which the execution order of the elements are prescribed at design time and not changed or slightly deviated from the prescribed descriptions at runtime. Therefore, they are readily applicable for event-based systems where components are highly decoupled from each other.

6 Conclusion

Supporting the evolution of event-based systems is challenging because software engineers have to deal not only with the complexity but also a large degree of flexibility of these systems. We address this challenge in this paper by introducing novel concepts and techniques for aiding the software engineers in better analyzing and implementing particular changes on an event-based system. At the low level of abstraction, our approach provides primitive change actions that encapsulates several programming language-level and/or event exchange-level statements. Although these primitives can be used by technical experts who are able to handle technical details, it is still tedious and error prone to use them directly. We propose, at a higher level of abstraction, change patterns for event-based systems that are frequently supported and used in several information systems nowadays along with their formal descriptions. If the change patterns can be used, the effort required for a change can be significantly reduced as shown in our evaluation. A limitation of our approach is that the change patterns only perform all required changes, if the change requirements do not deviate from the specified patterns. In such cases, the changes must be manually reviewed and maybe additional changes using the primitive actions are needed. We plan to further automate changes in our future work, e.g. by supporting parameterizable variants of the change patterns and suggesting useful additional changes through tool support.

Acknowledgement. This work was partially supported by the European Union FP7 project INDENICA, Grant No. 257483 and the WWTF project CONTAINER, Grant No. ICT12-001.

References

[1] van der Aalst, W.M.P., Pesic, M., Schonenberg, H.: Declarative workflows: Balancing between flexibility and support. Computer Science - Research and Development 23(2), 99–113 (2009)

[2] Arbab, F., Talcott, C. (eds.): COORDINATION 2002. LNCS, vol. 2315. Springer, Heidelberg (2002)

[3] Broy, M., Olderog, E.R.: Trace-Oriented Models of Concurrency. In: Bergstra, J., Ponse, A., Scott, S. (eds.) Handbook of Process Algebra, pp. 101–195. Elsevier Science B.V. (2001)

[4] Carzaniga, A., Rosenblum, D.S., Wolf, A.L.: Design and evaluation of a wide-area event notification service. ACM Trans. Comput. Syst. 19(3), 332–383 (2001)

[5] Fenton, N.E., Pfleeger, S.L.: Software Metrics: A Rigorous and Practical Approach, 2nd edn. PWS (1998)

[6] Fiege, L., Mühl, G., Gärtner, F.C.: Modular event-based systems. The Knowledge Engineering Review 17(4), 359–388 (2002)

[7] Hallerbach, A., Bauer, T., Reichert, M.: Capturing variability in business process models: the provop approach. J. Softw. Maint. Evol. 22, 519–546 (2010)

[8] Mühl, G., Fiege, L., Pietzuch, P.: Distributed Event-Based Systems. Springer (2006)

[9] Overbeek, S., Janssen, M., Bommel, P.: Designing, formalizing, and evaluating a flexible architecture for integrated service delivery: combining event-driven and service-oriented architectures. Service Oriented Computing and Applications 6, 167–188 (2012)

[10] Paton, N.W., Díaz, O.: Active database systems. ACM Comput. Surv. 31(1), 63–103 (1999)

[11] Redding, G., Dumas, M., ter Hofstede, A., Iordachescu, A.: Modelling flexible processes with business objects. In: IEEE Conf. on Commerce and Enterprise Computing (CEC), pp. 41–48 (2009)

[12] Reichert, M., Dadam, P.: Enabling adaptive process-aware information systems with ADEPT2. In: Handbook of Research on Business Process Modeling, pp. 173–203. Information Science Reference (2009)

[13] Reichert, M., Weber, B.: Enabling Flexibility in Process-Aware Information Systems - Challenges, Methods, Technologies. Springer (2012)

[14] Rinderle-Ma, S., Reichert, M., Weber, B.: On the formal semantics of change patterns in process-aware information systems. In: Li, Q., Spaccapietra, S., Yu, E., Olivé, A. (eds.) ER 2008. LNCS, vol. 5231, pp. 279–293. Springer, Heidelberg (2008)

[15] Schonenberg, H., Mans, R., Russell, N.: Process flexibility: A survey of contemporary approaches. In: Dietz, L.G., Albani, A., Barjis, J. (eds.) CIAO! 2008 and EOMAS 2008. LNBIP, vol. 10, pp. 16–30. Springer, Heidelberg (2008)

[16] Tran, H., Zdun, U.: Event-driven actors for supporting flexibility and scalability in service-based integration architecture. In: Meersman, R., et al. (eds.) OTM 2012, Part I. LNCS, vol. 7565, pp. 164–181. Springer, Heidelberg (2012)

[17] Weber, B., Rinderle, S., Reichert, M.: Change patterns and change support features in process-aware information systems. In: Krogstie, J., Opdahl, A.L., Sindre, G. (eds.) CAiSE 2007. LNCS, vol. 4495, pp. 574–588. Springer, Heidelberg (2007)

Subject-Oriented Semantic Knowledge Warehouse (SSKW) to Support Cognitive DSS

Tasneem Memon, Jie Lu, Farookh Khadeer Hussain, and Rajan Rauniyar

Decision Systems & e-Service Intelligence Laboratory (DeSI),
Centre for Quantum Computation & Intelligent Systems (QCIS)
University of Technology Sydney, Australia
{tasneem.memon,jie.lu,farookh.hussain}@uts.edu.au,
rajanrauniyar@gmail.com

Abstract. The communication between cognitive DSS and data warehouse tends to be inefficient due to their contradictory knowledge/data oriented nature. Data-to-knowledge conversion requires specialized techniques, whereas knowledge-to-data conversion results in *loss of knowledge*. To address these issues, a *subject-oriented semantic knowledge warehouse (SSKW)* is proposed, to provide *relevant* and *precise knowledge* to CDSS. The SSKW consists of: a) *object/process/event/relationship (OPER)* model to store domain knowledge in a unified fashion; and, b) a *subjective view* database, containing opinions of stakeholders about various OPER knowledge elements. A case study to compare the performance of the SSKW-based CDSS against a DW-based CDSS is presented. The results show that SSKW improves communication efficiency, provides *relevant* and *precise domain knowledge* to CDSS in less decision cycles, minimizes the *loss of knowledge*, and helps decision maker to quickly grasp the decision situation through its *human-centric* nature.

Keywords: Knowledge warehousing, knowledge representation, cognitive decision support, business intelligence.

1 Introduction

Despite *knowledge* being the key factor to gain competitive advantage, BI technologies, especially data warehouses (DW), do not support handling of *knowledge* [1–4]. DW filter out a great deal of *knowledge* during the *loading phase*, resulting in the loss of crucial *decision making knowledge* [5]. The field of CDSS, on the other hand, has made significant progress regarding *knowledge* processing; yet CDSS has to rely upon DW for back-end [6–9]. This poses two issues: 1) huge amounts of *knowledge* is lost during knowledge-to-data (CDSS-to-DW) conversion [2]. 2) *data mining* is essential to infer hidden patterns from data to create *knowledge* during data-to-knowledge (DW-to-CDSS) conversion. A knowledge-enabled back-end was thus deemed essential to support decision making, and *knowledge warehouses (KW)* were proposed [2, 4, 6]. KW are hybrid knowledge-base(KB) systems (collection of knowledge-bases) [2]. The main disadvantage of hybrid-KB approach is knowledge integration overhead, which can decrease the performance with growing KBs [10]. It is argued here that a KW should store knowledge in a unified format. A *subject-oriented semantic knowledge warehouse (SSKW)*, is proposed to store *knowledge* in *unified, subject-oriented* and *semantic* manner. SSKW addresses the issues of *inefficient communication*, and the *loss of knowledge*. The paper is organized as follows. Section 2 presents literature review, followed by SSKW schema in Section 3. Section 4 describes the SSKW architecture and implementation. The case study is given in Section 5, followed by comparison of SSKW and DW, and conclusion.

2 Literature Review

DW provide decison makers with huge amounts of structured data, but fall short in handling *knowledge*, thus only partially supporting decision making process [3,7,11]. To fill this gap, *knowledge warehouses (KW)* were proposed [2,4], based on Nonaka's [1] *knowledge spiral model*. The term has been used for various applications [3]; however, Nemati et al [2] have defined it as a *novel technology* to collect, organize, store and disseminate *knowledge*. This paper is based on this characterization of KW. The KW by Nemati et al [2] is a collection of knowledge-bases (hybrid knowledge-base). Dymond [4] also proposed a similar hybrid-KB based KW. Several such KW have been designed [12–15]. Among these, ISYMOD [13] is the closest to the SSKW storage structure, while Pedersen [12] proposes a powerful tool for *evolution of knowledge*, i.e. *subjective opinions*, demonstrating the importance of individual's opinions.

The relational technology falls short in representing the complex and unstructured nature of *knowledge* [16,17]. The fact can be observed from a KW based on relational technology, by Levy et al [18,19]. The limitations of relational model hinder real-world object/relationship representation in this KW. Also, OLAP reports produced are lengthy and time-consuming [19]. Object-oriented approach on the other hand, has proven to be effective in representing real world entities in intuitive format, providing basis for rich semantics [20–22]. We use *object-oriented model* as the foundation to employ *semantics* and *subject-oriented approach* in the design of *SSKW*.

3 Subject-Oriented Semantic Knowledge Warehouse (SSKW) Schema

This section describes SSKW, and its components, *OPER model* and *SV database*.

3.1 Object/Process/Event/Relationship (OPER) Model

To define *standard knowledge* format, we categorized *knowledge* at syntactic level to create "a virtual level of knowledge representation where the dominant traits are not domain-dependent" [23]. After conducting a study, three fundamental *knowledge types* were identified in a typical business environment: *objects, processes* and *events*. However, to simulate real-world, and the reasoning process of human mind, these elements must be linked in a meaningful way [24, 25]. Thus, *relationships* are added to them.

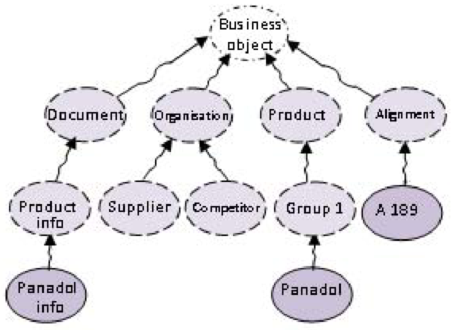


Fig. 1. Example of Object Hierarchy

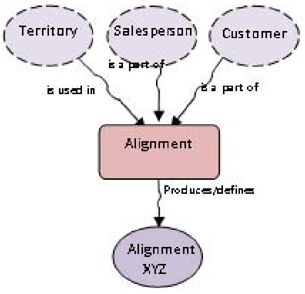


Fig. 2. Example of a Process

Object. *Object* defines an entity, such as *person, organisation* or *document*. The object hierarchy has three design levels (Fig. 1): 1) the abstract level, such as *business object*; 2) the sub-type level, such as *organisation*, which can span over multiple internal levels; and 3) the uniquely-identifiable level, such as *A189*. SSKW objects can have three types of properties; 1) *single-valued* such as name and age; 2) *objects*, such as organization; and 3) *processes* to define object's behavior, such as *decision making*.

Process. It is a skeleton for business procedures, such as *delivery, sample checking* and *marketing*. It may involve one or more *objects*, in two categories: 1) Input objects, which include *Objects working* and *Objects being worked on*; and 2) Output objects (objects created) (Fig.2)

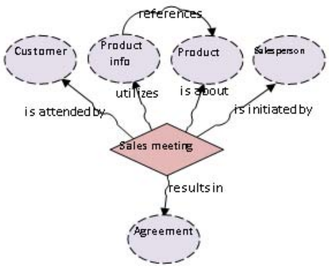
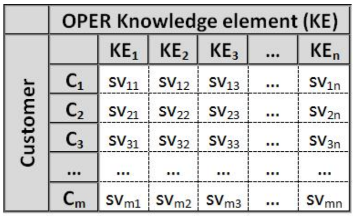


Fig. 3. Example of Event

Event. An *event* represents an occurrence, such as *meeting, cold call* and *presentation*. Event is a spatio-temporal unit, having *time* and *location* as its defining factors. One or more *objects/processes* may be involved. An event does not have an output 3.

Relationship. *Relationship* defines the nature of association between two *objects* (Fig. 1,2,3). The SSKW relationships are different than those in relational database where a relationship is applied to all the instances of an entity. That is, *instances* of the same type must have the same set of relationships. In SSKW however, *objects* of the same type may hold different relationships. SSKW *relationships* may be cause-effect, implication, subtype, similar-to, instance and reverse [26].

3.2 Subjective Views



		OPER Knowledge element (KE)				
		KE_1	KE_2	KE_3	...	KE_n
Customer	C_1	SV_{11}	SV_{12}	SV_{13}	...	SV_{1n}
	C_2	SV_{21}	SV_{22}	SV_{23}	...	SV_{2n}
	C_3	SV_{31}	SV_{32}	SV_{33}	...	SV_{3n}
	...	...	...	...	...	...
	C_m	SV_{m1}	SV_{m2}	SV_{m3}	...	SV_{mn}

Fig. 4. Customers' Subjective View

Based on *subject-oriented programming (SoP)* [27], *Subjective views (SV)* contain *opinions* and *(attitudes)* of stakeholders, such as customers, employees or competitors, about OPER knowledge elements, to assist decision makers in perceiving the situation from the perspective of stakeholders. A SV is a matrix, with OPER knowledge elements for one dimension, and the stakeholder-type (such as customers or managers) for the other. Fig.4 demonstrates a *customer SV*, where *rows* represent *customers*, and *columns* represent *OPER knowledge elements* such as *product*. Each element in the matrix contains customer's feedback about a specific knowledge element. This feedback can be a number (from likert scale), or text-based (from blogs). A SV matrix is used to calculate two values; a) the current trend for an OPER knowledge element, by aggregating the feedback of all the stakeholders; and b) the potential future preferences of a stakeholder, by studying their feedback about various OPER knowledge elements.

4 Subject-Oriented Semantic Knowledge Warehouse (SSKW) architecture and Implementation

This section gives an overview of SSKW conceptual model and implementation.

4.1 SSKW Architecture

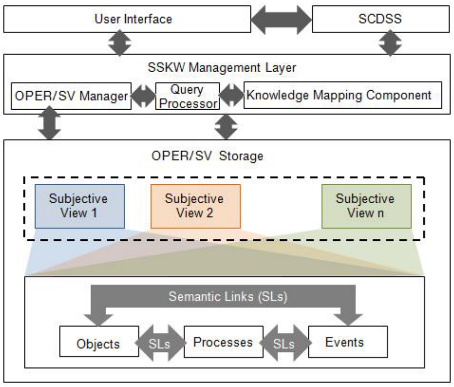


Fig. 5. The conceptual model of SSKW

The architecture consists of three main components; *OPER Storage, the SVs* and *Knowledge Management (KM) layer.* As discussed previously, OPER storage contains *objects, processes, events* and *semantic relationships* between them; while SVs contain the opinions of stakeholders about OPER knowledge elements. SSKW may contain more than one SVs (Fig 5). The *SSKW management layer* consists of the *OPER/SV manager* to interact with the storage, *query processor* to manage the queries received from GUI, and *knowledge mapping component (KMC)* to maps OPER knowledge elements with their counterparts in CDSS. Following are the formal definitions of OPER and SV components.

Let K be the set of all knowledge elements (objects (O), processes(P), events(E) and relationships(R)) in a particular domain, i.e.

$$K = \{k_1, k_2, \ldots, k_n \| k \in O, P, EorR\} \tag{1}$$

Where,O is the set of all the *objects* in the domain, P is the set of all *processes*, and E is the set of all *events*. Whereas *relationship* is defined as,

$$R = (m_{pre}, SL, m_{post}) \tag{2}$$

Here, m_{pre} and m_{post} are the knowledge elements of types O, P or E. SL is the set of all *semantic links* in the domain.

The complete set of *subjective views* is denoted by SV, and is defined as the set of matrices, in which each matrix represents *subjective views* of a particular group of stakeholders, about OPER knowledge elements. A SV is defined as follows:

$$SV = \{[sv_1], [sv_2], \ldots, [sv_l]\} \tag{3}$$

Where, each [sv] is a two dimensional matrix, as defined below:

$$\texttt{[sv]} \qquad \begin{array}{c} \\ P_1 \\ P_2 \\ \ldots \\ P_m \end{array} \begin{array}{c} \begin{array}{cccc} E_1 & E_2 & \ldots & E_n \end{array} \\ \begin{bmatrix} sv_{11} & sv_{12} & \ldots & sv_{1n} \\ sv_{21} & sv_{22} & \ldots & sv_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ sv_{m1} & sv_{m2} & \ldots & sv_{mn} \end{bmatrix} \end{array}$$

where [sv] represents the sets of *views* of *participants (P)* from a particular group of stakeholders, such as *customer group* or *executive group*.

Each *entry (element)* of this matrix is mapped to exactly one knowledge element (E) in the set K of equation (1).

4.2 Implementation of SSKW

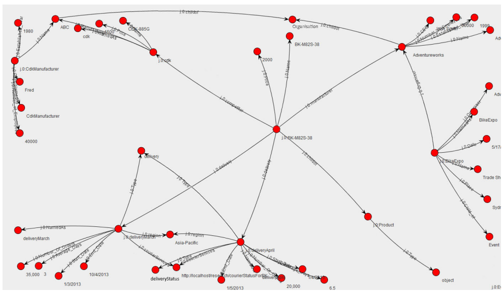


Fig. 6. Example of Knowledge storage in OPER

The SSKW architecture is implemented to test and evaluate its practicality and performance. The SVs use relational database, while OPER model is implemented using RDF. The main advantage of OPER storage is that unlike DW, in SSKW *knowledge* can be accessed through any dimension. As shown in Fig. 6, *BK-M82S-38* is type *product*, linked with its *manufacturer AdventureWorks*,*competitor CDK-bike*, and two *delivery* processes, *deliveryMarch* and *deliveryApril* of type *delivery*. When *BK-M82S-38* is accessed, all their associated knowledge elements become available through *semantic relationships*. Similarly, accessing *delivery* provides direct access to all its child processes for all the products.

5 Case Study

We test our system against a DW-based CDSS, FACETS by Niu et al [7]. The decision situation is: *"Why the sales of bike (BK-M82S-38) have dropped over the past 2 weeks?"* FACETS solved this problem in four decision cycles. The output of the system is in the form of lengthy reports [7, p.183].

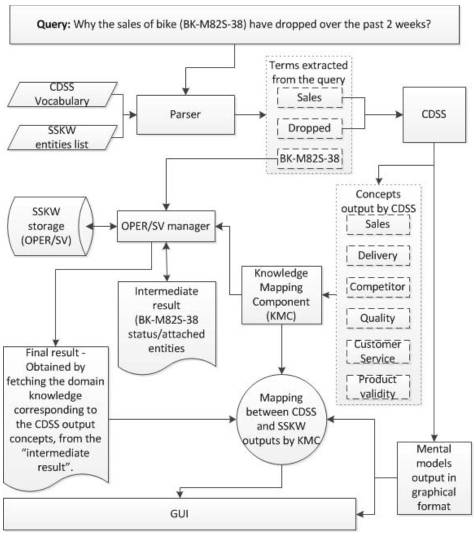

Query: Why the sales of bike (BK-M82S-38) have dropped over the past 2 weeks?
CDSS Vocabulary
SSKW entities list
Parser
Terms extracted from the query
Sales
Dropped
BK-M82S-38
CDSS
Concepts output by CDSS
Sales
Delivery
Competitor
Quality
Customer Service
Product validity
SSKW storage (OPER/SV)
OPER/SV manager
Intermediate result (BK-M82S-38 status/attached entities
Knowledge Mapping Component (KMC)
Final result - Obtained by fetching the domain knowledge corresponding to the CDSS output concepts, from the "intermediate result".
Mapping between CDSS and SSKW outputs by KMC
Mental models output in graphical format
GUI

Fig. 7. The decision process of SSKW-based CDSS

SSKW-based CDSS solves this problem as follows (Fig. 7). The system accepts above-mentioned query in natural language, and extracts main terms from it with the help of *CDSS* and *SSKW vocabularies*. The terms found in CDSS vocabulary, i.e. *sales*, verb *dropped*, are sent to CDSS; while the *SSKW* terms, i.e. *BK-M82S-38* are forwarded to *OPER/SV manager*. *OPER/SV manager* fetches the status of *BK-M82S-38* and related processes, events, relationships, and SVs, and keeps them as intermediate result. Meanwhile, CDSS extracts all the mental models containing *sales*, with *dropped* as *relationship* (Fig. 8). The CDSS then forwards the *list of concepts* from the resultant mental models to the *knowledge mapping component (KMC)*, which passes them to OPER/SV manager while retaining the unique identifiers of these *concepts*. The OPER/SV manager searches for these *concepts* in the intermediate result, and sends the result to KMC. KMC maps SSKW and CDSS output through the unique identifiers, and passes the output to GUI. Thus, when a concept, such as *bad delivery*, is clicked in CDSS output (Fig. 8), the corresponding *BK-M82S-38 delivery* report from SSKW is displayed (Fig. 9).

6 Comparison between OPER and DW

In DW, data is accessed through the dimensions previously declared at design phase, such as *marketing, sales,* and *finance.* In OPER model however, the business concepts and entities exist individually (but *semantically* associated),

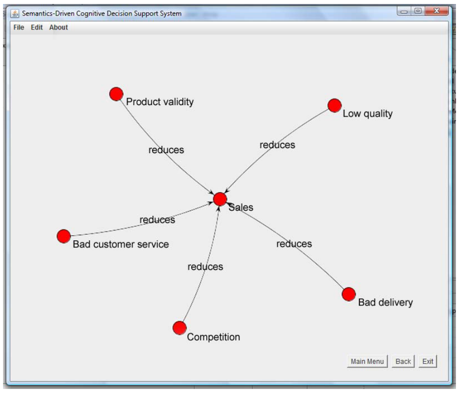
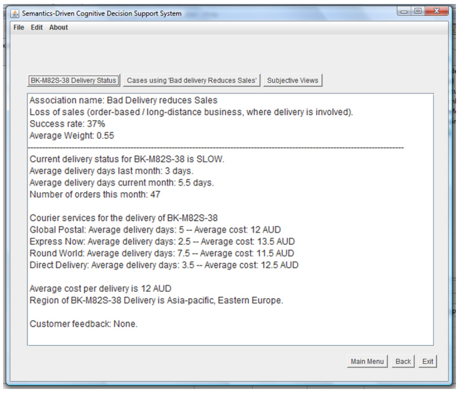


Fig. 8. Output of user query from CDSS


Semantics-Driven Cognitive Decision Support System
File Edit About
BK-M82S-38 Delivery Status
Cases using 'Bad delivery Reduces Sales'
Subjective Views
Main Menu
Back
Exit

Association name: Bad Delivery reduces Sales
Loss of sales (order-based / long-distance business, where delivery is involved).
Success rate: 37%
Average Weight: 0.55

Current delivery status for BK-M82S-38 is SLOW.
Average delivery days last month: 3 days.
Average delivery days current month: 5.5 days.
Number of orders this month: 47

Courier services for the delivery of BK-M82S-38
Global Postal: Average delivery days: 5 -- Average cost: 12 AUD
Express Now: Average delivery days: 2.5 -- Average cost: 13.5 AUD
Round World: Average delivery days: 7.5 -- Average cost: 11.5 AUD
Direct Delivery: Average delivery days: 3.5 -- Average cost: 12.5 AUD

Average cost per delivery is 12 AUD
Region of BK-M82S-38 Delivery is Asia-pacific, Eastern Europe.

Customer feedback: None.

Fig. 9. The delivery report from SSKW

defined as *objects, processes* or *events*. Every *object, process* and *event*, therefore, becomes the dimension of reference, through which knowledge can be searched and accessed.

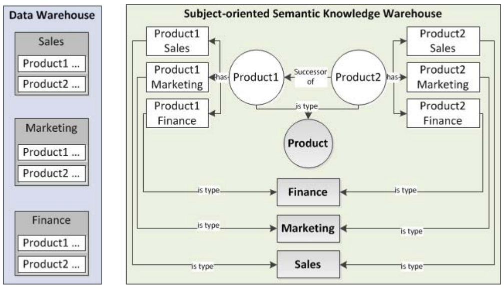


Fig. 10. The difference between the DW and OPER storage

This can be very powerful tool for decision maker, as the domain knowledge is not divided, but encapsulated through the *object, process* and *event* constructs (Fig.10). For instance, to get the status of sales for all the products stored, the process *sales* is accessed, which leads to all its child processes, *product1Sales* and *product2Sales*; whereas to output the status of *product1*, the object *product1* and all its *processes (product1Sales, product1Marketing, product1Finance* are fetched. This way, OPER enables the access of *relevant* knowledge in efficient and comprehensive manner, improving decision support.

7 Conclusion and Future Work

Considering the significance of knowledge in today's complex business environment, a *subject-oriented semantic knowledge warehouse (SSKW)* has been proposed, which stores knowledge in a unified, meaningful and subject-oriented manner using *subject-oriented* and *semantic* approaches. It facilitates intuitive and efficient communication with CDSS, prevents *loss of knowledge*, improves knowledge precision, and ensures efficient delivery of knowledge.

References

1. Ikujiro, N.: The knowledge-creating company. Harvard Business Review, 96–104 (1991)
2. Nemati, H.R., Steiger, D.M., Iyer, L.S., Herschel, R.T.: Knowledge warehouse: An architectural integration of knowledge management, decision support, artificial intelligence and data warehousing. Decision Support Systems 33(2), 143–161 (2002)
3. Garcia Perez, A., Mitra, A.: Revisiting knowledge warehousing: theoretical foundations. Int. J. Bus. Inf. Syst. 3(6), 572–586 (2008)
4. Dymond, A.: The knowledge warehouse: The next step beyond the data warehouse (2002)
5. Wade, A.E.: Hitting the relational wall (1998)
6. Chen, J.Q., Lee, S.M.: An exploratory cognitive dss for strategic decision making. Decision Support Systems 36(2), 147–160 (2003)
7. Niu, L., Lu, J., Zhang, G.: Cognition-Driven Decision Support for Business Intelligence. SCI, vol. 238. Springer, Heidelberg (2009)
8. Yadav, S.B., Khazanchi, D.: Subjective understanding in strategic decision making: An information systems perspective. Decision Support Systems 8(1), 55–71 (1992)
9. Fu, X., Wei, H.: Cognition inspired object oriented knowledge warehouse architecture. Journal of Software 5(9) (2010)
10. Drias, H., Aouichat, A., Boutorh, A.: Towards incremental knowledge warehousing and mining. In: Omatu, S., Paz Santana, J.F., González, S.R., Molina, J.M., Bernardos, A.M., Rodríguez, J.M.C. (eds.) Distributed Computing and Artificial Intelligence. AISC, vol. 151, pp. 501–510. Springer, Heidelberg (2012)
11. Goutam Kumar, S.: Business intelligence computing issues. Ubiquity 2007, 1 (2007)
12. Pedersen, K.V.: A framework for a clinical reasoning knowledge warehouse. In: IEEE Proceedings of the IDEAS Workshop on Medical Information Systems: The Digital Hospital. IEEE (2004)
13. Chabalier, J., Capponi, C., Quentin, Y., Fichant, G.: Isymod: a knowledge warehouse for the identification, assembly and analysis of bacterial integrated systems. Bioinformatics 21(7), 1246–1256 (2005)
14. Kassel, S., Grebenstein, K., Tittmann, C.: A knowledge-based decision-support-system for e-commerce. In: EUROMEDIA 2005: 11th Annual Euromedia Conference, Eurosis, Ghent (2005)
15. Zhang, H., Liang, Y.: A knowledge warehouse system for enterprise resource planning systems. Systems Research and Behavioral Science 23(2), 169–176 (2006)
16. Sun, X.: Osln: An object-oriented semantic link network language for complex object description and operation. Future Generation Computer Systems 26(3), 389–399 (2009)
17. Wade, A.E.: Hitting the relational wall (April 13, 2010, 2005)
18. Levy, M.: A conceptual model of a knowledge warehouse. In: Enterprise Information Systems Design, Implementation and Management: Organizational Applications, pp. 148–161. IGI Global (2011)
19. Levy, M., Pliskin, N., Ravid, G.: Knowledge warehouse for decision support in critical business processes: Conceptual modeling and requirements elicitation. In: Schuff, D., Paradice, D., Burstein, F., Power, D.J., Sharda, R. (eds.) Decision Support. Annals of Information Systems, vol. 14, pp. 131–148. Springer, New York (2011)
20. Fu, X., Wei, H.: Cognition inspired object oriented knowledge warehouse architecture. Journal of Software 5(9), 926–933 (2010)

21. Mitra, A., Lau, J.: Challenges of developing an interactive knowledge warehouse within the media industry: Significance of emergent frameworks. In: European Conference on Information Systems (ECIS), pp. 96–104 (2004)
22. Firestone, J.M.: Object oriented data warehousing. Executive Information Systems Inc. (1997)
23. de Abreu, P.F.: Knowledge classes or canonical representation (1996)
24. Sheth, A., Arpinar, B., Kashyap, V.: Relationships at the heart of semantic web: Modeling, discovering, and exploiting complex semantic relationships. In: Nikravesh, M., Azvin, B., Yager, R., Zadeh, L.A. (eds.) Enhancing the Power of the Internet. STUDFUZZ, vol. 139, pp. 63–94. Springer, Heidelberg (2004)
25. Bush, V.: As we think. The Atlantic Monthly 176(1), 101–108 (1945)
26. Zhuge, H.: The Knowledge Grid. World Scientific Publishing Co., Inc., River Edge (2004)
27. Harrison, W., Ossher, H.: Subject-oriented programming - (a critique of pure objects). Sigplan Notices 28(10), 411–428 (1993)

Improving Web Service Composition with User Requirement Transformation and Capability Model

Wenbin Li, Youakim Badr, and Frédérique Biennier

INSA-Lyon
Université de Lyon
Villeurbanne, France
{wenbin.li,youakim.badr,frederique.biennier}@insa-lyon.fr

Abstract. In order to discover and compose relevant Web services, most Web service composition approaches require users to describe composition requirements and constraints in formal expressions. However, requirements still focus on the technical level as they require domain-specific knowledge on functional and non-functional properties. As a matter of fact, the gap between users' *high-level requirements* describing business objectives and *composition requirements* remains a challenge in Web service composition. In this paper, we propose an end-to-end transformation approach to build a set of technical composition requirements from user high-level requirements, which are specified by an English-structured language to capture business objectives, and to specify actions that Web services can achieve.

Keywords: Web services, requirement transformation, service capability.

1 Introduction

Web service composition is the process of reusing existing Web services and logically recombining them into composite services in order to provide new functionalities that existing web services cannot provide alone. Composition requirements refer to the constraints imposed on the Web service composition process. Generally, they include various constraints on and among Web services (e.g., control flow constraints, functional and non-functional properties constraints, and dependency constraints, etc.).

However, existing Web service composition approaches [1] focus on how to compose Web services together and highly rely on formal specification of users' composition requirements. Nevertheless, these approaches fail to provide casual users with a natural-like means to express their high-level requirements and to describe their business objectives to be achieved. The gap between user's *high-level requirements* describing business objectives and *composition requirements* specifying Web services details raises a challenge as to how transform high-level requirements in natural-like languages to formalized composition requirements that can be directly used in Web service composition. In order to solve this challenge, we introduce a transformation method between high-level requirements and composition requirements via a Web service capability model. The capability model denotes what an action does in terms of changes in the state of Web services that are involved in compositions [2]. It describes what Web services can do without needing to know all the technical details. The composition requirement transformation method comprises three pillars: a composition requirement meta-model; a three-layer capability model; a transformation process for requirements at different levels.

The remainder of this paper is organized as follows: related works are presented in section 2. In section 3, we introduce the composition requirement meta-model, and then in section 4, we demonstrate how we achieve the automated transformation between different composition requirement models. Section 5 concludes our work and sheds light on future trends.

2 Related Work

Requirement transformation is the process of generation of target requirements from source requirements, according to transformation rules. Requirement transformation methods are generally divided into three categories [3]: transformations based on traceability, transformations based on behavior trees, and transformations based on model transformation. Nevertheless, most of transformation methods are designated to transform formalized requirements. They lack of formalisms to express nature-likes requirements easy to be understood by casual users. The work in [4] examines existing approaches that transform textual requirements into analysis models. As direct automatic transformations of natural language requirements are very difficult due to the inherent ambiguities of natural language, manual or semi-automated transformation are proposed based on user interventions [5][6] or natural requirements defined by given patterns [7][8]. Moreover, Web service discovery and selection either rely on syntactical keywords or semantic relationships based on domain ontology [9]. However, composition requirement does not only apply on functional and non-functional properties of Web services to be composed, but they also concern the composition process (e.g., control flow constraints, dependency constraints, etc.). By discovering Web services prior to the composition, users often are involved in manually specifying constraints on and among Web services. In conclusion, the gap between requirements specified by causal users and technical requirements on Web services persists and requires automated transformation solution to derive from requirements in natural-like languages, describing business objectives, multiple constraints on Web services and their composition process. In addition, some research efforts attempt to provide an abstraction view of Web services by introducing the Web service capability model such as IOPE (e.g., WSMO [10] and OWL-S [11]) or case-frames to model Web services as action-verbs and informational attributes, describing their behaviors [12]. However, we argue that describing service capabilities only based on action-verb and attributes are oversimplified and causing ambiguity without explicit descriptions of the domain by which service capabilities are manipulated.

3 Composition Requirement Meta-model

In order to allow either developers or business people expressing their composition requirements, we model composition requirements in three levels. namely High-level Requirement; Formalized Business Requirements; Web Service Requirements.

3.1 High-Level Requirement (HLR)

High-level requirements refer to constraints that are expressed in a natural-like language to describe user business objectives and desired actions to be performed. In our work, we model user's high-level requirement as a subset of Semantics of Business Vocabulary and Business Rules (SBVR) [13]. SBVR is a formal business rule language with a natural language interface, which implies that either casual users or domain experts can express their requirements with a natural-like language. SBVR provides **facts** to define *noun concepts* (i.e., simply relations among noun concepts) and write **rules** with *modal operators*, *quantifiers*, *qualifiers* and *facts*.

As depicted in Figure 1, we model the high-level requirements **(HLR)** as a set of functional requirements (**FR**), non-functional requirements (**NR**) and a **Context** such as: $HLR = FR \cup NR \cup Context$

1. **FR** consists of two parts: a) **Objective** descriptions based on SBVR facts to define a list of business objectives to be achieved; b) **Information** descriptions based on SBVR rules to describe a list of actions to be performed, where :

$$FR.Objective = \{obj_1, obj_2, \ldots, obj_n\},\ FR.Information = \{i_1, i_2, \ldots, i_n\}$$

2. **NR** is defined based on SBVR rules to specify a list of constraints on actions to be performed, where $NR = \{nf_1, nf_2, \ldots, nf_n\}$

3. **Context** is defined based on SBVR facts, describing generic domain states (e.g., date, location, relation between concepts, etc.), where $Context = \{ctt_1, ctt_2, \ldots, ctt_n\}$

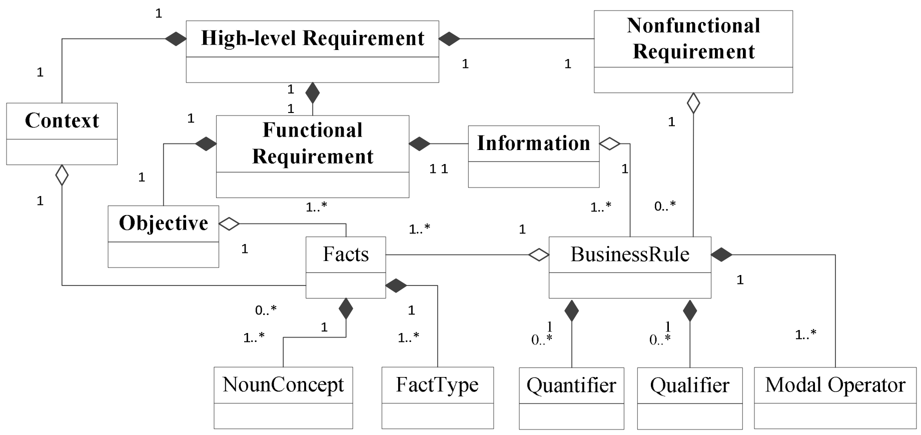


Fig. 1. High-level Requirement Model

In order to illustrate our composition requirement models and its transformation method as described later, we introduce a crisis scenario of two trains' crash in Paris with a list of negative facts describes damages, such as caused fire, interrupted electricity supply, etc. We assume that different actions, managing the crisis, are regarded as Web services while the whole crisis response process is regarded as the execution of composite Web services. Table 1 illustrates High-level Requirements for the train crisis scenario.

Table 1. High-level Requirement Example

FR.Objective	**obj$_1$**: *Manage* train crisis
FR.Information	**i$_1$:** Fire should *be extinguished.* **i$_2$:** Victims should *be assisted.* **I$_3$:** Electricity should *be recovered.*
NF	**nf$_1$:** It is obligatory that at least 10 firemen *extinguish* fire. **nf$_3$:** It is obligatory that the electricity *is recovered after* the fire *is extinguished.*
Context	ctt$_1$: Crisis place *is* Pairs. ctt$_2$: Crisis date *is* 2013/03/01.

3.2 Formalized Business Requirement (FBR)

Formalized Business Requirements describe users' business objectives and desired actions to be performed. In our work, we model formalized business requirements based on the Web service capability model, by which users describe requirements in terms of web service capability profiles. We also propose a three-layer capability model, describing Web service capabilities from different perspectives; request, offer, and composition perspectives. The three-layer capability model consists of: 1) the *objective layer* (**OL**) which describes the business objectives that the capabilities can achieve; 2) the *profile layer* (**PL**) which describes capabilities as action verb and noun pairs, and attributes, representing the real abilities that Web services perform; 3) the *composition layer* (**CL**) which creates relationships among capabilities. A Web service capability profile, denoted by **cap**, is defined as a tuple: *cap = <id, OL, PL, CL>*, where *id* is the identifier of this capability file. The capability files are mainly created by service providers or domain experts, and are associated with Web services. One capability file can be associated with one or more Web services. Fig. 2 shows a detailed UML class diagram of the capability model.

1. The Objective Layer

The objective layer consists of one or more objectives (**obj**) described by a pair of an *ActionVerb* and a *Noun* [12]. We associate action-verbs and nouns to domain specific ontologies that establish shared agreements on their semantics. The ontology is used to semantically match different concepts.

$OL = \{obj_1, obj_2, \ldots, obj_n\}$ where $obj_i = <action\_verb_i, noun_i>$ and $obj_i \in OL$

It is worth noting that an objective can be associated with different capabilities while a capability can have different objectives.

2. The Profile Layer

The capability profile layer is defined in terms of a capability *name* (**cname**) and a list of *attributes* (**attr**), describing the real ability and characteristics that a Web service performs. In a similar way to the objective layer, the capability name is defined as a pair of an ActionVerb and a Noun. However, the capability name describes the real ability that a Web service can perform whereas the capability objective describes the objective that the capability can satisfy. Each attribute $attr_i$ is defined in terms of attribute name (**attr_name**) and attribute value (**attr_value**) as follows:

$$PL = \{cname, attr_1, \ldots, attr_n\}, \text{ where}$$

$$cname = <action\_verb_i, noun_i>, attr_i = <attr\_name, attr\_value>, \text{ and } attr_i \in PL$$

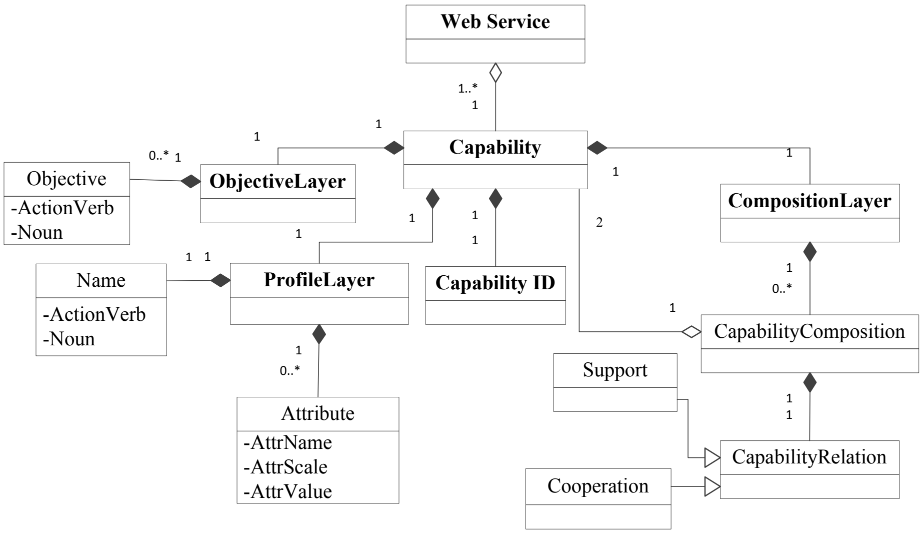


Fig. 2. The Capability Model

3. Composition Layer

The composition level consists of different capability compositions (**cc**) describing relations between capabilities. The composition layer is defined as

$CL = \{cc_i, cc_2, \ldots, cc_n\}$, $cc_i = <cap_p, cap_q, rel>$ and $cc_i \in CL$ such as cap_p and cap_q are two Web service capability profiles; *rel* is the relation between the two capabilities. We identify two kinds of composition relations between capabilities:

1) **Cooperation Relationship**: two capabities should cooperate with each other.

2) **Support Relationship**: the execution of one capability depends on the successful execution of its previous capabilty.

Based on the capability model, the formalized business requirements (FBR) for Web service composition are modeled as a list of capability profiles.

$$FBR = \{cap_1, cap_2, cap_3, \ldots, cap_n\}$$

Table 2. Capability Profiles Examples

ID	Objective Layer	Profile Layer		Composition Layer
		cname	attr	
cap$_{1.1}$	{<manage, crisis>}	<evacuate, population>	{(AvailableRegion, France)}	
cap$_{1.2}$	{<manage, crisis>}	<evacuate, population>	{(AvailableRegion, Marseille)}	
cap$_2$	{<manage, crisis>, < rescue, people>}	<transport, victim>	{(MaxTransportNumber, 100)}	{cap$_2$, cap$_3$, Support}
cap$_3$	{<manage, crisis>, <rescue, people>}	<assist, victim>	{(MaxAssistNumber, 300)}	{cap$_2$, cap$_3$, Support}

3.3 Web Service Requirement (WSR)

Web service requirements refer to constraints related multiple constraints on/among functional and non-functional Web service properties. They consists of desired Web services to be executed and constraints on/among them. We structure multiple constraints on/among Web services with Structure Rules and QoS Rules.

1) **Structure Rules** indicate composition patterns between Web services. A composition rule **sr**$_m$ is defined as a tuple: $sr_m = <s_l, s_r, cp>$, where s_l, s_r are the two sub services, *cp* is the composition pattern, and *cp is* sequence, parallel or branch operator, describing the control flow in the composed Web service.

2) **QoS Rules** represents constraints on Web services QoS attributes. The QoS of a service s_i is represented as $\mathbf{s_i.QoS} = \{q_1, q_2, \ldots, q_n\}$, where q_j represents the QoS attribute *j* of service s_i. A QoS rule $\mathbf{qr_n}$ is defined as: $qr_n = <s_i, q_j, expression>$ where $expression = <operator, value>$, and s_i, and q_j are respectively the Web service and its QoS attribute to be constrained. The *expression* describes constraints whereas *operator* $\in \{<, \leq, >, \geq, =, \supseteq, \ldots\}$ and *value* indicates QoS attribute values.

The Web service requirement (WSR) for composition is modeled as a list of Web services to be executed and a list of composition rules:

$$WSR = \{s_1, s_2, \ldots, s_l, sr_1, sr_2, \ldots, sr_m, qr_1, qr_2, \ldots, qr_n\}$$

4 Composition Requirement Transformation Process

In order to achieve requirement transformations between different levels, we propose **Capability Matching**, and **Association Discovery** algorithms as illustrated in Fig. 3.

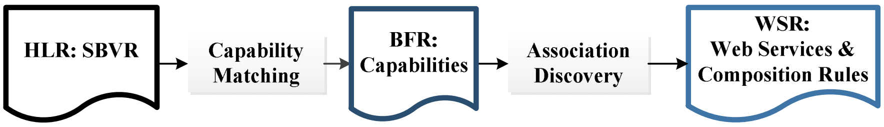


Fig. 3. General Methodology of Composition Requirements Transformation

These algorithms are based on semantic matching between concepts related to different requirement levels by using ontologies. We apply the concept matching algorithm proposed by [14] to cover three cases of matching between two concepts: 1) two concepts are identical, synonyms or one concept is generalization of another concept with reference to common ontology. Due to page size limit, we only describe these two algorhtms as follows:

Capability Matching Algorithm: transforms high-level requirements to formalized business requirements by matching concepts from the high-level requirement model with concepts from the capabilty model following two principles:
Principle 1. Functional reuqirements in the HLR are matched against objective layer and profile layer of capability profiles. Capabilities satisfying objectives and information defined in functional reuqirements are discovered and added to a primary set;
Principle 2. Non-functional requirements and context of HLR are matched against capability profiles layer. Capabilities dissatisfying rules specified in the NR are deleted from the primary result.

Association Discovery Algorithm: Given discovered capabilities from the capability-matching algorithm, the association discovery algorithm transforms capabilities' descriptions into composition rules among web services following three principles:
Principle 1. For each discovered capability, all Web services that are associated with the capability are discovered and added to the Web service discovery set.
Principle 2. If NR describes a constraint on and among certain capabilities, then a composition rule will be created on and among all Web services that are associated with these capabilities.
Principle 3. If a composition relationship between two capabilities is defined in the compositon layer of the capability profiles, a structure rule is created between Web services that are associated with the two capabilities.

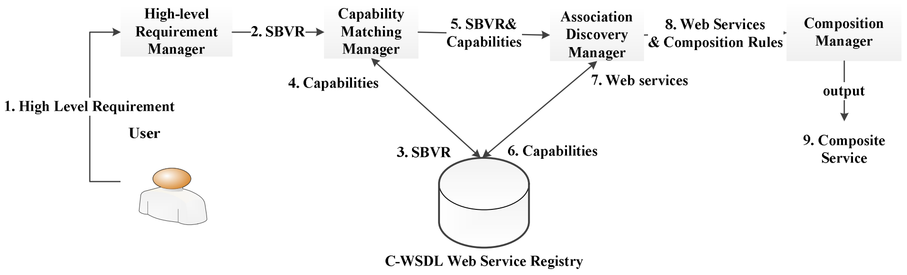

High-level Requirement Manager
2. SBVR
Capability Matching Manager
5. SBVR& Capabilities
Association Discovery Manager
8. Web Services & Composition Rules
Composition Manager
4. Capabilities
7. Web services
output
1. High Level Requirement
User
3. SBVR
6. Capabilities
9. Composite Service
C-WSDL Web Service Registry

Fig. 4. Prototype Architecture for Requirement Transformation

5 Conclusion

In this paper, we propose an end-to-end transformation approach between high-level requirements and composition requirements via a Web service capability model to reduce the gap between requirements expressed by casual users in SBVR and technical requirements related to Web service composition process. We have demonstrated the feasibility of our transformation method by developing a prototype comprising different modules implementing the capability matching algorithm and the association discovery algorithm (see Fig. 4). We have also integrated the C-WSDL Web service registry and the rule-driven Web service composition algorithm developed in our previous work [15] i.e., Service Farming. Based on the train's crisis case study, preliminary results illustrate the transformation process and generate a set of composition rules. As a future work, we are studying the Intuitionist Linear Logic to prove whether a set of capabilities can be found to ensure that high-level requirements can be translated in set of composition rules among a discovered set or Web services.

References

1. Dustdar, S., Schreiner, W.: A Survey On Web Services Composition. International Journal of Web and Grid Services 1, 1–30 (2005)
2. OASIS Reference Model for Service Oriented Architecture 1.0 (2006), `http://www.oasis-open.org/committees/download.php/19679/soa-rm-cs.pdf`
3. Jazuli Bin Kamarudin, N., Sani, N.F.M., Atan, R.: Transformation from requirement into behavior design: A review. In: 2012 International Conference on Information Retrieval & Knowledge Management (CAMP), pp. 153–157 (2012)
4. Yue, T., Briand, L.C., Labiche, Y.: A Systematic Review of Transformation Approaches Between User Requirements and Analysis Models. Requirements Engineering 16, 75–99 (2011)
5. Grünbacher, P., Egyed, A., Medvidovic, N.: Reconciling Software Requirements and Architectures With Intermediate Models. Software and Systems Modeling, 235–253 (2004)
6. Fliedl, G., Kop, C., Mayr, H.C., Salbrechter, A., Vöhringer, J., Weber, G., Winkler, C.: Deriving Static and Dynamic Concepts from Software Requirements Using Sophisticated Tagging. Data Knowledge Engineeriong 61, 433–448 (2007)
7. Subramaniam, K., Liu, D., Far, B.H., Eberlein, A.: UCDA: Use Case Driven Development Assistant Tool for Class Model Generation. In: 16th International Conference on Software Engineering and Knowledge Engineering (SEKE 2004), Banff, Canada (2004)
8. Samarasinghe, N., Somé, S.: Generating a Domain Model from a Use Case Model. In: Proceedings on Intelligent and Adaptive Systems and Software Engineering (2005)
9. Mukhopadhyay, D., Chougule, A.: A Survey on Web Service Discovery Approaches. In: Wyld, D.C., Zizka, J., Nagamalai, D. (eds.) Advances in Computer Science, Engineering & Applications. AISC, vol. 166, pp. 1001–1012. Springer, Heidelberg (2012)
10. Roman, D., de Bruijn, J., Mocan, A., Lausen, H., Domingue, J., Bussler, C., Fensel, D.: WWW: WSMO, WSML, and WSMX in a nutshell. In: Mizoguchi, R., Shi, Z.-Z., Giunchiglia, F. (eds.) ASWC 2006. LNCS, vol. 4185, pp. 516–522. Springer, Heidelberg (2006)
11. Martin, D., Paolucci, M., Wagner, M.: Towards Semantic Annotations of Web Services: Owl-S from the SAWSDL Perspective. In: Proceedings of the OWL-S Experiences and Future Developments Workshop at ESWC (2007)
12. Bhiri, S., Derguech, W., Zaremba, M.: Web Service Capability Meta Model. In: WebIST 2012, pp. 47–57 (2012)
13. Fayoumi, A., Yang, L.: SBVR: Knowledge Definition, Vocabulary Management, and Rules Integrations. International Journal of E-Business Development (2013)
14. Meditskos, G., Bassiliades, N.: Structural and Role-Oriented Web Service Discovery with Taxonomies in OWL-S. IEEE Transactions on Knowledge and Data Engineering 22, 278–290 (2010)
15. Li, W., Badr, Y., Biennier, F.: Service farming: An Ad-Hoc and Qos-Aware Web Service Composition Approach. In: Proceedings of the 28th Annual ACM Symposium on Applied Computing, pp. 750–756. ACM, New York (2013)

DOA-Trusted Cloud 2013 PC Co-Chairs Message

Welcome to DOA-Trusted Cloud 2013, the third International Symposium on Secure Virtual Infrastructures, organized as a component conference of the On-TheMove Federated Conferences & Workshops, held in Graz, Austria, September 2013.

Distributed computing is undergoing a radical paradigm shift, with cloud computing and service orientation enabling new business models based upon seamless provision of dynamically scalable, virtualized resources as services to be made available over the cloud, to be accessed via a web browser. For this vision to be realized, a number of problems need to be solved, involving the nuts-and-bolts of virtualization as well as the reliability, scalability, privacy, security and transparency of cloud computing.

This conference provides a forum for discussion of these topics, and linkage between these potentially diverse areas. Consequently, the major topics of interest of DOA-Trusted Cloud 2013 are "Cloud Data Management", "Cloud Computing Infrastructures and Architectures", "Cloud Computing Applications", and "Security and Trust in Cloud Computing". All submitted papers passed through a rigorous selection process. In the end, we decided to accept 5 regular papers from the original 17 submissions, grouped them into 2 sessions, "Technical Advances in Cloud Computing" and "Towards Trusted Cloud Computing", and additionally incorporated an also peer-reviewed special session "Privacy for the Cloud" containing 4 papers into the final program.

At this point we want to recognize all contributors DOA-Trusted Cloud 2013. First of all we would like to gratefully acknowledge the work of all of the authors, whether or not their paper was accepted, and thank them for choosing DOA-Trusted Cloud 2013 to present their research work. Secondly, we are grateful to the dedicated work of the leading experts in the field from all over the world who served on the Program Committee. Thirdly, we would like to thank Michele Bezzi for putting together the special session on "Privacy for the Cloud". Finally, we would like to thank the whole OTM team for its support and guidance, including the General Co-chairs Tharam Dillon and Robert Meersman, and the OTM Secretariat. We hope that you enjoy DOA-Trusted Cloud 2013 proceedings.

July 2013

Norbert Ritter
Makoto Takizawa
Volkmar Lotz

RouPar: Routinely and Mixed Query-Driven Approach for Data Partitioning

Ladjel Bellatreche[1], Amira Kerkad[1], Sebastian Breß[2], and Dominique Geniet[1]

[1] LIAS/ENSMA – Poitiers University
Futuroscope, France
{bellatreche,amira.kerkad,dgeniet}@ensma.fr
[2] University of Magdeburg
D-39016, Germany
bress@iti.cs.uni-magdeburg.de

Abstract. With the big data era and the cloud, several applications are designed around analytical aspects, where the data warehousing technology is in the heart of their construction chain. The interaction between queries in such environments represents a big challenge due to three dimensions: (i) the routinely aspects of queries, (ii) their large number, and (iii) the high operation sharing between queries. In the context of very large databases, these operations are expensive and need to be optimized. The horizontal data partitioning ($\mathcal{HDP}$) is a pre-condition for designing extremely large databases in several environments: centralized, distributed, parallel and cloud. It aims to reduce the cost of these operations. In $\mathcal{HDP}$, the optimization space of potential candidates for partitioning grows exponentially with the problem size making the problem NP-hard. In this paper, we propose a new approach based on query interactions to select a partitioning schema of a data warehouse in a divide and conquer manner to achieve an improved trade-off between the optimization algorithm's speed and the quality of the solution. The effectiveness of our approach is proven through a validation using the Star Schema Benchmark (100 GB) on Oracle11g.

1 Introduction

The big data era and the cloud bring two main aspects related to query optimization: (i) large amount of data [17] and (ii) the complexity of the repetitive nature of analytical queries that require multiple joins, aggregations, etc. One of the main characteristics of analytical queries in a data-warehouse environment is their interaction. *Multiple Query Optimization* (MQO) is a tool that captures this interaction by identifying common tasks among query plans (e.g., common sub-expressions, joins, etc.) through a single unified plan (sometimes called the *Multiple View Processing Plan* [19]). Unified plans represent a visualization tool of all queries and a data structure to capture the interaction among queries. Note that query interaction ($\mathcal{QI}$) has been largely used in the query optimization area in traditional databases [14] and recently in semantic databases (e.g., SPARQL queries) [8]. It contributed massively in selecting materialized views [19], caching and query scheduling [10,1].

On the other hand, the horizontal data partitioning ($\mathcal{HDP}$) is a pre-condition for designing databases on various environments: centralized [13], parallel [5], distributed [11], cloud [4,17], etc. Two versions of the $\mathcal{HDP}$ are available [3]: primary and derived $\mathcal{HDP}$. The primary $\mathcal{HDP}$ of a table $\mathcal{T}$ is performed using predicates defined on $\mathcal{T}$. The derived $\mathcal{HDP}$ is the partitioning of a table that results from predicates defined in other table(s). The derived $\mathcal{HDP}$ of a table $\mathcal{T}$ according to a partitioning schema of table $\mathcal{S}$ is feasible if and only if there is a join link between $\mathcal{T}$ and $\mathcal{S}$. This partitioning is well adapted in the context of relational data warehouses, where a fact table may be partitioned based on the partitioning schema of dimension tables. The derived $\mathcal{HDP}$ has two main advantages in relational data warehouses, in addition of classical benefits of data partitioning: **(1)** pre-computes join operation between fact table and dimension tables participating in the fragmentation process of the fact table. It uses the semi join operations to optimize the star join queries. The semi join has been largely used by commercial DBMS when optimizing star join queries [6]. **(2)** It optimizes selection operations defined on dimension tables. The number of horizontal fragments of the fact table generated by the partitioning procedure is given by the following equation: $\mathbf{N} = \prod_{\mathbf{i=1}}^{\mathbf{g}} \mathbf{m_i}$, where m_i is the number of fragments of the dimension table D_i. This number may be very large. Some studies propose to constrain this number [16].

The $\mathcal{HDP}$ has been implemented by academic and commercial DBMS and is nowadays an integral part of physical design in the most important DBMS such as *Oracle*, *DB2*, *SQL Server*, *PostgreSQL* and *MySQL*. A native Database Definition Language to support it is available by the means of various partitioning modes [15]: range, list, hash and composite.

The problem of selecting a partitioning schema is formalized as follows: Given a relational data warehouse with a set of d dimension tables $D = \{D_1, D_2, ..., D_d\}$ and one fact table F, a workload Q of queries $Q = \{Q_1, Q_2, ..., Q_n\}$, where each query Q_i $(1 \leq i \leq n)$ has an access frequency $AF(Q_i)$. The $\mathcal{HDP}$ selection problem consists in fragmenting the fact table F into N fragments minimizing the query cost subject to maintenance constraint $N \leq W$, where W is a threshold, configured by a database administrator (DBA), representing the maximal number of fact fragments that the partitioning algorithm may create. This problem has been shown as NP-hard [2,12]. A number of important algorithms were proposed in the context of databases and data warehouses. Most of these algorithms [2,9,11] do no consider the interaction between queries [18]. The main efforts focused on the algorithmic level, where several varieties of algorithms were proposed such as hill climbing, genetic, simulated annealing, and data mining driven algorithms. Each algorithm has its advantages and limitations. The algorithms that had a good performance use simple cost models to guide the exploration of solutions [12] and are time consuming. $\mathcal{HDP}$ has to integrate the routinely and mixed queries characterized applications build around the cloud. For example, in Alibaba's open data processing service distributed system implemented based on the *MapReduce* framework for massive data analysis daily runs more than 20,000 queries due to the business needs and 40% of the statements share similar data operations and structural characteristics [7].

To incorporate the $\mathcal{QI}$ in the $\mathcal{HDP}$ selection process, three main alternatives may be distinguished: (1) the $\mathcal{QI}$ is delegated to the used cost model when estimating the overall query processing, (2) the development of $\mathcal{HDP}$ algorithms exploiting the data structure of the unified graph and (3) an hybrid alternative. The first alternative may increase the complexity of the algorithms, whereas the second and the third alternatives may reduce significantly this complexity and offering partitioning schema with high quality.

In this paper, we propose a new approach for partitioning relational data warehouses used by $\mathcal{SSDB}$ considering the interaction between queries at the data structure and cost model levels.

Our paper is structured as follows. In Section 2, a motivating example is described to show the basic ideas of our approach and to facilitate the description of our algorithms. Section 3 discusses the basic definitions and algebra for manipulating our data structure. Section 4 presents an algebra managing partitions. Section 5 presents our partitioning algorithm, called Elected Query Algorithm. Section 6 presents the results of our heuristics and compares the results with those obtained by state of art studies. The paper concludes in Section 7.

2 Motivating Example

Let us consider a star schema database with one fact table *Sales* and three dimension tables $\{Time\ ;\ Product\ ;\ Customer\}$. The data is accessed by a workload of 10 star join queries. The execution plans of all queries are merged in one graph called *Multi-View Processing Plan* (MVPP). MVPP is a graphical representation of a workload proposed in the context of Multi-Query Optimization [14]. Note that selection operations are pushed down after constructing MVPP. This graph has four main levels: **(a)** leaf nodes representing the base tables, **(b)** selection nodes that may be used to partition the databases (e.g., $\{s_1, s_2 \dots s_8\}$), **(c)** binary operation nodes (e.g., joins $\{j_1; j_2 \dots; j_9\}$) and **(d)** root nodes (e.g., grouping, ordering and projection $\{gop_1;gop_2; \dots gop_{10}\}$).

The join operation j_3 between table *Sales* and *Product* is shared by four queries Q_4, Q_7, Q_9 and Q_{10}. If we want to optimize the query Q_4 involving two selections $\{s_1, s_2\}$ and a join j_3, the $\mathcal{HDP}$ using the selection attributes may be a relevant optimization structure. The optimization of the query Q_4 impacts not only the query Q_4, but its benefit will propagate through all queries interacting with join node j_3: Q_7, Q_9 and Q_{10}. To spread the benefit along query plans, having the constraint of threshold[1] W limiting the number of fragments, the choice is very hard to be done. If we consider the query interaction in our example workload, we can group queries into four disjoint subsets depending on shared joins: $\{Q_1\}$; $\{Q_2; Q_3\}$; $\{Q_4; Q_7; Q_9; Q_{10}\}$; $\{Q_5; Q_6; Q_8\}$, that we call

[1] For the rest of the paper, N will refer to the number of generated fragments of the fact table in the partitioning schema.

groups. Note that the number of groups equals the number of direct joins with the fact table (four in the example). The reason is that queries in the same group are correlated by sharing at least the first join. This join is very expensive since it involves the largest table (fact table). If the first join is optimized using $\mathcal{HDP}$ based on its selections, all queries in the same group will benefit from the partitioning.

The threshold W imposes to start by most beneficial queries and predicates before the maximal number of fragments is reached. The problem at this level is : (1) which queries should we favor to steer partitioning, and (2) by which attributes should we start splitting and merging data?
The choice of important queries and attributes is crucial, because it allows lowering execution cost of the workload without exceeding the threshold W.

– If we choose to optimize the least expensive query of each group, we may ensure the gain propagation through all queries. This is caused by the interaction of each chosen query with the remaining ones in its group by sharing at least one common join. On the other hand, the cheapest query in a group may have fewer operations than others. These operations are concentrated in lower levels in the MVPP (e.g., join j_3 in query Q_4). The choice of the cheapest query allows to optimize lower nodes in the MVPP which are accessed by most (or even all) queries in the same group.
– From each elected query for partitioning, a set of attributes may participate in the $\mathcal{HDP}$ process. To start by the most important attributes, the usage rate may be a relevant criterion to spread a larger benefit through queries.

Observing this phenomenon, we deduced the impact of such properties to conduct $\mathcal{HDP}$. Therefore, we propose a new approach for $\mathcal{HDP}$ that exploits query interaction in a divide and conquer strategy to prune the large search space, and lower execution cost of the workload considering the constraint W.
We use a tailor-made data structure for representing partitioning schemas, which are potential solutions for $\mathcal{HDP}$. This structure contains participating attributes and sub-domain decomposition. If the data structure is based on our MVPP, it can be used to identify the candidate selection predicates for partitioning. The sub-domains decomposition considers existing predicates in the MVPP, and the least used attributes can be removed from the data structure in order to prune the search space.

3 Background

In this section, we present the formalization of our problem and the basic concepts that facilitate the understanding of our proposal.

We formalize the problem of $\mathcal{HDP}$ as follows: given (1) a data warehouse schema DW with one fact table F and a set of dimension tables $D_1, D_2, \dots D_k$, (2) a workload $\mathcal{Q} = \{Q_1, Q_2, \dots Q_n\}$, and (3) a constraint of a threshold W limiting the maximal number of fact fragments; the $\mathcal{HDP}$ aims at providing a new schema for the data warehouse that partitions dimension tables into dimension fragments using primary horizontal partitioning, and the fact table into N

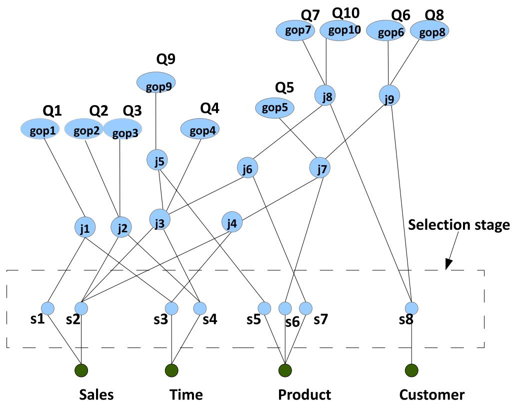


Fig. 1. Example of MVPP of 10 queries

Table 1. Existing Selections in the MVPP

Alias	Predicate	Table
s_1	$\sigma(quantity < 100)$	Sales
s_2	$\sigma(100 \leq quantity < 1000)$	Sales
s_3	$\sigma(season = "Automn")$	Time
s_4	$\sigma(season \in \{"Winter"; "Spring"\})$	Time
s_5	$\sigma(type = "T1")$	Product
s_6	$\sigma(color = "Color1")$	Product
s_7	$\sigma(type = "T2")$	Product
s_8	$\sigma(gender = "Female")$	Customer

fragments using derived horizontal partitioning from dimension fragments. The obtained schema DW' minimizes the execution cost of the workload $\mathcal{Q}$, and the number of fact fragments N must not exceed the threshold W: $N \leq W$.

3.1 Decomposition of an Attribute's Domain into Sub-domains

The $\mathcal{HDP}$ schema of a given $\mathcal{SSDB}$ is based on selection predicates defined in the queries of a given workload. Each selection predicate is defined as follows: $Att\ \theta\ Val$, where Att is the selection attribute, $\theta \in \{=, <, >, \leq, \geq, \in \dots\}$ and Val is a value, a list or an interval in the values domain of the attribute Att. Table 1 contains the set of selection predicates in the MVPP shown in Figure 1.

The sub-domains are intervals, lists, or values in the domain of a selection attribute. The decomposition of a domain to many sub-domains is usually done either by the DBA based on experiments or by query selection predicates [16]. Some attributes are harder to decompose then others. In the traditional $\mathcal{HDP}$, the attribute domain decomposition is performed independently from the MVPP. As a consequence, this decomposition is static and fixed. In this paper, we propose an incremental encoding that does the decomposition systematically based on the MVPP. Figure 2-a illustrates an example of attribute domain decomposition obtained from the MVPP.

3.2 Encoding Schema

Usually, each $\mathcal{HDP}$ schema may be represented by a fixed coding [2], which is a juxtaposition of arrays representing selection attributes of the selection stage of the MVPP (Figure 1), where each array represents the domain decomposition of an attribute. The value of each cell of a given array representing an attribute A_i^k of dimension table D_k belongs to $[1, n_i]$, where n_i represents the number of sub-domains of the attribute A_i^k. An example of a $\mathcal{HDP}$ schema encoding is given in

(a)

Quantity	<100	>=100 and <1000	else	
Season	Automn	Winter	Spring	Summer
Color	color1	else		
Type	T1	T2	else	
Gender	Female	Male		

(b)

Quantity	1	2	2	
Season	1	2	1	3
Color	1	2		
Type	1	1	2	
Gender	1	1		

Fig. 2. Selection attributes and their sub-domains (a) and encoding schema (b)

Figure 2-b. Based on this representation, $\mathcal{HDP}$ schema of each dimension table D_j is generated as follows:

1. All cells of a partitioning attribute A_i^k of D_j have different values: this means that all sub-domains will be used to partition D_j. For instance, the cells of each partitioning attribute *Color* in Figure 2-(b) are different. So attribute *Season* will participate in partitioning process by splitting table *Time* into three disjoint subsets.
2. All cells of a partitioning attribute A_i^k have the same value: this means that it will not participate in the partitioning process. Attribute *Gender* for example will not participate in $\mathcal{HDP}$.
3. Some cells of an attribute sub-domains have the same value: their corresponding sub-domains will be merged into one fragment. For example, *Season* will split data into three subsets : {Automn,Spring}, {Summer} and {Winter}.

The primary horizontal partitioning on a dimension table D_i is applied using all participating attributes of this table. The set of participating attributes provide all different combinations of predicates that generate dimension fragments of D_i. For example, table *Product* is partitioned based on attributes *Type* and *Color* to generate four fragments as follows :

– $Product_1$: $Type \in \{T_1, T_2\}$ and $Color =$ "$Color1$"
– $Product_2$: $Type \notin \{T_1, T_2\}$ and $Color =$ "$Color1$"
– $Product_3$: $Type \in \{T_1, T_2\}$ and $Color \neq$ "$Color1$"
– $Product_4$: $Type \notin \{T_1, T_2\}$ and $Color \neq$ "$Color1$"

If attribute *Color* for example were not participating, table *Product* would provide two fragments : $Product_1$ where $Type \in \{T_1, T_2\}$, and $Product_2$ where $Type \notin \{T_1, T_2\}$. Most of partitioning approaches use a static coding to represent and handle their solutions. In this work, we propose an incremental way to generate the encoding in order to keep only relevant attributes and sub-domains for partitioning process. In the experimental study, we will also show that resolution algorithms are encoding-sensitive.

4 Algebra

The definition of an encoding and partitioning algebra are required to deal with $\mathcal{HDP}$. To do so, we propose two functions : *Horizontal Split* and *Vertical Split* to incrementally generate the encoding used to represent the partitioning schema.

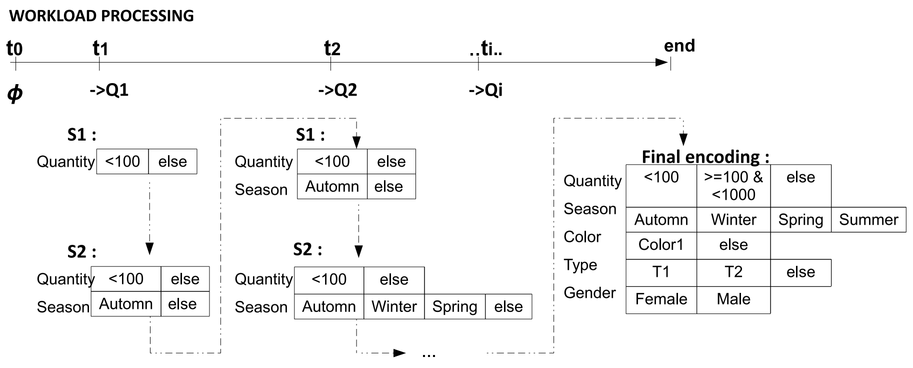


Fig. 3. Incremental encoding by Horizontal and Vertical Split on sub-domains of attributes

Then, two other functions : *Split* and *Merge* are used to manage partitions in the schema.

4.1 Generating Incremental Encoding

The encoding proposed in our work aims at providing a pruned set of attributes and sub-domains which are relevant to the MVPP. To generate new attributes in the encoding schema, a *Vertical Split* is applied to add a new array representing the new attribute. The set of sub-domains of each attribute A^k is updated by performing *Horizontal Split* on the existing array of A^k, which creates new ranges in sub-domains of A^k.

To obtain the set of selection attributes, we proceed by doing a first pruning of all attributes. The idea is to keep only attributes and their sub-domains that figure in the MVPP. The construction of the encoding is done incrementally by exploring query plans one by one, and updating the coding by new attributes and sub-domains, using the two functions : *Horizontal and Vertical Split.* The principle of this coding is to start from an empty set of attributes, and for each query, add its required selection attributes by creating new arrays. Each array contains one range. When a selection is found for the current query, three operations can be performed:

1. If the attribute does not exist in the schema, apply a vertical split to extend the schema vertically by adding a new array for the attribute. The range is split into many parts to cover the new sub-domains. An *else* range is added to ensure completeness. Figure 3 shows the added attributes to the empty set of selection attributes by query Q_1 (*Quentity, Season*).
2. If the attribute already exists, apply a horizontal split on the *else* range to add the new sub-domains. An example of added sub-domains by query Q_2 for attribute *season* is illustrated in Figure 3.
3. Finally, if the administrator knows the value remaining in the *else* range, it is replaced by this value. In Figure 3, the *else* is replaced by *"Male"* for attribute *Gender*, and by *"Summer"* for attribute *season.*

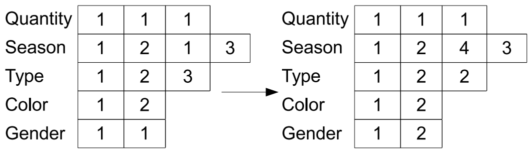


Fig. 4. Performing Split and Merge operations on the partitioning schema

The result of this phase is an encoding schema containing the set of selection attributes and their associated sub-domains. To run our encoding algorithm on our example, we start from an empty set of attributes. When the first query Q_1 is processed, two new attributes are added with their sub-domains in Q_1. The query Q_2 does not have new attributes, but it has new predicates that split horizontally the existing schema (Figure 3). Finally, when the last query Q_{10} is processed, the schema is adjusted with some known values to replace the *else* (Figure 3).

4.2 Partitioning Primitives

In most commercial DBMS, the $\mathcal{HDP}$ proposes primitives to manage partitions by the use of two basic functions: *Merge* and *Split.* The former consists in merging two partitions in one, and the later consists in splitting a partition into two partitions. These operations are performed on the physical level of a database. We propose to use these primitives to generate partitioning schemas.

More formally, the *Merge* and *Split* operations have the following signature: $Merge(sd_i^k, sd_j^k, A^k, PS) \rightarrow PS'$, where the *Merge* function is applied on two sub-domains sd_i^k and sd_j^k of the attribute A^k to get them in one partition. $Split(sd_i^k, A^k, PS) \rightarrow PS'$, where the *Split* function is applied on the sub-domain sd_i^k in order to be divided into two sub-domains if and only if it covers at least two sub-domains of attribute A^k. Figure 4 shows one *Merge* performed on attribute *Type*, and two *Splits* on attributes *Season* and *Gender.*

5 The Elected Query Algorithm

In this section, we show how query interaction is exploited in the $\mathcal{HDP}$ process by promoting overlapping join nodes. To exploit these nodes, we apply a divide and conquer algorithm (called *"Elected Query Algorithm"* (EQA)) to prune the space of relevant attributes and their sub-domains. We describe how EQA conducts the $\mathcal{HDP}$, and provide the detailed algorithm. Starting from the obtained set of candidate attributes kept in the encoding schema, a partitioning schema needs to be generated to decrease execution cost as possible. As already mentioned, existing solutions are either (1) *simplistic*, or (2) *heuristics*, based on cost models. To achieve an improved trade-off between the two approaches, we propose to add a new criterion to conduct the $\mathcal{HDP}$ process, to get a better efficiency and less complexity.

5.1 Electing Queries

As discussed in the motivating example, the join operation is very expensive, and at the same time, queries interacting with each other may share at least the first join. Considering the first join node is crucial, because it serves other queries sharing that node. To capture and exploit the interaction of queries and reduce the complexity of resolution algorithms, we propose an algorithm named *Elected Query Algorithm* (EQA) to get the set of most beneficial queries. Starting from the MVPP, the EQA generates groups of correlated queries, such that each couple of queries sharing at least one join node are in the same group. Figure 5 shows the obtained groups from the MVPP in the motivating example. From each group, the algorithm aims at electing one query to steer $\mathcal{HDP}$. We call this query : *Elected Query* (EQ) and consider it as the most important one in its group. The choice of the EQ_i for group i is done regarding the cost and the overlapping nodes of the query. We propose to choose the EQ as the one with minimal cost. The minimal cost allows having a minimal set of operations (less joins may sire less selections). The number of overlapping nodes is not considered, because it shares at least one join which may propagate the benefit through all queries in the same group. The EQA is given in the following algorithm. In addition, choosing the query with least cost allows optimizing lower nodes (operations) in the MVPP, which are the most used, so that it propagates the benefit through most queries.

The EQA aims at giving a subset of relevant selection attributes and subdomains to guide the process of HDP. The returned set of elected queries may prune the search space and discard less efficient possibilities such as less frequent selections and joins. The obtained attributes from MVPP in the previous phase will be pruned again by considering, at first, only attributes participating in the elected queries. In the partitioning process, the algorithm satisfies the elected queries rather than randomly chosen predicates. This pruning phase may reduce considerably the complexity of the partitioning algorithm by reducing the size of the search space. The details of the algorithm of $\mathcal{HDP}$ using EQA are given in the next section.

Algorithm 1. ElectedQueryAlgorithm()

```
1: for all $Q_i \in MVPP$ do
2:     $Group(Q_i) := Q_i$;
3: end for
4: for all $(Q_i, Q_j) \in MVPP$ do
5:     if ( $(i! = j)$ and $(interaction(Q_i, Q_j) = true)$) then
6:         $merge(Group(Q_i), Group(Q_j))$;{grouping queries based on interaction}
7:     end if
8: end for
9: for all Group a do
10:     $sort(a)$;{sort queries by minimal cost}
11:     $elect(a, 1)$;{elect a query by minimal cost}
12: end for
```

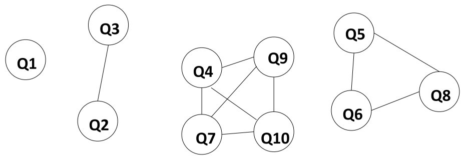


Fig. 5. Generating groups (subsets) of interacting queries from the MVPP

5.2 Exploiting EQA for HDP

Our algorithm for $\mathcal{HDP}$ called *Electing Queries for Horizontal Data Partitioning* (EQHDP) starts from the MVPP, and performs pruning of selection attributes by electing most "beneficial" queries using EQA. The EQA groups queries into disjoint subsets, then sorts queries inside each group by minimal cost. The EQA returns the set of elected queries EQ_i from each group i. The obtained set of elected queries from EQA is sorted by descending costs. That way, the most expensive queries are optimized first and the algorithm can ensure that the most expensive part of the workload is optimized before the threshold W is exceeded.

EQA computes first the elected queries and then prunes the schema of attributes depending on the elected queries requirements, i.e., only required attributes are taken, the others are ignored. The sub-domains are tagged with the total number of elected queries using each one. Let u_i be the number of EQ using a sub-domain of attribute a_i and let k_i be the maximal value of u_{ij} in a_i. The set of attributes is sorted by maximal use (value of k_i) in order to start by partitioning on most used attributes before the W is exhausted. After sorting the attributes, each attribute a_i is split/merged as follows:

1. The sub-domains, which are not used by any elected query ($u_i = 0$), are grouped in one partition P_0;
2. The most used sub-domains having k_i elected queries accessing them are all grouped in one partition P_k (where k_i is the maximal usage value of the attribute a_i);
3. Finally, if $N > W$ or $k_i \neq 0$ then, the remaining sub-domains are merged with the partition P_0; otherwise, the sub-domains accessed by $k_i - 1$ elected queries are grouped in a new partition. The same operation is repeated until $k_i = 0$ or $N > W$.

This allows creating partitions based on the requirements of most important queries. If partitioning is still possible ($N < W$), the obtained partitions are split depending on the correlation between queries accessing each sub-domain of the partition. If two subsets of elected queries require some sub-domains independently, a new partition is created to contain sub-domains used independently from the others.

If N is less than W after these merge and split operations regarding only the elected queries, then partitioning is still possible. For this reason, the optimization process moves to other queries to improve their performance as well. The next set of queries is the successors of current EQs that verify the same criterion as the already processed queries. If at least one group has still a successor,

Algorithm 2. AlgorithmeEQHDP()

```
1: generate_encoding();
2: EQA();{grouping by Elected Query Algorithm}
3: split_all();
4: $E := elected()$
5: while E not empty do
6:    prune_encoding(E);
7:    sort(E);
8:    $S := required\_attributes(E)$;
9:    usage(S);
10:   $attributes\_sort(S)$
11:   while $((N < W)$ and $(S$ not empty$))$ do
12:      for all $sd \in SubDomains(a)$ do
13:         $j := U(sd)$;
14:         $merge(sd, P_j)$; {merge sub-domains in partition $P_j$}
15:         $S := S - \{a\}$;
16:      end for
17:   end while
18:   $split\_disjoint()$;{split sub-domains used by disjoint sets of queries}
19:   $E := elected\_successors()$;
20: end while
```

a new set of elected queries is generated by the found successors of all groups. The same process is applied by extending the encoding schema with the new set of selection attributes incrementally. The partitioning is done until $N = W$ or no more queries are left in **any** group. We formalize our proposed EQHDP approach in the following algorithm:

6 Experimental Study

In this section, we contribute an extensive experimental study to validate our proposal. We start by presenting our data set and a simulator connected to Oracle11g for performing experiments. The obtained results are discussed and justified to reveal particularities of our proposal. Finally, we discuss another advantage of our proposal regarding query profiles and similarity that may support other optimization techniques.

6.1 Data Set

In our experimental study, we use the *Star Schema Benchmark* (SSB)[2] with a scale factor of 100, which generates 100 GB of data. The data warehouse contains a fact table *Lineorder*, and four dimension tables : *Customer, Supplier, Part* and *Dates* stored on Oracle11g located on a server of 32GB of RAM and an Intel® Xeon® CPU of 2x2.40GHz. The workload consists of 22 star join queries running under Oracle11g.

[2] `http://www.cs.umb.edu/~poneil/StarSchemaB.pdf`

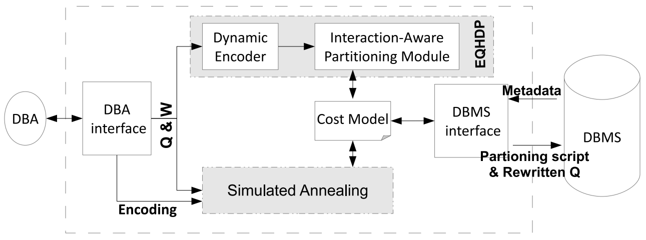


Fig. 6. The Components of our $\mathcal{HDP}$ Simulator

6.2 Infrastructure Components and Settings

To conduct our experiments, we use a simulator written in Java and connected to an Oracle11g instance. We illustrate the infrastructure of the experimental study in Figure 6. The main components are:

– *DBA interface* allows the DBA to provide the workload and the threshold $\mathcal{W}$ for both algorithms. The DBA provides also a pre-shaped encoding schema for Simulated Annealing.
– *DBMS interface* is responsible of extracting meta data from a Oracle11g DBMS, and at the same time, providing the partitioned schema with the rewritten queries to the DBMS.
– *Dynamic encoder* provides an encoding schema based on our algebra to be used by the partitioning module.
– *Interaction-Aware Partitioning Module* captures query interaction to steer the $\mathcal{HDP}$ using the generated encoding.
– *Cost Model* estimates the cost of executing the optimized queries on the obtained partitioning schema by both algorithms.

The entries are our database and its workload. The threshold W is varied to show its impact on performance. The metric used to quantify the quality of potential solutions is our cost model that provides the number of input/output operations required for executing queries. We compare EQHDP with *Simulated Annealing* (SA), because SA is widely used in such hard optimization problems. It computes results efficiently compared to a genetic algorithm [2]. To tune our SA, we set the iteration number to 500 and the temperature to 300. As the SA is a non-deterministic algorithm, it is run several times (from 5 to 10 times depending on the variance) to get the average performance. Contrary to our algorithm, which generates its encoding dynamically, the SA needs a pre-shaped encoding as additional input.

6.3 Simulation Results

Threshold Impact. In our first experiments, we compare the $\mathcal{HDP}$ schemas given by SA and our EQHDP algorithm. In Figure 7, the execution cost of the workload is given by the number of input/output operations. The results show that

EQHDP outperforms SA except for very large thresholds ($\mathcal{W} \geq 300$), with narrow differences in performance. The reason of that EQHDP efficiency is that it performs split/merge operations *on demand*, i.e., it aims at satisfying the largest number of queries, starting from the most efficient ones that contribute largely in improving the workload performance.

If $\mathcal{W} < 300$, SA performs a high number of iterations looking randomly for the best solution. In contrast, EQHDP detects and performs systematically the required moves to get the most beneficial partitioning schema. If $\mathcal{W}$ is very large (≥ 300), the EQHDP stops when all queries and their predicates are satisfied. Although further split/merge operations are still possible to decrease execution cost. These extra operations cannot be performed by the EQHDP, but can be reached randomly by SA. Note that $\mathcal{W}$ is the number of fact fragments, which is the number of resulting subschemas. The total number of fragments in the schema is much higher than $\mathcal{W}$.

We conclude from these experiments that the EQHDP can reach a high level of efficiency compared to a meta-heuristic. However, for very high values of threshold $\mathcal{W}$, the SA is more efficient with narrow differences.

Encoding Sensitivity. Another major advantage of our approach is that our algorithm is self-encoding. This means that the coding phase is a part of our algorithm and provides an encoding schema depending on the workload, contrary to the SA that needs a pre-computed encoding to run over. The given schema for SA is generally provided by the DBA, and may contain a better repartition of sub-domains for selection attributes.

To study encoding sensitivity of partitioning algorithms, we performed two experiments as follows: (1) The first one using incremental encoding process of the EQHDP, the obtained encoding is provided to the SA to run over it. (2) The second one is done using a DBA encoding schema for both algorithms. Figure 8 shows that using DBA encoding may make the SA more efficient, because more appropriate ranges and sub-domains are provided by the DBA, if he knows the workload and the data. But it may penalize the EQHDP because some query requirements do not fit in the given encoding.

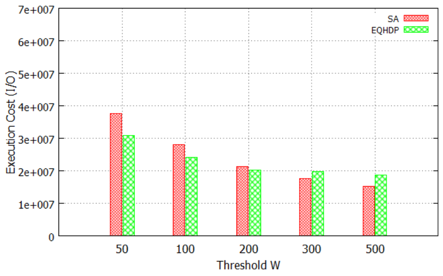


Fig. 7. Comparing algorithms performance by threshold variation

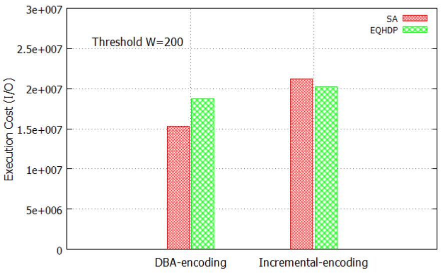


Fig. 8. Encoding sensitivity

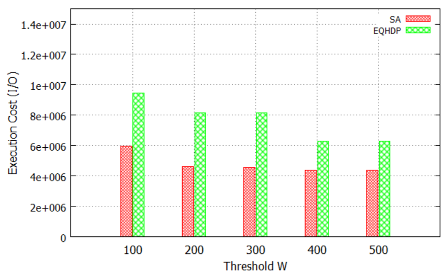


Fig. 9. Running SA and EQHDP on a workload with no interacting joins

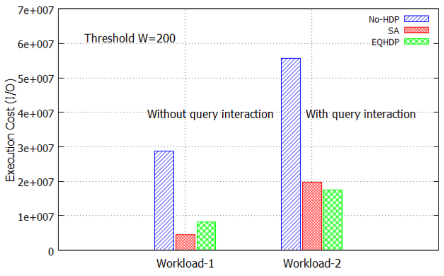


Fig. 10. The impact of query interaction on the efficiency of both algorithms

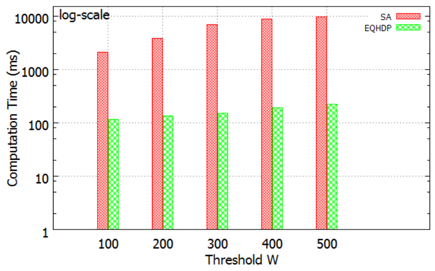


Fig. 11. Comparing reactivity of Simulated Annealing with the EQHDP algorithm

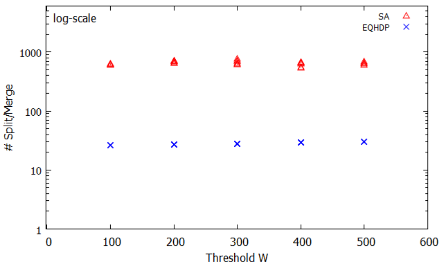


Fig. 12. Comparing #split/merge operations obtained by SA and EQHDP

From this experiments, we conclude that the resolution algorithms for the $\mathcal{HDP}$ are indeed encoding-sensitive. The SA cannot generate its own encoding. In the case where the DBA is not able to provide an efficient encoding schema, which is usually the case in real life, the EQHDP is more advantageous, because it is more autonomous and efficient.

Query Interaction Impact. In order to check the impact of query interaction on our approach, we use a new workload having no shared joins between queries (no common sub-expressions). Figure 9 shows the execution cost of the new workload optimized by both algorithms. The results show that the EQHDP is not efficient enough compared to the SA in case no interaction exists between queries. This is due to the fact that only a subset of queries is optimized, but the gain cannot propagate through other queries.

We compare now performance of EQHDP with and without query interaction. Figure 10 we can see that the EQHDP outperforms SA when queries are correlated, but execution cost of the workload is poor in case of no query interactions. From these experimentations, we argue that our approach is not efficient in case the workload does not have correlated queries, because the optimization process in our approach is guided mainly by query interaction.

Reactivity. We investigate the reactivity of both algorithms. Figure 11 shows the response time of the algorithms. By varying the threshold W, the runtime of SA increases dramatically, contrary to EQHDP's runtime, which remains very low compared to SA. This high computation cost is typical in such algorithms, contrary to our algorithm, which is much faster. This is due to the pruned search space phases and the interaction-based partitioning process.

Impact on Number of Split and Merge Operations. In the next experiment, we also check the speed of algorithms, but now in terms of moves and elementary operations of split and merge. Figure 12 shows the number of split/merge operations performed by the algorithms for different thresholds $\mathcal{W}$. As the SA is a non-deterministic algorithm, it is run several times, and different number of operations (split/merge) are given for each value of $\mathcal{W}$. On the other hand, the EQHDP is a deterministic algorithm, so that a unique solution is computed for each $\mathcal{W}$, and a fixed number of moves (split/merge) is performed. As shown in Figure 12, SA is very costly in terms of partitioning moves compared to the EQHDP for the same reasons as in previous experiments; i.e., our approach is interaction-based contrary to the SA, which explores a large search space randomly.

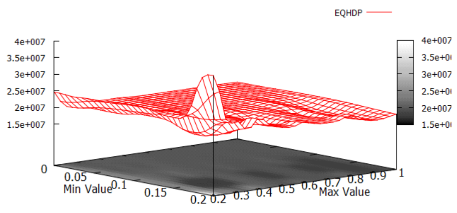


Fig. 13. Selectivity Factor intervals pour data partitioning

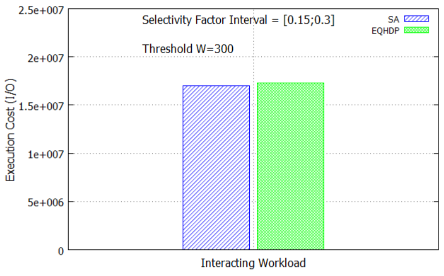


Fig. 14. Improving the EQHDP by defining selectivity factor interval

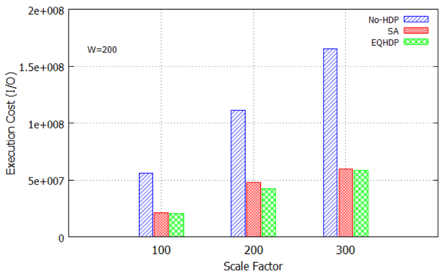


Fig. 15. Impact of Size Evolution on Query Performance

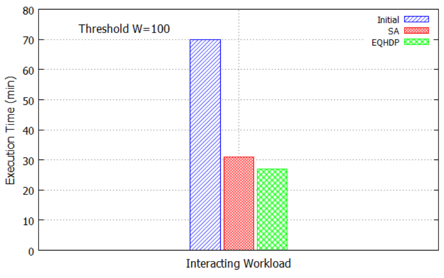


Fig. 16. Validating simulation results on Oracle11g

Selectivity Factor. Some relevant partitions for the EQHDP may not be suitable for partitioning process because of their selectivity factors which can be either too high or too low. Such selectivity factors generate unbalanced partitions and may penalize many queries. To avoid these partitions, we extend our algorithm by a new pruning criterion that consists of a selectivity factor interval.

We study the suitable selectivity factors for our workload as shown in Figure 13. In this experiment, we vary the minimal and maximal values of selectivity. Indeed, the highest (≈ 1) and lowest (≈ 0) selectivity penalize the workload. However, the best optimization values are found for the minimal value 0.15 and maximal value 0.30. This means that the sub-domains with selectivity factors in $[0.15; 0.30]$ are the most suitable ones for the optimization process. We compare the improved EQHDP algorithm with SA in the experiment illustrated by Figure 14, which shows that the yield of EQHDP is improved by this criterion even with higher Threshold ($\mathcal{W} = 300$) to be closer to SA performance.

Scale-up Analysis. In the final experiment, we study the impact of the database volume on the performance of optimization algorithms, as depicted in Figure 15. Varying the scale factor of the SSB allows to increase the database volume to 100 GB, 200 GB and 300 GB from a scale factor of 100, 200 and 300, respectively. When the database is in constant growth, the EQHDP algorithm remains more efficient than the SA for a Threshold of 200. This proves the effectiveness of our approach in the context of very large databases.

6.4 Validation

In order to validate our simulation results, we deploy both partitioning schemas obtained by SA and EQHDP with the same data set used in the simulations. Similarly to the theoretical results, Figure 16 shows the efficiency of our EQHDP approach compared with SA. From our experiments, we conclude that our approach remains considerably faster than SA especially when the threshold W is larger, which increases the computation cost of SA significantly. Additionally, our algorithm is deterministic, self-encoding, fast and capable to reach a high efficiency level of optimization based on query interaction. The algorithm is more efficient in highly correlated workloads, which is usually the case in a data warehouse environment.

7 Conclusion

The exploration of very extremely large databases is performed by the use of complex queries, involving joins and aggregations. These queries usually interact with each other, because the end users (scientists and decision makers) usually focus on identical parts of the databases. To optimize these queries, we propose the use of $\mathcal{HDP}$ because it is a pre-condition for designing several types of databases: centralized, parallel, distributed, cloud. In this paper, we motivate the consideration of query interaction in selecting $\mathcal{HDP}$ schemes for relational data warehouses. This interaction is handled by the use of incremental encoding of any horizontal partitioning schema. We contribute an algorithm for $\mathcal{HDP}$ that considers our encoding data structure. Our approach promotes the more sharable nodes of a unified query graph. Contrary to other partitioning algorithms, where all queries have the same probability to partition the database, our algorithm exploits the property of the big data and cloud environments, where queries have a high interaction with other queries. Another dimension is that our algorithm discards queries involving selection predicates either with high or low selectivity factors. We compare our algorithm with the simulated annealing algorithm, which is known as an efficient solution for $\mathcal{HDP}$. The results of our algorithms in terms of query processing cost and execution time are encouraging.

Currently, we are working on deploying our proposal in a cloud platform in considering the data placement problem in private and public clouds.

References

1. Ahmad, M., Aboulnaga, A., Babu, S., Munagala, K.: Interaction-aware scheduling of report-generation workloads. VLDB Journal 20(4), 589–615 (2011)
2. Bellatreche, L., Boukhalfa, K., Richard, P.: Referential horizontal partitioning selection problem in data warehouses: Hardness study and selection algorithms. International Journal of Data Warehousing and Mining 5(4), 1–23 (2009)
3. Ceri, S., Negri, M., Pelagatti, G.: Horizontal data partitioning in database design. In: SIGMOD, pp. 128–136. ACM (1982)
4. Curino, C., Jones, E.P.C., Popa, R.A., Malviya, N., Wu, E., Madden, S., Balakrishnan, H., Zeldovich, N.: Relational cloud: a database service for the cloud. In: CIDR, pp. 235–240 (2011)
5. Curino, C., Zhang, Y., Jones, E.P.C., Madden, S.: Schism: a workload-driven approach to database replication and partitioning. PVLDB 3(1), 48–57 (2010)
6. Galindo-Legaria, C.A., Grabs, T., Gukal, S., Herbert, S., Surna, A., Wang, S., Yu, W., Zabback, P., Zhang, S.: Optimizing star join queries for data warehousing in microsoft sql server. In: ICDE, pp. 1190–1199. IEEE (2008)
7. Ge, X., Yao, B., Guo, M., Xu, C.: Lsshare: An efficient multiple query optimization system in the cloud. To appears in DEXA (2013)
8. Le, W., Kementsietsidis, A., Duan, S., Li, F.: Scalable multi-query optimization for sparql. In: ICDE, pp. 666–677. IEEE (2012)
9. Mahboubi, H., Darmont, J.: Enhancing xml data warehouse query performance by fragmentation. In: SAC, pp. 1555–1562. ACM (2009)
10. O'Gorman, K., Agrawal, D., El Abbadi, A.: Multiple query optimization by cache-aware middleware using query teamwork. In: Proceedings of the International Conference on Data Engineering (ICDE), p. 274 (2002)
11. Özsu, M.T., Valduriez, P.: Principles of Distributed Database Systems, 2nd edn. Prentice Hall (1999)
12. Papadomanolakis, S., Ailamaki, A.: Autopart: Automating schema design for large scientific databases using data partitioning. In: SSDBM, pp. 383–392. IEEE (2004)
13. Sanjay, A., Narasayya, V.R., Yang, B.: Integrating vertical and horizontal partitioning into automated physical database design. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, pp. 359–370 (2004)

14. Sellis, T.K.: Multiple-query optimization. ACM Transactions on Database Systems 13(1), 23–52 (1988)
15. Oracle Data Sheet: Oracle partitioning. White Paper (2007), `http://www.oracle.com/technology/products/bi/db/11g/`
16. Stöhr, T., Märtens, H., Rahm, E.: Multi-dimensional database allocation for parallel data warehouses. In: VLDB, pp. 273–284. Morgan Kaufmann Publishers Inc. (2000)
17. Thusoo, A., Sarma, J.S., Jain, N., Shao, Z., Chakka, P., Zhang, N., Anthony, S., Liu, H., Murthy, R.: Hive - a petabyte scale data warehouse using hadoop. In: ICDE, pp. 996–1005. IEEE (2010)
18. Tzoumas, K., Deshpande, A., Jensen, C.S.: Sharing-aware horizontal partitioning for exploiting correlations during query processing. PVLDB 3(1), 542–553 (2010)
19. Yang, J., Karlapalem, K., Li, Q.: Algorithms for materialized view design in data warehousing environment. In: Proceedings of the International Conference on Very Large Databases, pp. 136–145 (August 1997)

Multi-tenant Network Monitoring Based on Software Defined Networking

Aryan TaheriMonfared and Chunming Rong

Department of Electrical Engineering and Computer Science
University of Stavanger, Norway
{aryan.taherimonfared,chunming.rong}@uis.no

Abstract. In any cloud service model multiple stakeholders are involved with different roles in the service provider-customer relationship. One service provider can use other services at the same time. It is vital to distinguish between stakeholders activities such as their communication in the network. Thus, there should be a monitoring system, which knows about stakeholders, their characteristics, and provisioned resources by them at any point of time. In most cases, traditional monitoring solutions does not have the capability to understand the difference between involved parties, while they're orchestrated to achieve a desired result (e.g. delivering a service for the end user).

In this study an improved architecture of the networking service for a cloud platform is introduced. Software Defined Networking (SDN), Network Virtualization, and traditional methods are adapted for gathering evidence and auditing activities on a per-tenant basis. Four approaches are investigated for monitoring and identifying tenants' traffic in an infrastructure. They are discussed and implemented to fulfil the cloud's network monitoring requirements. The focus is on the effectiveness of software defined networking for monitoring tenants' virtual networks.

Keywords: Software Defined Networking, Network Virtualization, OpenFlow, Network Monitoring, Multi-Tenant.

1 Introduction

In a virtualized environment, physical network resources can provide several virtual networks (VN). Traditional Internet Service Providers (ISPs) have two major roles, consisting of: physical infrastructure management and end-to-end service delivery. Network virtualization is realized by separating these roles and assigning them to different entities. In such an environment, an Infrastructure Provider (InP) is responsible for the physical infrastructure management, and a Service Provide (SP) is responsible for delivering the end-to-end network service [1]. Functionality decoupling leads to co-existence of multiple virtual networks which can be created by different SPs. These virtual networks are isolated and may span multiple InPs [2].

Network virtualization can be seen as a utility for studying new networking architecture or an important attribute of the new architecture [3]. The main idea behind network virtualization can be implemented using multiple technologies [1], such as: Virtual Local Area Network (VLAN) [4], Virtual Private Network (VPN) [5], active and programmable networks [6], overlay networks, and Software Defined Networking (SDN) [7]. SDN is a platform for network virtualization [8] that abstracts the hardware from the programming interfaces [9].

In the Infrastructure as a Service (IaaS) model, a customer has a set of virtual machine instances distributed over geographical regions. Instances' connectivity can be realized using Network Virtualization. A proper network monitoring solution must distinguish between customers' activities in the infrastructure's network. This is challenging due to the inherited properties of virtualization (e.g. utilizing shared pool of resources). Observation points should be placed in multiple layers with the functionality to monitor in different granularities. This study investigates applicability of existing monitoring techniques to the new computing model. Traditional techniques are improved using a network controller (e.g. an OpenFlow controller), which provides a unified view of networking substrates as well as the information about provisioned resources by each tenant.

1.1 Related Works

There has been a few efforts on the monitoring aspect of SDN, either using SDN for monitoring an infrastructure, or evaluating new characteristics of SDN enabled networks. Jose et al. [10] propose a low overhead measurement framework for commodity network switches that support OpenFlow. The solution is tuned to identify large traffic aggregates. Yu et al. [11] designed a push-based monitoring system using control messages sent to the OpenFlow controller by switches. It estimates links utilization in a passive way by analysing the stream of PacketIn and FlowRemoved messages. Ballard et al. [12] introduce OpenSAFE as a reliable and high performance approach of routing traffic for security monitoring purposes. In addition, a language (ALARMS [12]) is defined for management of network monitoring equipments. Braga et al. [13] presented a low overhead detection method for DDoS flooding attacks, which uses self organizing maps. The method periodically polls flow records from all OpenFlow enabled switches, and extract DDoS related features. Tootoonchian et al. [14] designed an OpenTM for traffic matrix (TM) estimation using capabilities in OpenFlow switches. TMs are crucial for anomaly detection in a network. Based on the routing information provided by the OpenFlow controller, OpenTM chooses a set of switches for querying. An efficient querying strategy lowers the overhead in network equipment.

The rest of the paper is structured as follows: Section 2 explains a common cloud platform architecture. Section 3 gives an overview of the network monitoring in a cloud platform with visibility into tenants' activities. Section 4 discusses challenges of designing such an awareness and monitoring system, as well as shortcomings of existing techniques. Then, Section 5 proposes four techniques to overcome challenges and shortcomings. Finally, Section 7 concludes the paper and compares proposed techniques.

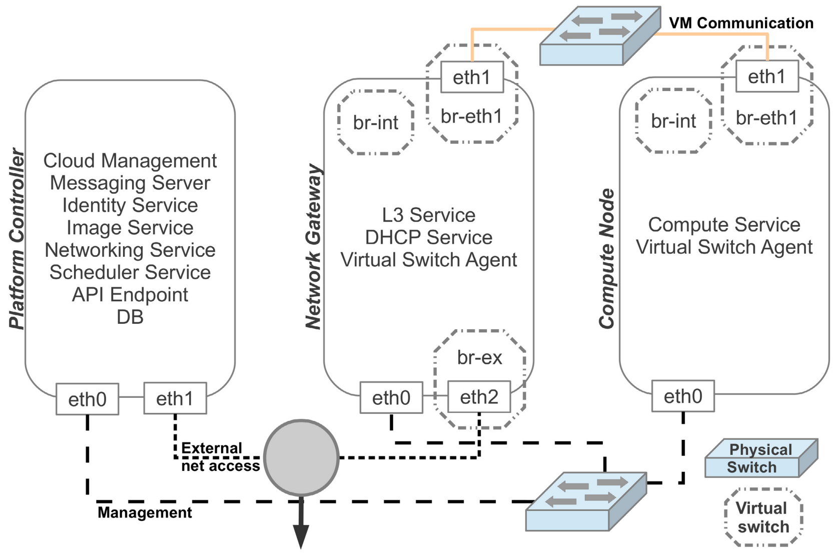


Fig. 1. Abstract Architecture

2 Abstract Cloud Platform Architecture

In the following, an abstract architecture of a cloud environment is explained. The architecture is the basis for our testbed. Later, the architecture is improved by delivering the networking service using SDN.

Three types of nodes are introduced in the architecture (Figure 1), comprising: Platform Controller, Network Gateway, and Compute Node. The platform controller runs core services, such as: inter-module message passing, authentication and authorization, virtual machine image management, basic networking service, job scheduling, and database service. Although core services are centralized in a single node, each service can be replicated on multiple nodes. The network gateway provides core networking services, including: DHCP service, L3 connectivity to the external network. The compute node hosts virtual machine instances and handles their network connectivity through a dedicated virtual switch.

From the networking perspective, four types of logical network are available that serves diverse requirements: infrastructure management, VM's communications, API access, and VMs access to the external networks.

Figure 2 gives more details about the network gateway and the compute node. The compute node has two and the network gateway has three virtual switches. Virtual switches are responsible for inter-VM connectivity. For the sake of simplicity one tenant (i.e. *Tenant A*) in a single compute node is depicted. Each tenant in the cloud platform has a dedicated virtual domain in the network gateway. The domain consists of virtual routers and networking services required for VMs connectivity and operation.

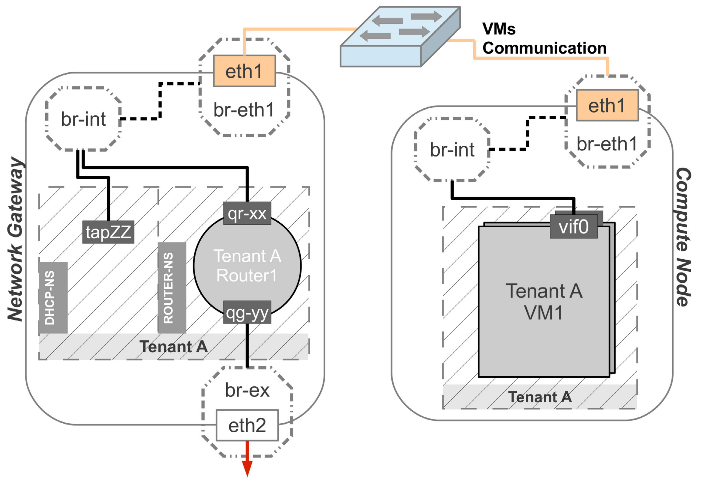


Fig. 2. Architecture Networking Details

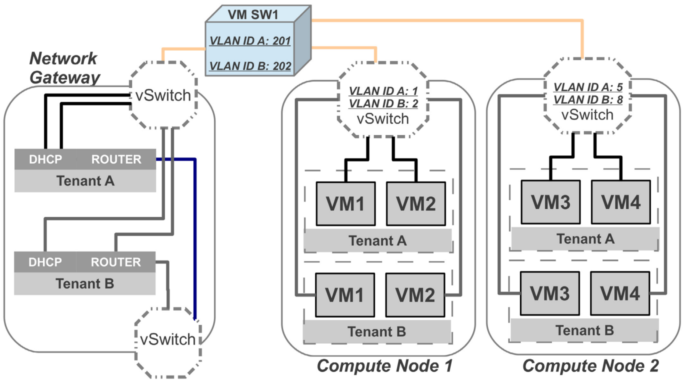


Fig. 3. VLAN Configuration Details

As depicted in Figure 3, *Tenant A* has several VMs distributed over multiple compute nodes. It has dedicated VLAN IDs (VIDs) on the physical switch (i.e. VM Switch 1), and virtual switches. VIDs on the physical and virtual switches may not be identical, since they're in different domains. The virtual switch in the compute node is responsible for translating VIDs between the physical and virtual domain. The mapping information is provided by the controller node, and is dependent on the tenant. All VMs, belonging to a particular tenant, are in the same broadcast domain, although they might be in multiple compute nodes.

In the testbed in Figure 1, the OpenStack Compute project (Nova[1]) provides the compute virtualization platform, and the networking project (Quantum[2])

[1] http://www.openstack.org/software/openstack-compute/
[2] http://www.openstack.org/software/openstack-networking/

is responsible for the networking services. Virtual switches are Open vSwitch[3] instances controlled by the Quantum server and they support the OpenFlow standard [15]. This feature is enabled to build a unified view of our network and fulfil the requirements of our monitoring solution (5.2). In addition, virtual switches are capable of exporting NetFlow[16], and sFlow[17] data, which are used for maintaining network awareness.

3 Tenant Based Network Monitoring Overview

In a distributed environment such as a cloud several stakeholders are involved. Stakeholders can be both service providers and consumers at the same time. Complexity of these interactions and the new characteristics of the networking architecture hinder the adaptation of traditional monitoring approaches. The following section explains our approach for monitoring a networking environment that supports network virtualization.

Monitoring can be done in different layers. By separating the layers based on the provider type, there will be three layers: service provider layer, platform provider layer, and infrastructure provider layer. There is no clear cut between these types and a single provider can cover several responsibilities.

– **The Infrastructure provider** is responsible for providing underlying substrates, such as physical compute nodes, networking equipment, cooling facilities, etc. (e.g. a data center provider). These components might be heterogeneous and distributed over a variety of geographic locations, under multiple jurisdictions.
– **The Platform provider** delivers functionalities which utilizes underlying components. It builds a unified view of distributed substrates for the upper layer entities. As an example a platform provider is responsible for deployment, maintenance, and operation of a cloud platform (e.g. OpenStack[4]), or a network virtualization platform (e.g. an SDN environment using standards such as OpenFlow [15])
– **The Service provider** uses functionalities exposed by the platform provider to build a variety of services. The provider can deliver one or more services from the cloud service models (e.g. Infrastructure as a Service (IaaS), Platform as a Service (PaaS), Software as a Service (SaaS) [18]).

However, in a cloud environment where several stakeholders are involved, fine-grained monitoring capability is crucial. It should be possible to monitor all resources used by a specific tenant (e.g. service customer). As an example, in a security incident the responsible Computer Emergency Response Team (CERT) should be able to monitor resources which are or have been provisioned by a tenant. Switches, routers, and interfaces (either physical or virtual) are considered to be resources, and a proper set of observation points should be identified for fine-grained monitoring.

[3] `http://openvswitch.org/`
[4] `http://www.openstack.org`

This study concentrates on the Infrastructure as a Service model of Cloud. Therefore, network monitoring approaches for auditing low level resources (e.g. virtual machines, block storages) are investigated.

4 Tenant Based Network Monitoring Challenges

New generations of cyber-attacks use sophisticated communication methods. The communication period is short and the traffic may not be significant. As an example Stuxnet uses five methods to propagate itself: peer-to-peer communication, infecting WinCC machine, network shares, MS10-061 Printer Spooler, and MS08-067 Windows Server Service [19]. None of these methods employ heavy scanning or long period communication. Thus, it is crucial to have a complete view of flows, and active services. Berthier et al. [20] proposes a solution for maintaining such a view, and we will investigate its application in a virtualized environment.

NetFlow services [16] provide information on IP flows from network elements, such as routers and switches. Flow data are sent from data networks to collectors. The data has detailed information about resource usage and can be helpful for obtaining network awareness. According to [16], a flow is considered to be unidirectional and bidirectional flows (Biflows) is a sequence of packets that pass through a network in both directions between two endpoints [21]. Thus, Biflows are more informative for security purposes. Although it is possible to determine Biflows in the collector using NetFlow data, it's more efficient to have it as an extension in the protocol [21]. The IP Flow Information Export (IPFIX) protocol [22] exports Biflows, though it is not widely used in substrates and is not supported by Open vSwitch. Thus, analytic tools are developed for identifying Biflows.

NetFlow is considered to be the most suitable protocol because of its accuracy. Virtual switches are configured to export their NetFlow data to a collector for further analysis. Nfcapd, nfdump[5], nfsen[6], and NFSight [20] are used for the initial collection and processing of data.

Monitoring the virtual switch in each node provides better visibility of the VMs communication. There are two virtual switches in each compute node, `br-eth1` and `br-int`. `br-eth1` is our observation point, since it is connected to the Top of the Rack (ToR) switch and handles all VMs' traffic inside the physical machine. On the network gateway, three virtual switches are configured, and two of them are monitored, `br-eth1`, and `br-ex` (external network access for VMs).

As depicted in Figure 4, the aggregated traffic on a virtual switch is visible. However, we can not differentiate tenants' traffic yet. Since tenants may have used overlapping IP addresses, traffic classification based on network addresses is not sufficient. Based on the current architecture, tenants can be distinguished from each other using their VIDs, which have been assigned by virtual and physical switches. These IDs are not essentially identical among switches. Moreover,

[5] `http://nfdump.sourceforge.net/`
[6] `http://nfsen.sourceforge.net/`

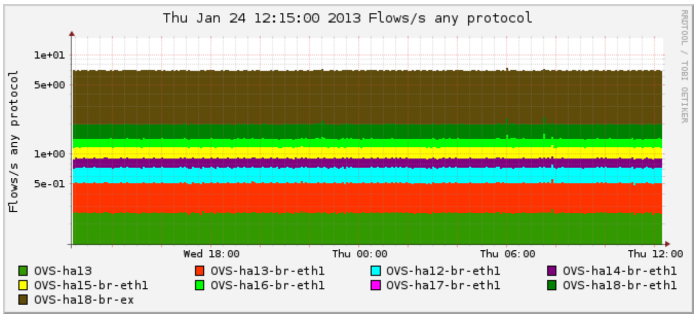


Fig. 4. Network traffic observed by a monitoring tool without the knowledge about tenants' co-existence

this technique is useful when VLAN tagging has been employed for tenants' isolation. VLAN tagging may not be implemented everywhere, due to its disadvantages (e.g. VLAN table scalability) for an environment with a large number of customers. For instance, the FloodLight OpenFlow controller does not use VLAN tagging for creating isolated logical layer 2 networks.

The following part summarize challenges of network awareness in a cloud platform, when multiple tenants coexist.

Elasticity and Dynamicity of The Environment: Tenants provision resources on demand and release them when there is no more need. Provisioned resources can scale up or down to meet the requirements, and can be migrated to other physical substrates or regions. Scalability, elasticity, and migration of virtual resources in the cloud, make monitoring much harder. A tenant is not bounded to a particular region or hardware. Therefore, the monitoring solution must be able to provide the tenant based statistics, independent of the physical location.

Complex Stakeholder Relations: In the new computing model, several stakeholders are involved and their interactions are complicated. Traditional monitoring solutions are not adapted to this model, and they can not logically distinguish between tenants in a cloud environment.

Incident Handling and Network Forensics: In an incident, the forensic team asks for relevant information. If the incident happens in a tenant domain, only data related to that specific tenant should be exposed to the forensic team. Proper clustering and anonymization of data become crucial, due to the privacy and security of other tenants.

Reliable Monitoring and Non-Repudiability: A malicious tenant may try to forge its networking specifications. The motivation can be concealing its own identity or impersonating other tenants. Monitoring approaches should not solely rely on the information provided by tenants, but provide mechanisms for assessing the originality of data. For instance, IP addresses can be spoofed by a tenant, however, VIDs are assigned by switches out of tenants' domains.

Hardware and Software Limitations: Existing hardware and software may not fully support new protocols and standards (e.g. NetFlow v9 and OpenFlow are not supported widely). Proposed techniques for the new model should not rely on standards or protocols, which are not mature yet.

Invisibility of tenants' internal communications: Instances in a tenant communicate with each other or with external entities (e.g. other tenants, off-premises entities). Observation points must be chosen such that they cover all types of traffic.

Tenants' Interests for Monitoring Services: Some companies are interested in using external services and moving their resources off-premises. A big obstacle on the migration is the lack of visibility and control over resources. Offering tenant based monitoring services can relax the issue, and satisfies more customers.

5 Tenant Based Network Monitoring Solutions

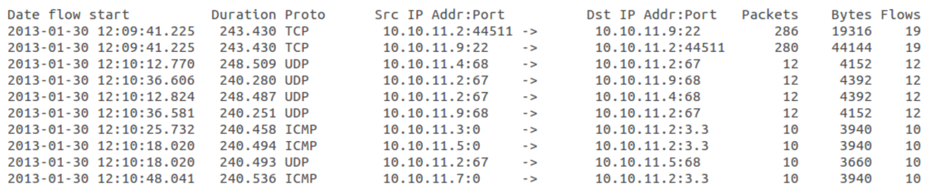
There should be multiple techniques to fulfil requirements of a heterogeneous infrastructure. Four approaches in two categories are proposed for a tenant's network monitoring in a cloud infrastructure:

- Non-OpenFlow enabled components
  - Tenants monitoring using IP datagram header.
  - Tenants monitoring using data link frame header.
  - Tenants monitoring using virtual components in the network gateway.
- OpenFlow enabled components

Following sections discuss each approach and the advantages and disadvantages.

5.1 Non-OpenFlow Enabled Components

SDN is in its early stages, and OpenFlow capable components are the realization tools. Thus, we can not assume that all equipments in an infrastructure will support the standard in a near future. Three techniques are explained for answering the network monitoring demands. Although the technology behind these techniques are not new, they should be adapted to the new model.



```
Date flow start          Duration Proto      Src IP Addr:Port          Dst IP Addr:Port   Packets    Bytes Flows
2013-01-30 12:09:41.225   243.430 TCP         10.10.11.2:44511 ->       10.10.11.9:22          286    19316    19
2013-01-30 12:09:41.225   243.430 TCP         10.10.11.9:22    ->       10.10.11.2:44511       280    44144    19
2013-01-30 12:10:12.770   248.509 UDP         10.10.11.4:68    ->       10.10.11.2:67           12     4152    12
2013-01-30 12:10:36.606   240.280 UDP         10.10.11.2:67    ->       10.10.11.9:68           12     4392    12
2013-01-30 12:10:12.824   248.487 UDP         10.10.11.2:67    ->       10.10.11.4:68           12     4392    12
2013-01-30 12:10:36.581   240.251 UDP         10.10.11.9:68    ->       10.10.11.2:67           12     4152    12
2013-01-30 12:10:25.732   240.458 ICMP        10.10.11.3:0     ->       10.10.11.2:3.3          10     3940    10
2013-01-30 12:10:18.020   240.494 ICMP        10.10.11.5:0     ->       10.10.11.2:3.3          10     3940    10
2013-01-30 12:10:18.020   240.493 UDP         10.10.11.2:67    ->       10.10.11.5:68           10     3660    10
2013-01-30 12:10:48.041   240.536 ICMP        10.10.11.7:0     ->       10.10.11.2:3.3          10     3940    10
```

Fig. 5. Tenants monitoring using IP datagram header fields

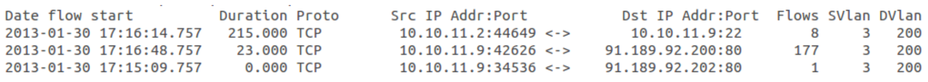
Tenants Monitoring Using IP Datagram Header. Each tenant in a cloud platform can have several networks and subnets. The tenant's traffic can be identified using the IP addresses assigned to its subnets. The virtual switch (i.e. Open vSwitch) supports both sFlow and NetFlow v5 monitoring protocols. Most physical switches supports at least one of these protocols as well. Given that NetFlow is supported by all switches in an infrastructure, we can collect raw data and extract tenant's statistics from them.

Tenant's networks and subnets list can be retrieved from the networking service (e.g. OpenStack Quantum). Assigned IP addresses to the tenant is in the subnets information. The information can be used for processing the monitoring data that are collected from observation points. The analysis output shows the statistics for all instances which belongs to a particular tenant. For illustration, *Tenant A* is attached to two subnets, and we are interested in the subnet 10.10.11.0/24. Figure 5 shows *Tenant A*'s flows in all switches.

The implementation is simple, and requires several queries to the networking service and the monitoring data collector. Despite its simplicity, it's quite useful as it will preserve all capabilities provided by sFlow or NetFlow, as well as the visibility into tenants' traffic and behaviour.

However, the statistics may not be accurate or reliable. Because tenants may have overlapping IP address or a nefarious tenant can forge IP addresses. Thus, a tenant can impersonate another one during an attack. Auditing mechanisms fail to distinguish between the attacker and the impersonated tenant. It leads to accountability issues and contradicts non-repudiability principle. The same scenario can be imagined for billing issues, when tenants are charged according to their traffic. Overlapping IP addresses lead to inconsistent billing information and traffic measurement. So, the solution should be extended to cover these shortcomings.

Tenant Monitoring Using Data Link Frame Header. Monitoring tenants' traffic using VLAN information is more robust. VLAN tags are located in the data link layer frame header, and tenants can not forge them. Since, the virtual switch maintains a binding between instances in a particular tenant and the VID. In addition, using data link layer frame information makes the solution independent of network layer protocols. Thus, it will not be limited to IP.



```
Date flow start          Duration Proto      Src IP Addr:Port          Dst IP Addr:Port  Flows SVlan DVlan
2013-01-30 17:16:14.757   215.000 TCP        10.10.11.2:44649 <->       10.10.11.9:22        8     3   200
2013-01-30 17:16:48.757    23.000 TCP        10.10.11.9:42626 <->    91.189.92.200:80      177     3   200
2013-01-30 17:15:09.757     0.000 TCP        10.10.11.9:34536 <->    91.189.92.202:80        1     3   200
```

Fig. 6. Tenant monitoring using data link frame header fields

Table 1. NetFlow version 5 Flow Record Format [23]

Bytes	Contents
0-3	srcaddr
4-7	dstaddr
8-11	nexthop
12-13	input
14-15	output
16-19	dPkts
20-23	dOctets
24-27	First
28-31	Last
32-33	srcport
34-35	dstport
36	pad1
37	tcp_flags
38	prot
39	tos
40-41	src_as
42-43	dst_as
44	src_mask
45	dst_mask
46-47	pad2

As it was explained earlier, the virtual switch supports sFlow and NetFlow v5. NetFlow v5 is not suitable, as it doesn't deliver VLAN information, as shown in Table 1. For this approach sFlow or NetFlow v9/IPFIX must be used.

The networking service stores the mapping of tenants and physical switches VIDs, but tenants' VIDs in virtual switches are not known to it. An agent in each compute node should expose the VID mapping (i.e. tenant : VID : virtual switch ID). Then, the mapping can be used for processing sFlow monitoring data. The implementation is not trivial, because tenants' VIDs may not be identical among switches. First, we identify tenants' VIDs in each switch. Then, a query is created for aggregating the statistics from all virtual switches. As an example, *Tenant A* is using VID 3 in the virtual switch OVS-ha13 and VID 200 on the physical switch (Figure 6).

Tenant Monitoring Using Virtual Components in the Network Gateway. According to the architecture, tenants have their own dedicated network name-spaces in the network gateway. Tenant's virtual routers and interfaces are visible in the its name-space. Components in the tenant's name-space are responsible for delivering networking services such as DHCP and layer 3 connectivity.

These components (in the hatched area inside the network gateway in Figure 2) can be considered as observation points (i.e. `Route1, tapZZ, qr-xx, qg-yy`). But only a part of the tenant traffic is passing through these points, including: DHCP traffic, ingress/egress traffic from/to tenant's network. In other words, internal communication of instances inside a tenant is not observable. Therefore, it has several disadvantages: *a)* monitoring these points will not provide complete visibility of the tenant's traffic, *b)* tenant's traffic in each virtual switch is not distinguishable, *c)* deploying multiple redundant network gateways can disrupt the visibility and will require additional data correlations.

Although the approach has multiple drawbacks, its benefits make it interesting to be offered as a service. The implementation is simple and scales well. It has low overhead and the resource consumption is considerably less than other solutions. A tenant can use the service for monitoring communications with other networks. Networks can be either other tenants' networks or destinations outside the cloud platform. This is an efficient solution when all internal entities inside a tenant are trusted, or when there is no interest in the internal communication.

5.2 OpenFlow Enabled Components

The networking service is provided by the Quantum project, that means all virtual switches are configured by the Quantum. Although Open vSwitch supports OpenFlow, activating the controller in the virtual switch overwrites the previous configuration. Thus, we need to commit Quantum's command through an OpenFlow controller. In this architecture, the controller is connected to all capable switches and has a unified view of the network. Therefore, the monitoring node communicates with the controller to build per-tenant view of the network and generates monitoring information for each tenant, Figure 7.

There are multiple Quantum plug-ins providing this functionality, such as: Ryu OpenFlow Controller plugin[7], NEC OpenFlow plugin[8], Floodlight OpenFlow Controller[9] plugin. The most suitable controller, for our use-case in this period, is Floodlight controller. Floodlight has an application module called Virtual Network Filter (VNF) that provides virtualized layer 2 networks over a single layer 2 domain. VNF listens for PacketIn messages, and verifies the source and destination MAC addresses. If they belongs to the same virtual network, the packet will pass; otherwise it will be dropped. This is a simple approach with some limitations, such as: virtual networks can only be defined in a single physical layer 2 domain, and limited number of gateways in a tenant network[24].

The list of tenants are retrieved from the identity service (Figure 8 (1)), and their networks and subnets are loaded from the networking service (Figure 8 (4, 5)). Then, we need to find devices in the network and their attachment

[7] `http://www.osrg.net/ryu/`

[8] `http://wiki.openstack.org/Quantum-NEC-OpenFlow-Plugin`

[9] `http://floodlight.openflowhub.org/`

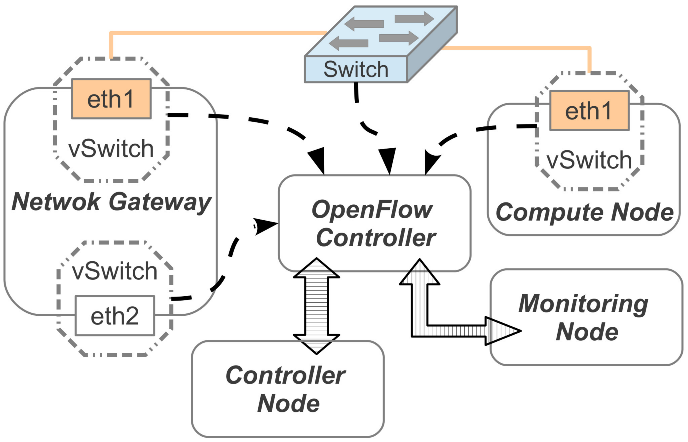


Fig. 7. OpenFlow enabled Lab Architecture

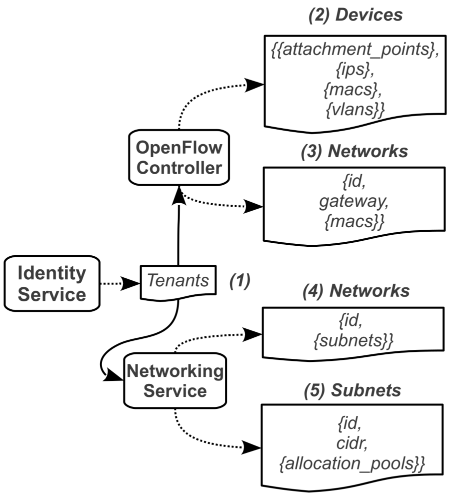


Fig. 8. Mapping switch ports to tenants

points (Figure 8 (2, 3)). Finally, the information required for monitoring tenants' network activity is ready. The information is in form of:

$$tenant_a : \{device_x : \{(mac_\alpha, ip_\alpha, port_\alpha, switch_\alpha)\}\}$$

Multiple types of report can be generated, including: *a)* a simple view of a tenant's traffic in the network by aggregating statistics of ports belonging to the same tenant, *b)* a sophisticated tenant's flow information by analysing NetFlow data using the information provided by the controller. The proof of concept code is developed and results are satisfactory.

6 Discussion

There are several shortcomings in the design of existing monitoring tools, such as: *a)* lack of tenant logic with complex stakeholder relationships, *b)* lack of flexibility for the new characteristics of the cloud model (e.g. elasticity, on-demand provisioning/release), and *c)* lack of robustness for a heterogeneous infrastructure. Four approaches in two category are discussed to cover requirements imposed by the model's criteria: Non-OpenFlow approaches, and the OpenFlow enabled approach. Non-OpenFlow enabled solutions use tenants' internet layer attributes (e.g. IPs), data link frame attributes (e.g. VLAN IDs), virtual devices in the Linux network stack namespace.

The first category uses core cloud platform APIs and standard monitoring data. Required parameters for building a tenant-based view are gathered, and filters are created. Parameters define networking attributes of a tenant, such as IP subnets, VIDs, and MAC addresses. Then, monitoring data are analysed using these parameters, and a tenant-based view of the network activities is created. The second category benefits from the OpenFlow controller's capabilities. The controller is responsible for the control plane of switches in the network, and it is aware of the tenant logic in the platform. Tenant awareness of the controller helps in developing a robust solution for recognizing and distinguishing tenants' networking behaviours.

Advantages, disadvantages, and requirements of of proposed solutions are briefly explained in Table 2.

7 Conclusion

In a multi-tenant model like the cloud computing, several stakeholders are involved. Distinguishing tenants' activities and provisioned resources, in the time domain, is the key factor for accountability, anomaly detection, as well as load-balancing. We have started with analysing the IaaS service model, and several stakeholders in the provider layers are identified. At each layer, monitoring can be performed with different granularities. Depending on the coarseness, the result exposes various characteristics of the system.

The architecture is powered by virtual switches and OpenFlow controllers. Two types of solutions are introduced, addressing a heterogeneous environment. The first type introduces methods for monitoring tenants' activities, when an OpenFlow controller is not available. In these methods, tenants' specifications are retrieved from the networking and identity service. Then, raw monitoring data are processed for building per-tenant traffic statistics. The second type benefits from an OpenFlow controller to build the per-tenant view. In this approach, the controller provides the unified view of the network, and is aware of the tenant logic.

An obstacle for deploying these mechanisms in a large scale production environment, is the storage and processing of large data sets. We will continue to enhance our monitoring solution by collecting flow information in a distributed

Table 2. Solution comparison (NS: Networking Service, IS: Identity Service)

Solution	**Additional Requirements**	**Advantages**	**Disadvantages**
Tenants monitoring using IP datagram header	* NetFlow data * NS Access * IS Access	+ Ease of implementation + Complete view of tenants IP activity + Per vSwitch statistics	- Prone to IP spoofing - Lack of accuracy
Tenants monitoring using data link frame header	* sFlow data * Monitoring agent in each vSwitch * NS access * IS access	+ Network protocol independence + Accuracy + Reliability against IP spoofing + Complete view of tenants network activities + Per vSwitch statistics	- Not working with NetFlow v5 - An agent should be deployed in each compute node
Tenants monitoring using virtual components in the network gateway	* NetFlow data * Monitoring agent in each network gateway * NS access * IS access	+ Simplicity + Complete view of ingress/egress traffic from/to the tenant network + Network protocol independence + Reliability against IP spoofing	- No per vSwitch statistics - No visibility to the internal traffic of a tenant - An agent should be deployed in each network gateway
Tenants monitoring using OpenFlow controller	* OpenFlow controller * NS OpenFlow agent * NetFlow data * NS access * IS access	+ Transparent integration + Network protocol independence + Complete view of tenants network activities + Accuracy + Flexibility + Reliability against IP spoofing	- Complicated implementation - Monitoring solution implementation depends on the controller

data-intensive processing framework. Thus, we will be able to divide a query task and execute it over a cluster of commodity servers. This leads to efficient statistical analysis of monitoring data. The result will be used for providing real-time feedback to the OpenFlow controller for updating networking decisions.

Acknowledgment. The authors would like to thank Martin Gilje Jaatun from SINTEF ICT, who provided valuable comments and assistance to the undertaking of this research.

References

1. Chowdhury, N.M.K., Boutaba, R.: A survey of network virtualization. Computer Networks 54(5), 862–876 (2010)
2. Chowdhury, N.M.K., Boutaba, R.: Network virtualization: state of the art and research challenges. IEEE Communications Magazine 47(7), 20–26 (2009)
3. Anderson, T., Peterson, L., Shenker, S., Turner, J.: Overcoming the internet impasse through virtualization. Computer 38(4), 34–41 (2005)

4. L.S. Committee: IEEE standard for local and metropolitan area networks virtual bridged local area networks. IEEE Std 802.1Q-2005 (Incorporates IEEE Std 802.1Q1998, IEEE Std 802.1u-2001, IEEE Std 802.1v-2001, and IEEE Std 802.1s-2002), 1–285 (2006)
5. Ferguson, P., Huston, G.: What is a VPN?. The Internet Protocol Journal 1(1) (1998)
6. Campbell, A.T., De Meer, H.G., Kounavis, M.E., Miki, K., Vicente, J.B., Villela, D.: A survey of programmable networks. ACM SIGCOMM Computer Communication Review 29(2), 7 (1999)
7. Lantz, B., Heller, B., McKeown, N.: A network in a laptop, pp. 1–6. ACM Press (2010)
8. Drutskoy, D., Keller, E., Rexford, J.: Scalable network virtualization in software-defined networks. IEEE Internet Computing, pp. 99, 1 (2012)
9. McKeown, N.: Software defined networking. In: IEEE InfoCom (2009)
10. Jose, L., Yu, M., Rexford, J.: Online measurement of large traffic aggregates on commodity switches. In: Proceedings of the 11th USENIX Conference on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services, Hot-ICE 2011, p. 13. USENIX Association, Berkeley (2011)
11. Yu, C., Lumezanu, C., Zhang, Y., Singh, V., Jiang, G., Madhyastha, H.: FlowSense: monitoring network utilization with zero measurement cost (2013)
12. Ballard, J.R., Rae, I., Akella, A.: Extensible and scalable network monitoring using OpenSAFE. In: Proceedings of the 2010 Internet Network Management Conference on Research on Enterprise Networking, INM/WREN 2010, p. 8. USENIX Association, Berkeley (2010)
13. Braga, R., Mota, E., Passito, A.: Lightweight DDoS flooding attack detection using NOX/OpenFlow, pp. 408–415. IEEE (October 2010)
14. Tootoonchian, A., Ghobadi, M., Ganjali, Y.: OpenTM: traffic matrix estimator for OpenFlow networks. In: Krishnamurthy, A., Plattner, B. (eds.) PAM 2010. LNCS, vol. 6032, pp. 201–210. Springer, Heidelberg (2010)
15. McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., Turner, J.: OpenFlow. ACM SIGCOMM Computer Communication Review 38(2), 69 (2008)
16. Claise, B.: Cisco Systems NetFlow Services Export Version 9. RFC 3954 (Informational) (October 2004)
17. Phaal, P., Lavine, M.: sFlow version 5. Technical report (July 2004)
18. Armbrust, M., Stoica, I., Zaharia, M., Fox, A., Griffith, R., Joseph, A.D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A.: A view of cloud computing. Communications of the ACM 53(4), 50 (2010)
19. Falliere, N., Murchu, L.O., Chien, E.: W32.Stuxnet dossier. Technical, Symantec, Version 1.4 (February 2011)
20. Berthier, R., Cukier, M., Hiltunen, M., Kormann, D., Vesonder, G., Sheleheda, D.: Nfsight: netflow-based network awareness tool. In: Proceedings of the 24th International Conference on Large Installation System Administration, LISA 2010, pp. 1–8. USENIX Association, Berkeley (2010)
21. Trammell, B., Boschi, E.: Bidirectional Flow Export Using IP Flow Information Export (IPFIX). RFC 5103 (Proposed Standard) (January 2008)
22. Claise, B.: Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of IP Traffic Flow Information. RFC 5101 (Proposed Standard) (January 2008)
23. Cisco Systems: NetFlow FlowCollector installation and user guide. Technical Report OL-2587-01 (1999)
24. Kanui, B.: FloodLight Virtual Network Filter (October 2012)

Federated Authorization for Software-as-a-Service Applications

Maarten Decat[1], Bert Lagaisse[1], Dimitri Van Landuyt[1], Bruno Crispo[2], and Wouter Joosen[1]

[1] iMinds-DistriNet, KU Leuven, 3001 Leuven, Belgium
`first.last@cs.kuleuven.be`

[2] Department of Information Engineering and Computer Science, University of Trento, Trento, Italy
`crispo@disi.unitn.it`

Abstract. Software-as-a-Service (SaaS) is a type of cloud computing in which a tenant rents access to a shared, typically web-based application hosted by a provider. Access control for SaaS should enable the tenant to control access to data that are located at the provider based on tenant-specific access control policies. To achieve this, state-of-practice SaaS applications provide application-specific access control configuration interfaces and as a result, the tenant policies are evaluated at the provider side. This approach does not support collaboration between provider-side and tenant-side access control infrastructures, thus scattering tenant access control management and forcing the tenant to disclose sensitive access control data. To address these issues, we describe the concept of federated authorization in which management and evaluation of the tenant policies is externalized from the SaaS application to the tenant. This centralizes tenant access control management and lowers the required trust in the provider. This paper presents a generic middleware architecture for federated authorization, describing required extensions to current policy languages and a distributed execution environment. Our evaluation explores the trade-off between performance and security and shows that federated authorization is a feasible and promising approach.

Keywords: Federation, authorization, access control, Software-as-a-Service.

1 Introduction

Software-as-a-Service or SaaS is a form of cloud computing in which the *tenant* rents access to a shared application hosted by the *provider* [21]. The tenant is an organization representing multiple end-users, who use the application through a thin client, typically a web browser. A SaaS application is used by multiple tenant organizations and each organization can be tenant for multiple SaaS applications. While traditional SaaS applications such as Google Apps (an office suite) and Salesforce (CRM) mainly target small enterprises, large enterprises

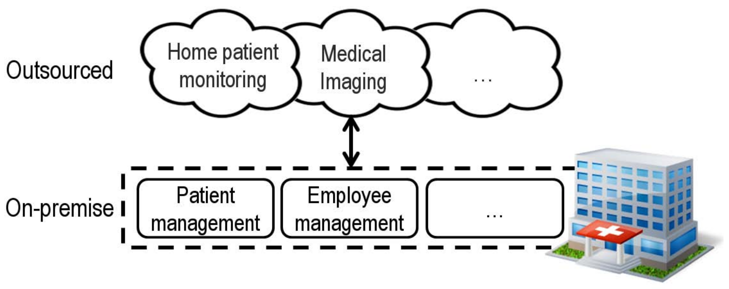


Fig. 1. Large organizations such as hospitals have started to adopt SaaS applications. These organizations often employ on-premise infrastructures for managing their users efficiently. Access control for SaaS should integrate with these on-premise infrastructures, which still poses challenges.

have recently started to adopt SaaS as well, for example in the domains of document processing [6], workforce management [6] or e-health [4,5]. This evolution stresses key challenges, such as the increased importance of security in general and access control in particular.

While the data in a SaaS application is hosted by the provider, the tenant should still be able to control access to it based on tenant-specific access control policies. While large organizations often employ on-premise access control infrastructures for managing their users centrally and efficiently (illustrated in Fig. 1), state-of-practice SaaS applications offer tenants an application-specific access control configuration interface. As a result, the tenant policies are stored and evaluated provider-side. This causes two major problems: (i) This approach fails to integrate with the on-premise infrastructures of the tenant and again distributes and scatters its access control management. Since a single organization can be tenant of multiple SaaS applications, this leads to large administrative overhead and eventually to inconsistencies and security holes. (ii) This approach forces the tenant to disclose to the provider all access control data required for evaluating its policies. While SaaS applications outsource specific functionality, the organization-wide policies of the tenant apply. These policies reason about data of the organization stored in the on-premise infrastructures, such as attributes of a patient or a physician. Although the tenant may trust the provider with the data in the application, it does not necessarily want to trust the provider with this sensitive on-premise data needed for access control. Moreover, regulatory requirements such as the European DPD [11] even forbid the tenant to share the data.

To address the problems identified above, we describe the notion of *federated authorization.* In analogy to federated authentication [1,2], federated authorization externalizes tenant policy evaluation from the provider to the tenant. This approach fully centralizes the access control management of the tenant and enables the tenant to enforce policies on the SaaS application without disclosing the sensitive data that are required to evaluate the policies.

This paper first introduces the concept of federated authorization for SaaS applications. The paper then presents a generic middleware architecture for federated authorization in which the XACML policy language is extended and a supporting distributed execution environment is defined. This architecture has three key features: (i) requesting an access control decision from the tenant, (ii) handling local and remote attributes and (iii) handling local and remote obligations. The paper then evaluates federated authorization in terms of performance based on a prototype of the middleware and finally discusses the trade-off between performance and security, showing that the notion of federated authorization is a feasible and promising approach. While similar approaches to federated authorization exist in related work (e.g., [7,8,19]) and its essence is used in practice in specific domains (e.g., 3-D Secure for internet payments [3]), this work focuses on a generic, policy-based middleware architecture. While this work originated from the domain of SaaS, we believe that federated authorization is an important enabler for all cross-organizational applications.

The remainder of this paper is structured as follows. Section 2 describes the context of this work, i.e., access control and federation techniques. Section 3 elaborates on the problem statement based on a realistic case study in the domain of e-health. Section 4 describes the concept of federated authorization and presents our supporting architecture and language extensions. Section 5 evaluates federated authorization in terms of performance based on our prototype and Section 6 discusses the results and the impact on security. Section 7 gives an overview of the related work and Section 8 concludes this paper.

2 Background

This work fits in the domain of federated access control, applied to SaaS applications. Since SaaS was introduced in the previous section, this section discusses access control and support for federation.

2.1 Access Control

Access control is an important part of application-level security that limits the *actions* which a *subject* (e.g., an end-user) can take on an *object* in the system (e.g., a file). The process of access control is generally divided into authentication and authorization: authentication confirms the stated identity of a subject and authorization confirms that the authenticated subject is allowed to do the desired action on the object by evaluating *access control policies*. Several models have been proposed for expressing these, of which attribute-based access control (ABAC, [16]) employs key-value properties called *attributes* of the subject (e.g., the subject id, a username or a role), the object (e.g., the object id, location or content) and the environment (e.g., the time, physical location or usage context).

The reference architecture for policy-based authorization middleware was defined by IETF and DMTF and refined by the XACML standard [22]. In the reference architecture (see Fig. 2), the policy decision point (PDP) makes the

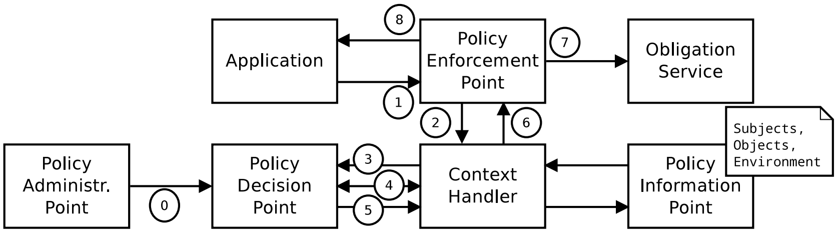


Fig. 2. The XACML reference architecture for policy-based authorization middleware [22]

actual authorization decision. The policy enforcement point (PEP) is integrated in the application (e.g., as an API or a reference monitor) and requests an access control decision from the PDP through the context handler. A decision request generally consists of information about the subject, the object, the action and the environment. The context handler gathers required attributes from one or more policy information points (PIPs, e.g., a database), which the PDP uses to evaluate the applicable policies loaded from the policy administration point (PAP). If needed, the PDP can request additional attributes from the context handler. Finally, the PDP returns its response to the PEP. Such a response consists of the access control decision itself (permit or deny) and possibly obligations. An obligation represents an action to be executed after the access control decision, e.g., updating an attribute such as the subject's usage quota. The PEP fulfills the obligations using the obligation service and enforces the PDP's decision.

2.2 Support for Federation

The XACML reference architecture is aimed at access control systems within a single organization. For applications in which multiple organizations are involved, *federation techniques* have been developed. A *federation* can be defined as an organizational structure in which multiple organizations have set up collaboration agreements. Each organization represents a separate domain in terms of security and administration and is bound to other domains by the means of (limited) trust. SaaS applications are an example of this. While a federation can in general comprise more than two organizations, this paper focuses on the provider and the tenant.

One federation technique is *federated authentication* [1,2]. For SaaS, federated authentication allows to externalize authentication from the provider to the tenant: the tenant authenticates the subject locally and afterward states his or her identity and attributes to the provider. The advantages are threefold: (i) It gives the tenant full control over the means of authentication. (ii) It gives the tenant full control over the access control data shared with the provider. (iii) It effectively centralizes user management at the tenant side, offering scalable user administration to the tenant.

Supported by federated authentication, two main strategies for authorization exist in state-of-the-art access control for SaaS: (i) Full provider-side authorization:

In this case, all access control data and policies are located at the provider side. The provider evaluates the policies based on local data and no federation techniques are used. (ii) Provider-side authorization with federated authentication: In this case, the access control data is centralized at the tenant side, but the policies are located at and evaluated by the provider and the tenant data is shared using federation techniques. In both strategies, only the provider evaluates access control policies.

3 Problem Elaboration and Illustration

This section presents the problem statement of this paper. First, Section 3.1 presents an illustrative case study in the domain of e-health. From this, we derive three key requirements and argue their relevance for SaaS in Section 3.2.

3.1 Case Study in e-Health

As stated in the introduction, large enterprises have started to adopt SaaS. An example of such an application inspired by a number of research projects [4,5] is a home patient monitoring system provided to hospitals as a service (see Fig. 3). The system allows patients to be monitored continuously after leaving the hospital, for example by a chest band or a wrist sensor. The measurements (the application data) are sent from the patients to the application back-end using a smart-phone as an intermediary device. The measurements are stored and processed by the provider: telemedicine operators continuously check the patient's status and the patient's physician at the hospital is notified in case of important evolutions. A patient's status can also be viewed by other physicians and nurses and by the patients themselves. The hospitals are the tenants of the application and each tenant represents multiple patients and physicians, i.e., the end-users of the application. Next to the monitoring system, the hospital employs other on-premise applications, e.g., for patient management, and other SaaS applications, e.g., for medical imaging.

Example policy. A typical example of an organization-wide policy employed by the hospital in this case study can be summarized as follows: "*a physician*

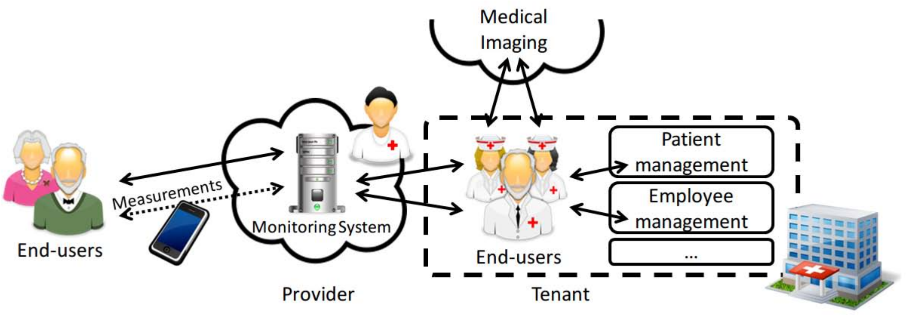


Fig. 3. The case study that inspired this work: a home patient monitoring system offered to hospitals as a SaaS application

can only view or alter patient data when currently treating the patient who owns the data and if that patient has given consent or when the data is relevant to his specialization or when the patient is in a life-threatening situation".

3.2 Access Control Requirements

It is clear that the provider wants to control access to the application. The provider constrains its own end-users, i.e., the telemedicine operators, and constrains its tenants, e.g., to ensure a tenant has enough credit to perform a certain action or is credited afterward. However, e-health applications are subject to regulatory requirements (e.g., the European DPD [11]) and even if the data is hosted by the provider, the hospital is still accountable for it. Therefore, the hospital has to be able to control access to the application as well and the hospital-wide policy presented above also applies to the SaaS application. In this work we take the point of view of the tenant and focus on the following three requirements:

Scalable administration. Hospitals can have a medical staff of thousands and treat even more patients each day, continuously producing large amounts of new application data. Manual security administration on this scale would incur too much administrative overhead and would quickly lead to inconsistencies and security holes. To avoid this, hospitals have extensive centralized security infrastructures at hand, e.g., organization-wide patient management systems. When employing SaaS applications such as the patient monitoring system, this degree of centralization should be upheld and security administration should remain centralized at the hospital.

Complex and large policies. Current e-health policies require detailed and extensive information. For example, the small policy presented above requires user roles, treating relationships, patient consent, resource content and more. Furthermore, the complexity of current policies makes it impossible to statically determine the required data up-front.

Sensitive access control data. Some of the access control data required for evaluating the policy presented above is sensitive in nature, such as the list of patients being treated by a physician, patient diseases or patient consent. While the hospital may trust the provider with the monitoring data in the application, it does not necessarily want to trust the provider with this sensitive access control data. Moreover, regulatory requirements such as the European DPD [11] even forbid the hospital to share this data.

As mentioned in Section 2.2, the hospital policies are evaluated by the provider in state-of-the-art access control for SaaS. As a consequence, the hospital policies are still distributed and fragmented over the multitude of applications it uses. Moreover, all tenant data required for evaluating the tenant policies has to be shared with the provider. Therefore, state-of-the-art federated access control does not fulfill the requirements stated above: it leads to limited administrative scalability and forces the tenant to disclose sensitive access control data.

4 Solution: Federated Authorization

We address the problems identified in the previous section by externalizing tenant policy evaluation from the provider to the tenant. This effectively centralizes tenant access control management and enables the tenant to enforce policies on the SaaS application without having to share the access control data needed for evaluating these. In analogy to federated authentication, we call this *federated authorization.*

Section 4.1 first describes the key features required for realizing federated authorization and their impact on the XACML reference architecture and current policy languages. Section 4.2 presents our generic architecture for federated authorization and finally Section 4.3 illustrates the required policy language extensions using the XACML policy language.

4.1 Key Features

With federated authorization, the provider evaluates its own policies while the evaluation of tenant policies is externalized from the provider to the tenant. With respect to the XACML reference architecture (see Section 2.1), federated authorization adds three key features: (i) requesting an access control decision from the tenant, (ii) handling local and remote attributes and (iii) handling local and remote obligations. For each of these new features, we determine what should be added to the XACML reference architecture and to current policy languages such as XACML [22], Ponder [12] or EPAL [10].

Requesting Tenant Access Control Decisions. As a first key feature, the provider is able to request an access control decision from the tenant concerning the SaaS application. Afterward, the provider combines the tenant response with its own policy results so that the requested action is only permitted if both parties permit it. Allowing the provider to request a tenant access control decision impacts both the architecture and the policy language.

Architecture. The tenant should provide a service to receive decision requests from the providers of the SaaS applications it employs. Such a request should identify the provider and should contain information about the subject, the object, the action and the environment, similar to a request from PEP to PDP. Using this information, the tenant determines and evaluates the applicable policies and returns its response. This response contains the decision itself (permit or deny) and possibly obligations, similar to a response from PDP to the PEP. Notice that the provider does not reference a particular tenant policy, but rather the tenant as a whole behaving like a PDP.

Policy Language. The policy language should allow the provider to refer to the access control decision service of the tenant and specify how the result should be processed, similar to on-premise policies.

Handling Local and Remote Attributes. As a second key feature, all required attributes are made available to the respective policies. As mentioned before, the provider reasons about its end-users (e.g., the telemedicine operators in the case study) and its tenants (e.g., the hospital) and the tenant reasons about its end-users (e.g., the hospital physicians and nurses). For the provider policies, the subjects are therefore its end-users and tenants, the objects are the application data and the environment is the SaaS application. Thus, all attributes required by the provider policies are available locally. However, for a tenant policy, the subjects are its end-users, the objects are the application data and the environment comprises both the SaaS application and the local infrastructure. Data about the tenant end-users and local infrastructure is stored in its on-premise applications, while the rest is hosted by the provider. Thus, the attributes required by the tenant policies are distributed over tenant and provider. Allowing the tenant to handle both local and remote attributes impacts both the architecture and the policy language.

Architecture. For evaluating the tenant policies tenant-side, the attributes of the objects in the SaaS application and the attributes of the provider-side environment have to be made available to the tenant. In addition, while attributes can be added to the initial request to the tenant, the complexity of current policies makes it impossible to statically determine the required attributes up-front. Therefore, the provider should provide a service to the tenant to dynamically fetch required attributes.

Policy Language. When evaluating the tenant policies, the context handler should know where to find the required attributes. Therefore, the policy language should allow to define the location of each attribute referenced in the policies.

Handling Local and Remote Obligations. As a third key feature, the tenant is able to handle both tenant-side obligations, e.g., for logging, and provider-side obligations, e.g., for updating the access control history of an object. Allowing the tenant to handle local and remote obligations impacts both the architecture and the policy language.

Architecture. The response from the tenant policy evaluation service should contain the obligations specified by the tenant which will be fulfilled by the provider. This was already mentioned before. The tenant response should not contain locally fulfilled obligations.

Policy Language. The policy language should allow the tenant to specify where obligations should be fulfilled: locally or remotely.

4.2 Generic Middleware Architecture for Federated Authorization

Based on the architectural requirements listed above, we now present a generic architecture for federated authorization. We first describe the decomposition of the architecture aligned to the XACML reference architecture presented in Section 2.1 and then illustrate the resulting access control flow.

Architecture Decomposition. Figure 4 shows the decomposition of the architecture. First of all, the provider hosts the SaaS application and therefore also the

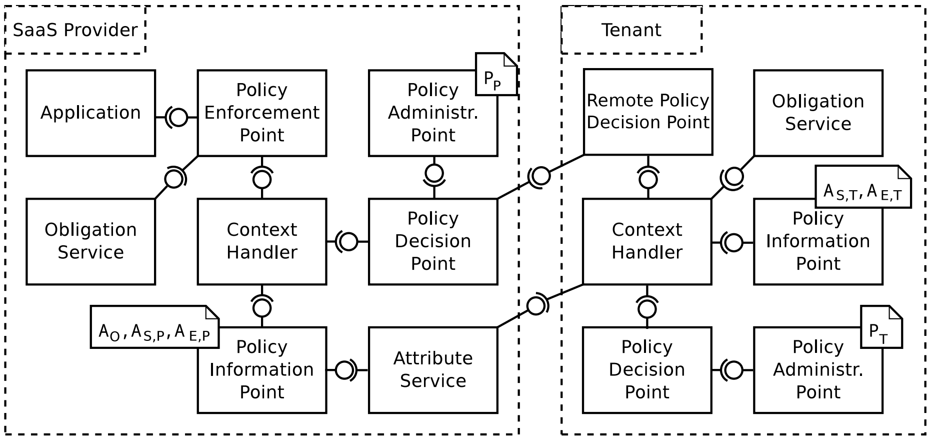


Fig. 4. The generic architecture for federated authorization. P_P and P_T are the provider and tenant policy sets respectively and A_O, $A_{S,P}$, $A_{E,P}$, $A_{S,T}$ and $A_{E,T}$ are as defined in Section 4.2.

PEP, no application components are located at the tenant side. Since both parties evaluate policies and process obligations, both have a PAP, a PDP, a context handler, one or more PIPs and an obligation service. The provider PIP contains the attributes of the objects in the SaaS application (A_O), the attributes of the provider policy subjects ($A_{S,P}$) and the attributes of the provider-side environment ($A_{E,P}$). The tenant PIP contains the attributes of the tenant policy subjects ($A_{S,T}$) and the attributes of the tenant-side environment ($A_{E,T}$). For handling decision requests, the tenant offers a Remote Policy Decision Point (RPDP) to the provider. For handling attribute requests, the provider offers an attribute service to the tenant. The provider PDP is extended with functionality to contact the RPDP and the tenant context handler is extended with functionality to contact the provider attribute service. To summarize, the resulting interface between provider and tenant consists of two services: the tenant policy evaluation service and the provider attribute service. Notice that the architecture still allows the tenant to use its infrastructure for on-premise applications as well, which is not shown in the figure.

Access Control Flow. Figure 5 illustrates the resulting access control flow. The presented flow starts after the provider and tenant policies are loaded from their respective PAPs and the end-user has been successfully authenticated. The remainder of the flow is as follows: (1) With each request an end-user makes to the application, the PEP constructs an access control request and forwards this to the local context handler. (2) The context handler collects statically known attributes from the local PIPs (only one is shown in Fig. 5), adds these to the request and sends it to the local PDP. (3) The PDP determines the applicable provider policies and evaluates them, thereby dynamically requesting attributes from the context handler. When the PDP encounters a remote policy reference,

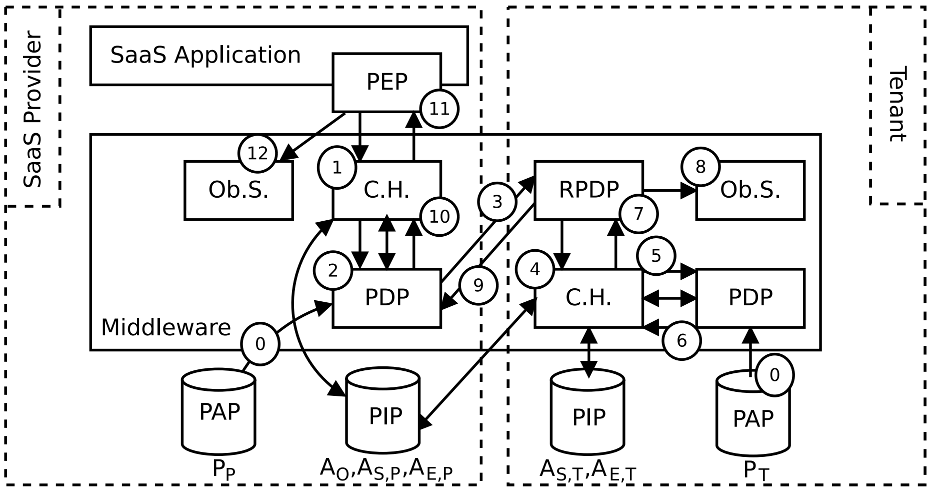


Fig. 5. The resulting access control flow for the generic architecture. C.H. is context handler, Ob.S. is obligation service, P_P and P_T are the provider and tenant policy set respectively, PEP, PDP, PIP and PAP are as defined in Section 2.1 and A_O, $A_{S,P}$, $A_{S,T}$, $A_{E,P}$ and $A_{E,T}$ are as defined in Section 4.2. For readability reasons, the provider attribute service is not shown explicitly.

it asks the context handler to determine how and where to contact the tenant. The PDP then constructs a decision request and sends it to the tenant RPDP. (4) From the tenant point of view, the RPDP acts similarly to a PEP and forwards the request to the tenant context handler. (5) The context handler collects statically known attributes from the local PIPs (only one is shown in Fig. 5), adds these to the request and forwards it to the PDP. (6) The PDP determines the applicable tenant policies and evaluates them, thereby dynamically requesting attributes from the context handler. The context handler fetches tenant attributes locally and contacts the provider attribute service for provider attributes. The tenant PDP eventually returns its response (i.e., decision and obligations) to the context handler. (7) The context handler returns the response to the RPDP. (8) The RPDP fulfills tenant obligations using the tenant obligation service and removes these from the response. (9) The RPDP returns the resulting response to the provider PDP. (10) The provider PDP combines the tenant decision with the result of the provider policies so that the request is only allowed if both parties allow it and returns the overall response to the provider context handler. (11) The provider context handler returns the response to the PEP. (12) The PEP fulfills the obligations using the provider obligation service and enforces the decision.

4.3 Extensions to Current Policy Languages

The three key enablers to support federated authorization identified in Section 4.1 all required policy language extensions. To illustrate the practical impact of these requirements, we have extended the XACML policy language v2 [22]. We opted for XACML because of its wide-spread use in both academia and industry and its active development. In this section we describe the extensions we introduced, their specifications are provided on-line[2]. We start by introducing XACML first.

The XACML policy language. In addition to the access control reference architecture, the XACML standard also defines a policy language [22], hereafter simply referred to as "XACML". XACML expresses policies using XML. Three main elements are defined: `<PolicySet>`, `<Policy>` and `<Rule>`. A policy set can contain multiple policies and a policy can contain multiple rules. A rule specifies an effect (permit or deny), attribute-based conditions for this to hold and obligations to fulfill with it. A policy combines the results of its rules using a rule-combining algorithm (e.g., deny overrides), a policy set combines the results of its policies using a similar policy-combining algorithm. Each of these elements also contains an attribute-based target specifying in which situations it applies.

Referencing remote policies. For referencing remote policies, the `<Remote-PolicyReference>` element is introduced. As mentioned before, the tenant behaves as a remote policy from the provider point of view. Therefor, evaluating a remote policy reference returns a policy evaluation result and the element can be part of a policy set, similarly to the `<Policy>` element. The `PolicyId` attribute specifies the id of the remote policy. Following the XACML design principles, it is left to the context handler to determine how and where to contact it. A remote policy reference can also contain a description, a target and obligations for local use.

Handling local and remote attributes. For differentiating between local and remote attributes, we follow the XACML design choice of having the context handler infer the location of an attribute based on its id instead of defining attribute properties declaratively. Thus, XACML is not extended for this requirement.

Handling local and remote obligations. For differentiating between tenant and provider as obligation targets, the `<Obligation>` element is extended with the optional `FulfillWhere` attribute. This attribute specifies whether the obligation should be fulfilled locally (with the tenant) or remotely (with the provider), local fulfillment being the default. The new attribute is only to be used in tenant policies. It is the responsibility of the tenant to remove local obligations from its response so that the provider can interpret all obligations as before.

5 Performance Evaluation

Following the description of the supporting middleware, this section evaluates the concept of federated authorization in terms performance based on a prototype of the middleware. The performance evaluation explores the impact of federation on the time it takes for the provider to reach an access control decision. Because federated authorization only affects the evaluation of the tenant policies, the provider policies are not taken into account.

Test Setup. The tests compare federated authorization against the two strategies for SaaS authorization in state-of-the-art mentioned in Section 2.2. Thus, three different cases of authorization are compared: full provider-side authorization, provider-side authorization with federated authentication and federated authorization. Notice that only federated authorization keeps sensitive access control data of the tenant confidential.

The tests employ policies which require 10, 20 and 30 attributes. Access control studies in a number of research projects [4,5,6] show that these numbers represent modest to large policies. Because multiple strategies for fetching attributes exist, ranging from one-at-a-time to combined requests, the tests also compare the two extreme strategies: fetching all required attributes separately and fetching all attributes at the same time as one multi-valued attribute. All attributes are fetched just-in-time and no attribute caching is used.

Prototype. The prototype of the middleware instantiates the architecture described in Section 4.2. The prototype extends the SunXACML policy evaluation engine[1] with the new `<RemotePolicyReference>` element. Attributes are stored in SQLite databases[1]. Cross-organization communication is realized out-of-band using SAML [1] and the SAML profile of XACML [20] over SOAP web-services[1] implemented on top of Apache Tomcat 7[1] using the Apache CXF services framework[1] and the OpenSAML Java library[1]. For similarity, both the provider and the tenant PDP are run on top of Tomcat. In order to focus on policy evaluation time, the prototype also omits channel encryption or authentication. The prototype (6KLOC) is publicly available[2].

The prototype contains four main components: (i) the provider PDP, (ii) the tenant PDP (iii) the provider attribute web service and (iv) the tenant attribute web service. Each of these components is run on a separate machine with 4GiB RAM and two cores of 2.40GHz running Ubuntu 12.04. Local databases are run on the same node as the services that use them. To simulate the distance between tenant and provider in a realistic SaaS setting, a fixed single-way network delay of 5ms between tenant and provider is applied. Tests are run sequentially and PDP evaluation is done single-threaded. Each test starts with five warm-up requests and is repeated until the confidence interval lies within 1% of the sampled mean for a confidence level of 95%.

Results. The whole set of results is publicly available[2]. Fig. 6 shows the decision time in terms of the number of single-valued attributes. If the policy only requires tenant attributes (Fig. 6a), provider-side authorization performs significantly worse since each attribute is fetched separately the moment the PDP requires it

[1] `http://sourceforge.net/projects/sunxacml/`, `http://www.sqlite.org/`, `http://cxf.apache.org/`, `http://tomcat.apache.org/`, `https://wiki.shibboleth.net/confluence/display/OpenSAML/`

[2] `http://people.cs.kuleuven.be/~maarten.decat/doa-trusted-cloud-2013/`

and each query takes about 20ms as a result of the network latency and XML operations. If the policy only requires provider attributes (Fig. 6b), federated authorization performs worse; full provider-side authorization and provider-side authorization give the same results since all attributes are hosted by the provider.

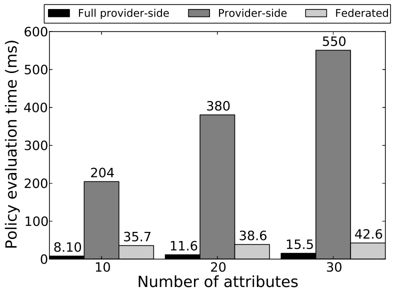
Fig. 7 combines many tenant and provider attributes and shows the decision time in terms of the percentage of tenant attributes in a total of 30 attributes. The figure shows that full provider-side authorization performs best in all cases since no remote queries are required and that federated authorization outperforms provider-side authorization when a little more than half of the attributes are tenant attributes. The asymmetry is a consequence of the extra request needed with federated authorization with respect to provider-side authorization.

Fig. 8 shows the decision time in terms of attribute size, i.e., the number of values in a single attribute. In this case, we employ policies which require a single attribute with 10, 20 or 30 values. If the policy only requires a tenant attribute (Fig. 8a), full provider-side authorization again performs best and both other cases perform similarly since one query between provider and tenant is required. The difference between provider-side and federated authorization is due to the complexity of the tenant-side operation: attribute fetch vs policy evaluation. If the policy only requires a provider attribute (Fig. 8b), federated authorization performs significantly worse than the others, since only this case requires remote queries. In all cases, the decision time remains constant with the size of the required attribute. The absolute difference between Fig. 8a and Fig. 8b is due to two requests instead of one. The absolute difference between many and large attributes in the full provider-side case is due to the fact that the number of required XML translations grows linearly with the number of attributes. Since these numbers are constant with respect to the attribute size, we did not combine them as in Fig. 7.

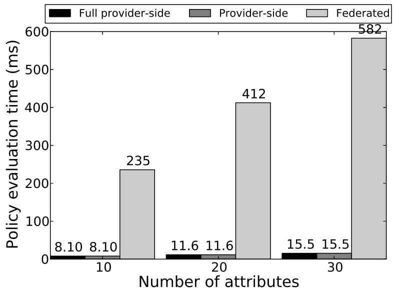
From these results, we can conclude that federated authorization comes with a performance penalty compared to full provider-side authorization. However, depending on the relative amount of tenant attributes in the tenant policies, federated authorization can achieve better performance than provider-side authorization with federated authentication. To illustrate a realistic case, the example policy rules from the e-health case study presented in Section 3.1 require significantly more tenant attributes than provider attributes: the tenant hosts the subject roles, treating relationships, patient consent and patient diseases while the provider hosts ownership relations and the application data itself.

6 Discussion

In this paper, we described federated authorization. The two main goals of federated authorization were (i) to enable scalable access control management for SaaS applications by integrating it with the on-premise tenant infrastructures and (ii) to lower the required trust in the provider by removing the need to share sensitive access control data. Federated authorization effectively achieves both by allowing the tenant to evaluate its policies locally, but does come with a performance penalty, as shown in Section 5. Full provider-side authorization



(a) Many tenant attributes



(b) Many provider attributes

Fig. 6. Decision time in terms of the number of tenant and provider attributes

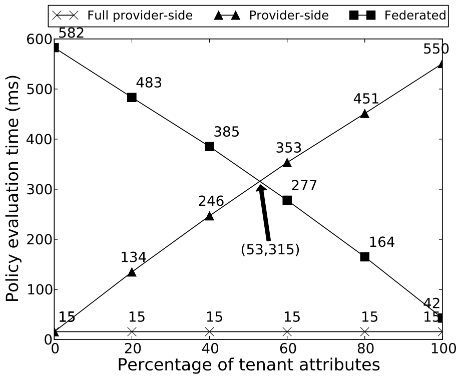
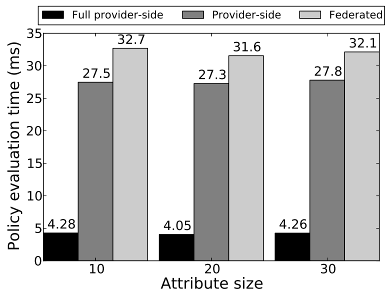
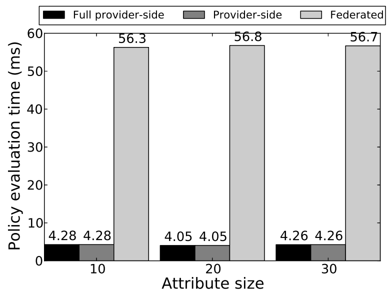


Fig. 7. Policy evaluation time in terms of the percentage of tenant attributes in a total of 30 attributes. The policies from the case study require significantly more tenant attributes than provider attributes, resulting into better performance for federated authorization than provider-side authorization.



(a) Large tenant attribute



(b) Large provider attribute

Fig. 8. Decision time in terms of the size of provider and tenant attribute

achieves better performance, but does force the tenant to disclose sensitive access control data to the provider. Therefore, we can conclude that federated authorization poses a clear trade-off between security and performance.

While federated authorization removes the need to share sensitive access control data, a potential threat lies in the fact that the provider could still infer information about this data using the collection of access control requests and responses from the tenant collected over time. We argue (i) that the provider has no business incentive for this, since very little can be gained w.r.t. the contract with the tenant and (ii) that the possibly inferred knowledge is limited since both the tenant policies and the access control data used for evaluating these are kept unknown. However, future work is required to answer this question more quantitatively, for example using techniques such as logical abduction. Also notice that externalizing policy evaluation to the tenant allows it to lie or provide incorrect results. However, the tenant has no incentive to do this, since the policies control access to its own data.

Also, while lowering the required trust in the provider, federated authorization does introduce new security threats. From the point of view of a network attacker, the main difference between provider-side and federated authorization is the externalization of the policy evaluation process. This results in a new communication channel between tenant and provider, which introduces a denial of service threat: since a tenant decision is required for every request to the SaaS application, blocking the channel or the services is a way to deny the use of the application for a specific tenant. This threat cannot be easily mitigated. The channel should also be secured against the threats of information disclosure, tampering and spoofing and against attacks such as replay. For SOAP web services, WS-Security [18] provides the necessary security primitives.

Towards the future, the performance of federated authorization can be improved. First of all, the performance evaluation shows that the overhead of federated authorization is highly dependent on the number of requests between provider and tenant. Therefore, a good strategy for fetching attributes is essential to improve performance. For example, combining multiple attributes in a single request is able to lower the number of required requests. In the extreme case, this approaches the tests with a single multi-valued attribute. Secondly, employing caching also has the ability to lower the required provider-tenant communication. However, caching also has an impact on security, as discussed by Gheorghe et al. [15]. Next to attributes, access control decisions can also be cached or even inferred, as shown by Wei et al. [25]. We plan to investigate both in future work. Finally, a more fine-grained and dynamic policy deployment strategy also has the ability to improve performance while maintaining the trust advantages. This paper explored two extremes: the tenant policies are either completely evaluated by the tenant or by the provider. Based on the location of the required data and its sensitivity, the tenant policies could be split and distributed for optimal performance while maintaining the confidentiality of the tenant data. We recently explored this strategy [13] and plan to refine this work in the near future.

7 Related Work

Related to this work, xDAuth, the work of Lischka et al. and the work of Ardagna et al. take on similar requirements. xDAuth [7] provides confidential access control on remote applications by introducing an authorization broker. However, this approach effectively shifts the required trust to a central third party. Lischka et al. [19] introduce the concept of *deductive policies*, which resemble our policy references. While their work focuses on the policy language, this work focuses on supporting middleware and the trade-off between performance and security. Finally, Ardagna et al. [8] build on XACML and SAML to achieve privacy-preserving access control based on anonymous credentials, which can be used for claiming to adhere to a certain policy without disclosing the user's attributes. With respect to their work, this paper investigates this aspect more thoroughly in the context of SaaS.

Apart from federated authorization, a large body of work also exists about federated access control in general, for example in grid computing. Grids comprise federations of resource providers and consumers joined together in *virtual organizations* and access control in this domain also focuses on scalable management. The most relevant for this work is CAS [23]. Similar to federated authorization, CAS allows resource owners to grant access to a community as a whole and lets the community itself manage fine-grained access control, thereby separating both roles in access control management. However, CAS does not allow community policies to reason about the objects located at the provider side, thereby not achieving full federated authorization.

Some alternative approaches to federated authorization exist as well. Firstly, automated policy deployment is an alternative for achieving administrative scalability. For example, Stihler et al. [24] argue that policy writing should be split between provider and consumer and propose a system in which the consumer automatically deploys its policies to the provider. Efforts such as SPML [14] try to standardize an access control configuration interface for SaaS applications, focusing on user management. However, while these techniques provide administrative scalability, all policies are still evaluated by the provider and they do not address the necessary disclosure of sensitive tenant access control data.

Another alternative for achieving confidentiality is encrypting the access control policies and data when sharing them with the provider. This has been explored by Asghar et al. [9] and could theoretically allow the provider to evaluate encrypted policies based on locally available and asynchronously deployed encrypted tenant attributes. However, a large performance overhead is introduced for policy evaluation and the expressiveness of the supported policies is still limited, making federated authorization a more generally applicable technique.

Finally, XML gateways such as IBM Tivoli Access Manager [17] are a third alternative for federated access control on remote services. An XML gateway is placed at the perimeter of the organization and reviews every request to a remote service, similar to a firewall. This approach is also able to enforce tenant access control policies based on local access control data. However, the policies are limited to reasoning about service messages and are mainly used for transparently adding credentials or for encryption. Moreover, SaaS applications are a good match with mobile users, while XML gateways require every request to originate from the physical network of the tenant.

8 Conclusions

In this paper, we have described the concept of federated authorization, supported by a generic middleware architecture and policy language extensions. Based on a prototype of the middleware, we have shown that the notion of federated authorization is a feasible and promising approach that provides a clear trade-off between security and performance.

This work was motivated in the context of SaaS. While traditional access control focuses on the provider controlling access to its own data, access control for SaaS should also involve the tenant. SaaS requires federated authorization because of the inherently limited trust between tenant and provider: the tenant may trust the provider with the data in the application, but not necessarily with sensitive data required for authorization. Moreover, because an organization can be tenant for a large number of SaaS applications, centralization of tenant access control management is essential to the adoption of SaaS. However, federated authorization is applicable in many distributed applications that involve multiple organizations. In today's growing ecosystem of service-oriented business coalitions, the need for federated access control and for federated authorization in particular will only grow.

Acknowledgements. We would like to thank Wouter De Borger for his in-depth discussions and feedback. This research is partially funded by the Research Fund KU Leuven, by the EU FP7 project NESSoS and by the Agency for Innovation by Science and Technology in Flanders (IWT). With the financial support from the Prevention of and Fight against Crime Programme of the European Union (B-CCENTRE).

References

1. Security Assertion Markup Language (SAML) v2.0 (March 2005), http://www.oasis-open.org/standards#samlv2.0
2. OpenID Authentication 2.0 - Final (December 2007), http://openid.net/specs/openid-authentication-2_0.html
3. 3-D Secure - Wikipedia, the free encyclopedia (July 2013), http://en.wikipedia.org/wiki/3-D_Secure
4. E-Health Information Platforms (E-HIP) (July 2013), http://distrinet.cs.kuleuven.be/research/projects/showProject.do?projectID=E-HIP
5. Healthcare professional's collaboration Space (Share4Health) (July 2013), http://distrinet.cs.kuleuven.be/research/projects/showProject.do?projectID=Share4Health
6. Permission, User Management and Availability for multi-tenant SaaS applications (PUMA) (July 2013), http://distrinet.cs.kuleuven.be/research/projects/showProject.do?projectID=PUMA

7. Alam, M., Zhang, X., Khan, K., Ali, G.: xDAuth: a scalable and lightweight framework for cross domain access control and delegation. In: ACM SACMAT, pp. 31–40 (2011)
8. Ardagna, C.A., De Capitani di Vimercati, S., Neven, G., Paraboschi, S., Preiss, F.-S., Samarati, P., Verdicchio, M.: Enabling privacy-preserving credential-based access control with xacml and saml. In: IEEE CIT, pp. 1090–1095 (2010)
9. Asghar, M.R., Ion, M., Russello, G., Crispo, B.: ESPOON: Enforcing encrypted security policies in outsourced environments. In: 2011 Sixth International Conference on Availability, Reliability and Security, ARES, pp. 99–108. IEEE (2011)
10. Ashley, P., Hada, S., Karjoth, G., Powers, C., Schunter, M.: Enterprise privacy authorization language (EPAL). Technical report, IBM (2003)
11. European Commision. Directive 95/46/EC. Directive of the European Parliament and of the Council of 24 October 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data (1995)
12. Damianou, N., Dulay, N., Lupu, E., Sloman, M.: The Ponder policy specification language. In: Sloman, M., Lobo, J., Lupu, E.C. (eds.) POLICY 2001. LNCS, vol. 1995, pp. 18–38. Springer, Heidelberg (2001)
13. Decat, M., Lagaisse, B., Joosen, W.: Toward efficient and confidentiality-aware federation of access control policies. In: Proceedings of the 7th Workshop on Middleware for Next Generation Internet Computing. ACM (2012)
14. Cole, G., et al.: Service Provisioning Markup Language (SPML) Version 2.0. OASIS Committee Specification (2006)
15. Gheorghe, G., Crispo, B., Carbone, R., Desmet, L., Joosen, W.: Deploy, adjust and readjust: Supporting dynamic reconfiguration of policy enforcement. In: Kon, F., Kermarrec, A.-M. (eds.) Middleware 2011. LNCS, vol. 7049, pp. 350–369. Springer, Heidelberg (2011)
16. Jin, X., Krishnan, R., Sandhu, R.: A Unified Attribute-Based Access Control Model Covering DAC, MAC and RBAC. In: Cuppens-Boulahia, N., Cuppens, F., Garcia-Alfaro, J. (eds.) DBSec 2012. LNCS, vol. 7371, pp. 41–55. Springer, Heidelberg (2012)
17. Karjoth, G.: Access control with IBM Tivoli access manager. ACM TISSEC 6(2), 232–257 (2003)
18. Lawrence, K., Kaler, C., Nadalin, A., Monzillo, R., Hallam-Baker, P.: Web Services Security: SOAP Message Security 1.1 (WS-Security) (2006)
19. Lischka, M., Endo, Y., Sánchez Cuenca, M.: Deductive policies with XACML. In: Proceedings of the 2009 ACM Workshop on Secure Web Services, SWS 2009, pp. 37–44. ACM, New York (2009)
20. Lockhart, H., Parducci, B., Rissanen, E.: SAML 2.0 Profile of XACML, Version 2.0
21. Mell, P., Grance, T.: The NIST definition of cloud computing. National Institute of Standards and Technology 53(6), 50 (2009)
22. Moses, T., et al.: eXtensible Access Control Markup Language (XACML) Version 2.0. OASIS Standard, 200502 (2005)
23. Pearlman, L., Welch, V., Foster, I., Kesselman, C., Tuecke, S.: A community authorization service for group collaboration. In: Proceedings of the Third International Symposium on Policies for Distributed Systems and Networks, pp. 50–59. IEEE (2002)
24. Stihler, M., Santin, A.O., Calsavara, A., Marcon, A.L.: Distributed usage control architecture for business coalitions. In: IEEE International Conference on Communications, ICC 2009, pp. 1–6. IEEE (2009)
25. Wei, Q.: Towards improving the availability and performance of enterprise authorization systems. PhD thesis, University of British Columbia (2009)

Policy4TOSCA: A Policy-Aware Cloud Service Provisioning Approach to Enable Secure Cloud Computing

Tim Waizenegger[1], Matthias Wieland[1], Tobias Binz[2], Uwe Breitenbücher[2], Florian Haupt[2], Oliver Kopp[1], Frank Leymann[2], Bernhard Mitschang[1], Alexander Nowak[2], and Sebastian Wagner[2]

[1] Institute of Parallel and Distributed Systems
[2] Institute of Architecture of Application Systems
University of Stuttgart
Universitätsstr. 38
70569 Stuttgart, Germany
`firstname.lastname@informatik.uni-stuttgart.de`

Abstract. With the growing adoption of Cloud Computing, automated deployment and provisioning systems for Cloud applications are becoming more prevalent. They help to reduce the onboarding costs for new customers as well as the financial impact of managing Cloud Services by automating these previously manual tasks. With the widespread use of such systems, the adoption of a common standard for describing Cloud applications will provide a crucial advantage by enabling reusable and portable applications. TOSCA, a newly published standard by OASIS with broad industry participation provides this opportunity. Besides the technical requirements of running and managing applications in the cloud, non-functional requirements, like cost, security, and environmental issues, are of special importance when moving towards the automated provisioning and management of Cloud applications. In this paper we demonstrate how non-functional requirements are defined in TOSCA using policies. We propose a mechanism for automatic processing of these formal policy definitions in a TOSCA runtime environment that we have developed based on the proposed architecture of the TOSCA primer. In order to evaluate our approach, we present prototypical implementations of security policies for encrypting databases and for limiting the geographical location of the Cloud servers. We demonstrate how our runtime environment is ensuring these policies and show how they affect the deployment of the application.

Keywords: Cloud Computing, TOSCA, Cloud Service, Cloud Management, Policy-Framework, Security, Green-IT, Sustainable Cloud Service.

1 Introduction

In recent years, the steadily increasing use of IT in enterprises has lead to higher management efforts concerning the whole application lifecycle. This becomes a challenge for enterprises as the degree of complexity increases with each new application and technology [5], whilst manual operator errors account for the largest fraction of failures [17].

Cloud Computing tackles these issues by enabling enterprises to automate and outsource their IT, as well as its management [7]. Therefore, automated deployment and provisioning systems for Cloud-based applications become more and more important and prevalent. They help to reduce the onboarding costs for new customers as well as the financial impact of managing Cloud Services by automating these previously manual tasks. With the widespread use of such systems, the adoption of a common standard for describing C applications that prevent vendor lock-in. Therefore, the Topology and Orchestration Specification for Cloud Applications (TOSCA [14]) was published as a new Cloud standard by OASIS in 2013 with broad industry participation.

TOSCA enables application developers to model and package their Cloud applications including all management aspects in a portable, interoperable, and standardized fashion. This allows automating the whole application lifecycle management from provisioning, over maintenance, to termination. In addition, applications can be deployed on various kinds of infrastructures, integrated and combined with any XaaS offerings, and even migrated between different Cloud providers. As non-functional requirements, like cost, security, and environmental issues, are very important when providing, using and managing applications, TOSCA allows specifying policies that express such non-functional requirements. However, TOSCA lacks a detailed description of how to apply, design, and implement policies.

In this paper, we tackle this issue and demonstrate how non-functional requirements can be defined in TOSCA as policies. We exemplarily use policies emerging from the security domain. We further demonstrate how a prototypical TOSCA runtime environment processes those policies. We propose two mechanisms for implementing policy-specific logic: (i) Policy-aware Management Plans and (ii) Policy-aware Management Operations. The presented approach enables Cloud providers as well as application developers to specify, design, and implement various kinds of policies.

The remainder of this paper is structured as follows: Section 2 describes related work in the field of Cloud service deployment. Section 3 provides an introduction into the TOSCA standard and briefly introduces our prototypical TOSCA runtime implementation. Section 4 explains the formal definition of policies in the TOSCA standard and introduces our example applications and policies. Section 5 presents the automated processing of policies in TOSCA following two approaches including a discussion. Finally, Section 6 concludes the paper.

2 Related Work

NIST [10] defines Cloud Computing as a model for enabling ubiquitous, convenient, and on-demand network access to a shared pool of configurable computing resources. An important aspect of Cloud Computing is the fast deployment of the resources with minimal management effort. To achieve this goal, the resources have to be formally described in a so-called *Cloud service*. A Cloud Service is a *composite application* that consists of different components. The setup of this Cloud Service is defined in an *application model* which defines the *topology* of the service.

There are different approaches available that enable the description of composite applications. Unger et al. [21] present an application model, which allows describing dependency and deployment relations between components. Cafe [11] is a system that supports the modeling and deployment of composite applications. It is based on a formal definition of an application model. Leymann et al. [8] broadens this model and adds labels to optimize the distribution of applications between different Clouds.

Furthermore, there are systems available in the industry to describe application models, like the Service Component Architecture (SCA) [1]. SCA allows composing applications out of services by defining functional relations. Hence, other relations, e.g., where a service is deployed, are not captured. In software architecture, design, and development languages are used to describe the structure of applications. There are different languages available, e.g., Acme architectural description language [6] or the well-known Unified Modeling Language (UML) [16]. For instance, Machiraju [9] uses UML to model application structures in the scope of topology discovery. However, they target mostly application architectures and do not allow to model formal policy-aware service topologies, as it is needed in this paper.

The Topology and Orchestration Specification for Cloud Applications (TOSCA) is a standardization initiative by OASIS to define the topology and management aspects of Cloud applications. This paper is using the concepts of TOSCA, because TOSCA allows describing the Cloud Service topologies, the deployment process of the services and the policies that have to be followed. Thus, all aspects we need for Cloud Service provisioning can be specified in TOSCA.

The focus of the previously published paper [22] proposing a policy-framework for the deployment and management of Cloud Services is the definition of a taxonomy for policies and their aspects. Based on that a signature of a policy can be defined. Furthermore, a high-level architecture of a policy-framework is presented. The current paper describes a technical solution for the deployment stage in the lifecycle aspect of a policy defined by Waizenegger et al. [22]. Furthermore, the taxonomy defined by Waizenegger et al. is used for the formal definition of policies in this paper.

The goal of this paper is to introduce different aspects of policies and to evaluate how to use policies for Cloud Service deployment and management with TOSCA. One use case for policy-aware Cloud Service provisioning is security. A good way to achieve security in Cloud Computing is certification [19]. Certification is only possible if the Cloud Service description is not changed after it was certified, so different offerings for one Cloud Service description may be available. An offering defines which policies of the set of possible policies defined in the Cloud Service description should be enforced when the Cloud Service is being deployed. A Cloud Service description can be certified, but customers may still select the needed configuration based on choosing a proper offering. There are publications that focus on specifying frameworks for implementing different methods to provide security features, e.g., authentication across different providers or trust management [20].

In an earlier publication [4], we describe a framework that enables the fully automated provisioning of applications under compliance with policies that are used to configure, guide, and extend the provisioning. However, this approach is limited as polices can be attached only to nodes and relations, not to the overall Topology Template. This restricts application developers from defining high-level requirements such as all-encompassing data-location policies that apply to the whole application. This limitation forces them to use fine-grained policies on individual elements, which is not sufficient in many cases as discussed in Sect. 5.

The approach described by Nowak et al. [13] allows to build TOSCA service descriptions based on a set of different service description patterns. However, these service descriptions patterns are defined in a generic manner. Thus, using policies as described in this work would enable the configuration of those service descriptions regarding concrete use cases.

3 Introduction to TOSCA

A Cloud Service in TOSCA is defined by two parts, first, the topology defining the structure of the application and, second, the orchestration part which defines

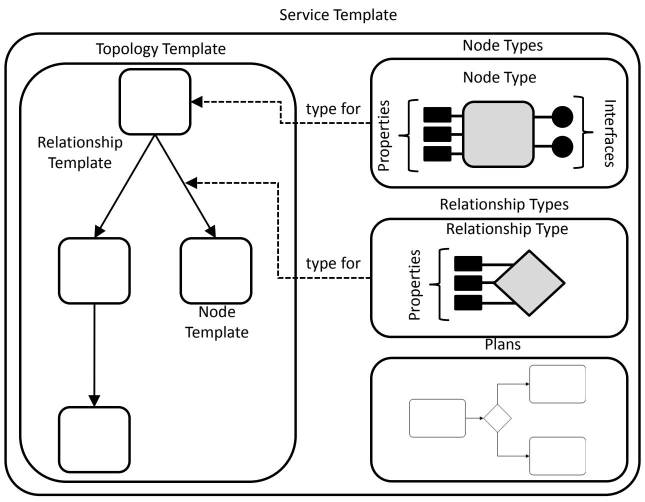


Fig. 1. TOSCA Service Template

the deployment and management of the applications components. In the TOSCA jargon, an application model comprised of the above mentioned elements is called a *Service Template* (see Fig. 1).

The topology is a graph of typed nodes and directed, typed edges called a *Topology Template*. The types, which can be extended and derived, define, for example, the properties and management operations of the respective nodes and edges. The node type *Apache Webserver*, for example, has properties like "port" or "version" and management operations like "start webserver" or "deploy an application".

Either management operations are shipped as part of the TOSCA application, called *Implementation Artifacts*, or they are realized by an external service like the *Amazon EC2 Management Web Service*, or a combination of both. Types are instantiated in the topology by so-called *Node Templates* or *Relationship Templates*, respectively. Templates define the actual properties and how the components of the application are connected. The actual implementation of the nodes is provided as a *Deployment Artifact* in the templates, for example, a ZIP file containing the Apache Webserver or a WAR file embodying the Web service node.

Figure 2 shows an example TOSCA application topology. The topology shows a LAMP-based application (Linux, Apache, MySQL, and PHP) which runs on two virtual machines hosted on Amazon EC2. The rounded rectangles depict node templates, the vertical arrows between the node templates depict relationship templates of type "hosted-on". As we use Vino4TOSCA [3] to visualize application topologies, all types are depicted as text enclosed by parentheses.

TOSCA addresses the automation and portability of the application's management aspects through Management Plans. Management Plans are based on existing workflow technology and orchestrate the type-specific management operations of the nodes and edges into higher-level application-specific management operations [2]. A special *Build Plan* defines how to instantiate the service. Refering to our example topology in Fig. 2 the Build Plan first starts the two Ubuntu virtual machine instances on Amazon EC2, then installs the Apache Webserver and MySQL RDBMS, creates a new database, and in the last step imports the database schema, installs the Java application, and configures the database connectivity in the Java application. Another Management Plan for the backup of the example application may create a dump of the database and archive this file.

All the artifacts required for the application are packaged into a standardized *Cloud Service ARchive* (CSAR) together with the actual TOSCA files, serialized in XML. Following the TOSCA principles, the CSAR file is portable between different TOSCA runtimes. The TOSCA runtime is responsible for running the Implementation Artifacts and Management Plans, provide the artifacts contained in the CSAR file, and hold the state of the application. A generic architecture of an imperative TOSCA runtime is described by OASIS [15]. Here, *imperative* means that plans provide the deployment and management logic, and not the runtime itself.

On a high-level, the CSAR file is processed as follows: (1) The TOSCA definitions are validated and analyzed. (2) Then the Implementation Artifacts are deployed by the TOSCA runtime into a suitable container for the respective Implementation Artifacts. Our TOSCA runtime consits of a container for executing the plans (BPEL engine) as well as another container that hosts the Implemenation Artifacts (Apache Tomcat). (3) Afterwards, the Management Plans, which use the operations provided by the Implementation Artifacts, are bound and deployed to a process engine. (4) Now it is possible to deploy and manage the application provided by the CSAR file.

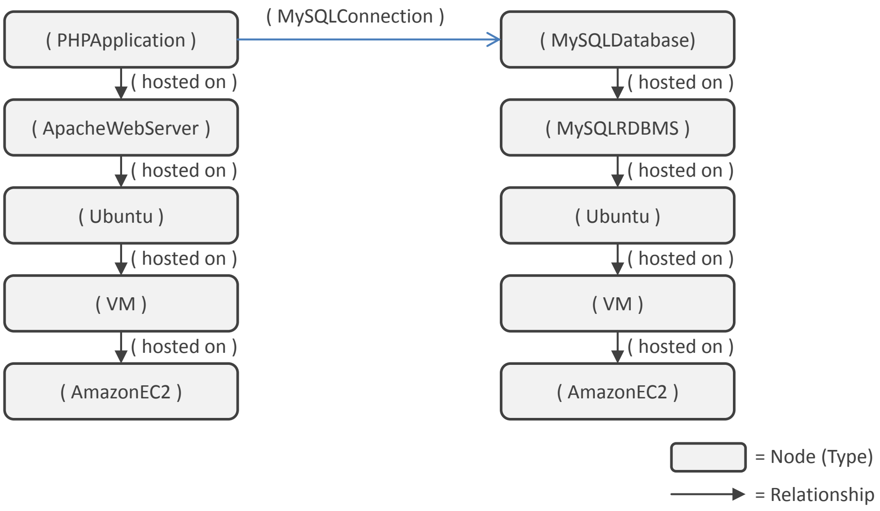


Fig. 2. Topology Template of a LAMP-based TOSCA Application

Similar to the other entities in the TOSCA standard, a policy has an abstract definition; a *Policy Type* (see Listing 1.1). It is instantiated by defining a *Policy Template* (see Listing 1.2). While the Policy Type describes the structure and required parameters of a policy, the Policy Template is used to define a specific policy instance that is annotated to an entity of the topology. A Service Template then contains the topology comprised of Node and Relationship Templates as well the Policy Templates that define which policies are in effect for which entities of the topology. The generic type definitions for relationships, nodes and policies are also contained in the Service Template since they are required as definitions for the specified templates.

4 Policy Fundamentals and Formal Definitions

In order to make automatically deployable Cloud Services secure and still keep the application model flexible, we propose the use of policies that formalize non-functional security requirements. In this paper, we present two mechanisms for

```
<PolicyType name="xs:NCName"
            policyLanguage="xs:anyURI"?
            abstract="yes|no"?
            final="yes|no"?
            targetNamespace="xs:anyURI"?>
  <Tags> ?
    <Tag name="xs:string" value="xs:string"/> +
  </Tags>

  <DerivedFrom typeRef="xs:QName"/> ?
  <PropertiesDefinition element="xs:QName"? type="xs:QName"?/> ?

  <AppliesTo>
    <NodeTypeReference typeRef="xs:QName"/> +
  </AppliesTo> ?

  policy type specific content ?
</PolicyType>
```

Listing 1.1. Policy Type definition according to TOSCA

```
<PolicyTemplate id="xs:ID" name="xs:string"? type="xs:QName">
  <Properties>
    XML fragment
  </Properties> ?

  <PropertyConstraints>
    <PropertyConstraint property="xs:string"
                        constraintType="xs:anyURI"> +
      constraint ?
    </PropertyConstraint>
  </PropertyConstraints> ?

  policy type specific content ?
</PolicyTemplate>
```

Listing 1.2. Policy Template definition according to TOSCA

processing policies during the deployment and management of Cloud Services in TOSCA and demonstrate how they affect these processes using example policies and a Service Template. These example policies and application are introduced at the end of this Section.

In the following, we present a taxonomy for classifying and describing Cloud service policies and show how policies, application models and offerings interact.

Considering the reusability of application models and the involvement of multiple parties like service providers, developers and Cloud providers, the defined policies must follow a common standard that describes what their effect is. This way the suitability of a certain Service Template can be evaluated, and multiple implementations can be compared. Therefore, we propose a taxonomy for the description of policies that was introduced by Waizenegger et al. [22]. In the following, we outline this taxonomy and introduce the syntax and semantics of formal policy definitions in TOSCA.

As an abstract definition, we see a Cloud Service policy as a tuple of elements that describe its behavior and effect. This tuple is given in Definition (1).

$$\text{policy} = \langle \text{stage}, \text{layer}, \text{effect}, \text{property} \rangle \tag{1}$$

We define our taxonomy based on the values of the elements of this tuple. They allow the categorization and comparison of policies in different service templates but are not meant to provide the means for automatically deriving the policy implementation.

The first three elements identify any given policy while the *property* is used to specify the behavior of the policy. Depending on the requirements of the specific policy, this element can be a complex type, an atomic value or it can be absent if the policy is sufficiently described by the other elements. The following will give a brief description of these parameters.

Stage: Identifies the stage in the lifecycle of the Cloud Service at which the policy is being applied. The stages include solution building, instantiation, runtime and termination. It should be noted that the policy might still affect other stages than the one it was applied to.

Layer: The layer defines which part of the topology the policy applies to. It allows for the definition of localized policies that apply to individual nodes and relationships as well as global policies that apply to the entire topology. A discussion of global and localized policies is given in Section 5.

Effect: This element gives a description of what the effect of the policy is i.e. what purpose it has or what aspect of the service it modifies.

In the following, two example policies are given that we use as an application for our taxnomomy and the policy processing methods presented in the remainder of this paper. The first policy, called the *Region Policy*, determines the geographical location of the data center in which the service is deployed. This is necessary for services that are legally obligated to store and process their data in a certain jurisdiction. Since this policy determines an aspect of the underlying infrastructure, it is tightly coupled with the Cloud provider. For any given (virtual) machine, it is a non-trivial problem to determine its physical location with virtual means alone. Therefore, we use an API offered by the Cloud provider to advise them to use a data center in a certain region.

With our second policy (*Database Encryption Policy*), we enable database encryption since this is a frequent customer requirement. This policy does not require interaction with the Cloud provider as it is a purely internal setting of the database component used in the Service Template. It serves as an example to show how policies affect the setup and configuration of service components.

Using the taxonomy introduced above, the values for the Region Policy are as follows. *Stage:* instantiation, *layer:* virtual machines, *effect:* determine the geographical location. The *property* now indicates the specific geographical region chosen for the service, e.g., EU.

The Database Encryption policy is described as follows. *Stage:* instantiation, *layer:* database node, *effect:* enable encypted datastore, *property:* AES256.

In order to apply a single, or multiple policies to a Service Template, an offering as given by Definition (2) is constructed.

$$\text{offering} = \langle \text{service\_template}, \{\text{policies}\} \rangle \tag{2}$$

It is comprised of the Service Template itself as well as a list of policies that govern the lifecycle of the service described by this offering. The process of selecting an offering is given in Fig. 5.

The function *instantiate* represents the service deployment process and is given in Definition (3).

$$\text{instantiate}(\text{offering}, \text{service\_parameters}) \tag{3}$$

It creates an actual instance of the Service according to the given list of policies. This process is implemented by the TOSCA runtime which interpretes the Service Template and applies the policies contained in the offering. Additional *service parameters* are passed to this function to represent the generic, non-policy related information required by the deployment process. A detailed description of this process in given by Section 5 and referring Fig. 6.

In order to provide an application for our policies and to demonstrate their processing during service instatiation, we created a Service Template for the Web-based learning management system Moodle. The topology of our Service Template is given in Fig. 3. Moodle is a prime candidate for our two policies introduced above since it deals with sensitive personal information that is often required to be processed and stored in a certain jurisdiction. Database encryption therefore is also often a requirement in Moodle installations.

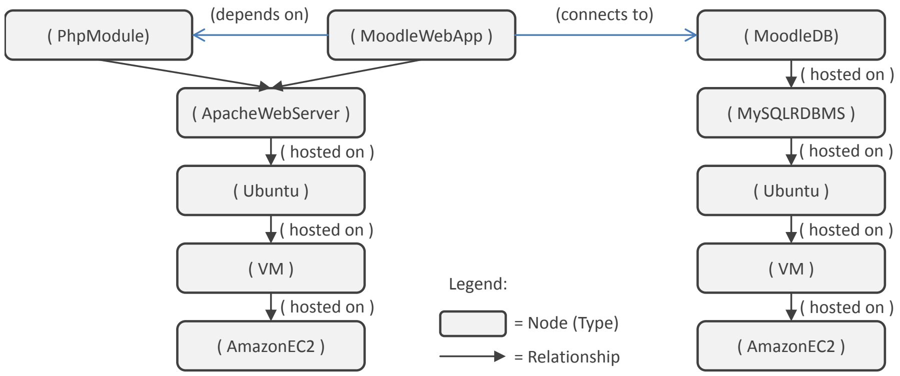


Fig. 3. Moodle Application Topology using Vino4TOSCA [3]

5 Methods for Policy-Aware Cloud Service Provisioning

As far as the TOSCA standard goes, only the annotation of policies in the Service Template is standardized as described above. This leaves various aspects of policy definition and processing up to interpretation and could lead to the emergence of different incompatible implementations that are all standard compliant. We therefore had to make certain assumptions and decided on a concept for processing policies that is in line with the idea of the TOSCA standard.

Enabling and Configuring Policies through Offerings: Once a policy is annotated in the Topology Template, we consider this policy supported by the solution package. There still exists the need to enable or disable certain policies as well as allowing their configuration by providing the property described in Sect. 4. The TOSCA standard does not yet cover any of those requirements, so two approaches to this issue are possible.

First, all policy properties could be defined in the annotation and then read by the plans. This would require modifying the Service Template in order to configure the policies to the customer's specific needs, prohibiting the use of generic application models and making it impossible to validate and digitally sign Service Templates.

Therefore, we chose a second approach in which configuration parameters for the annotated policies are given by an *Offering*. This is implemented by providing different instances (Policy Templates) of a Policy Type that are each part of an offering. This way it is possible to choose a specific set of policies to be active from a single CSAR file without requiring to modify the Service Template. This is especially useful in scenarios with high numbers of customers and policies that require individual configuration.

Interfering Policies: Since it is the purpose of policies to modify the structure and behavior of the service, the possibility exists that certain policies interfere with each other either by affecting the same aspect of the service, or by affecting each other. Resolving such interferences automatically is a non-trivial task, especially since the formal definition of policies in TOSCA does not cover their specific effect. Therefore, we perceive resolving policy interferences a duty of the Cloud security officer who implements the policies in the Service Template. She has to ensure that dependencies and interferences are taken into account by providing policy implementations that are aware of these issues and behave accordingly.

Global and Local Policies: As described in our taxonomy in Sect. 4, a policy can be annotated at different layers in the Service Topology. It follows that a policy can be effective for multiple nodes or relationships at the same time. Therefore, these entities have to be made policy-aware by the Cloud Service security officer even if they are not directly annotated with a policy themselves.

5.1 Cloud Service Lifecycle

In the TOSCA primer [15] and the CloudCycle project [12] the Cloud Service lifecycle is described as shown in Fig. 4. The steps of this lifecycle are the building of a solution package that represents the Cloud Service (step 1). This Solution Package is installed in step 2. Afterwards the service can be instantiated multiple times (step 3) by calling the Build Plan and is managed by management plans

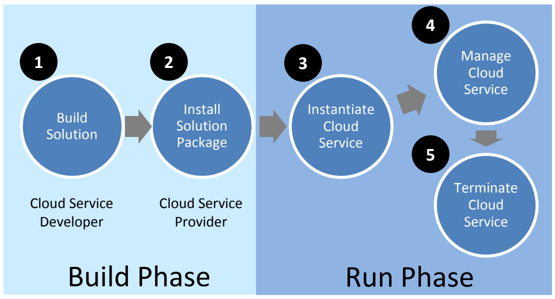


Fig. 4. Lifecycle of a Cloud Service, Based on [12,15]

later on (step 4). When the Cloud Service is not needed anymore it is terminated by a Termination Plan (step 5).

We enhanced the Cloud Service lifecycle with new steps for enabling secure Cloud Computing based on policies and offerings (see Fig. 5).

In the secure lifecycle, a Cloud Service developer bundles a service into a solution package (step 1). As part of the Solution Build Phase, a Security Specialist (Cloud Service Security Officer) secures the application by annotating policies and implementing their functionality through modified Plans or Implementation Artifacts. She combines different selections of policies and their configuration into Offerings and adds them to the policy annotation (step 2). This concludes the design and implementation of the secure Service Template, which is now ready to be provisioned.

Now, a Cloud Service Provider installs this secure solution package in his TOSCA runtime and presents the included offerings to potential customers (step 3). This concludes the Build Phase of the Cloud Service.

In step 4 the Cloud Service Customer selects a solution package and chooses the desired offering (step 5) that determines the policies, which he requires. After this Selection Phase, the Run Phase of the Cloud Service begins. Each time a new instance of the Cloud Service is requested, a Cloud Service Instance

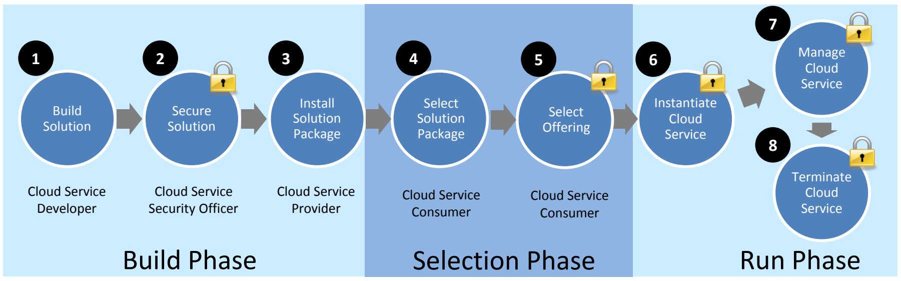


Fig. 5. Security Enhanced Lifecycle of a Cloud Service

is instantiated (step 6) and managed (step 7) by the Cloud Service Provider. When the Cloud Service is no longer needed, the Customer issues its termination (step 8).

5.2 Method for Policy-Aware Cloud Service Provisioning

In this paper, we consider two approaches for implementing policies in an existing Service Template, which can be used in combination or oy themselves. They are performed by the Cloud Service Security Officer.

In the *Plan-based approach (P-Approach)* the Build, Management and Termination Plans are modified to execute additional operations, which implement the security features required by the annotated policies. The *Implementation Artifact based approach (IA-Approach)* on the other hand does not modify the plans, but rather replaces the Implementation Artifacts with security-enabled ones. These new Implementation Artifacts provide the same API as the previous ones, but the operations they provide perform additional steps that implement the required security features. Finally, the service topology is annotated with policies, which can be fulfilled by the replaced Implementation Artifacts and/or Plans. The concrete steps for the IA-Approach are described in Sect. 5.2 and for the P-Approach in Sect. 5.2.

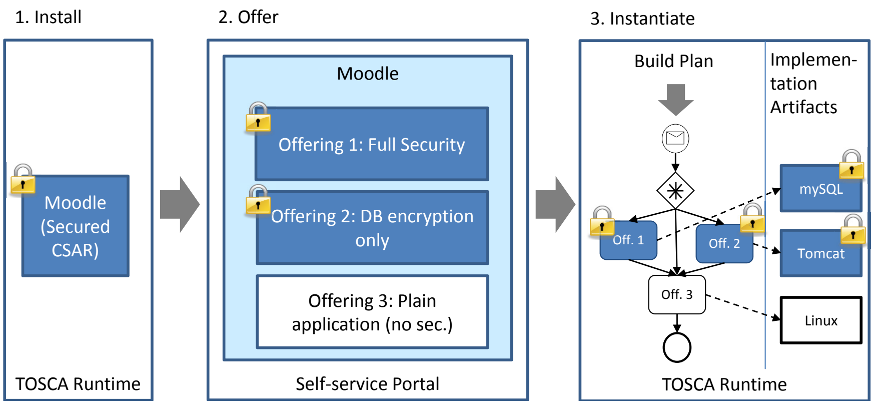


Fig. 6. Automated Policy Processing with Plans and Implementation Artifacts

Figure 6 shows the process of a policy-based Cloud Service instantiation. First, the Cloud Service archive is installed in the TOSCA runtime, in our example, the Moodle application ("1. Install"). This application archive supports different offerings that determine which policies should be active and how they are configured. In our example, these are the Region Policy and the Encrypted Database Policy. The installed archive is then presented in the self-service portal where the users can select their applications. Here different offerings can be selected for the Moodle application ("2. Offer"). Figure 6 shows three example offerings: Offering 1 provides Moodle with all security policies, offering 2 only provides database encryption and offering 3 would be the cheapest, because no security is requested.

Based on the selected offering the Build Plan receives different input parameters for the application instantiation ("3. Instantiate"). Using these parameters the selected offering is passed to the Build Plan as well as additional properties required for service instantiation.

Then the Build Plan is executed and checks with an `if`-statement whether a policy has to be enforced or not and processes the whole topology. Depending on the selected offering, the concrete Implementation Artifacts will be used. In Fig. 6 this is depicted by the arrows connecting the Build Plan activities and the called Implementation Artifacts.

5.2.1 Implementation Artifact Based Policy Enforcement: IA-Approach

In TOSCA, Implementation Artifacts realize service management operations provided by Node Types. When introducing policies, the execution of those management operations will be influenced. For example, depending on whether a database is required to be encrypted or not, its deployment and configuration has to be done differently. Therefore, we propose to support policies in TOSCA through policy-aware Implementation Artifacts.

To realize our approach, existing Implementation Artifacts have to be extended with additional policy enforcing implementations. The existing implementations will remain, as we still need the capability to provide the basic management operations. The extended Implementation Artifacts are thus comprised of two alternative implementations for each service management operation: one that is policy enforcing and one that is not. For each call of a service management operation the Implementation Artifact has to check if any policy has to be enforced. Depending on this check, the Implementation Artifact selects the appropriate implementation to execute the called operation. To realize this concept we propose a two-stage procedure which is given in Fig. 7.

First, a policy support check is performed during the CSAR deployment (steps 1 and 2). For this first stage, we extend the TOSCA processing flow of the container. When parsing a newly deployed TOSCA topology, the container passes information about Node Templates and their associated policies to the corresponding Implementation Artifacts. Each Implementation Artifact then responds if it is capable of enforcing the policies associated with the respective Node Template (step 3).

The second stage is executed by the Implementation Artifacts for each call of the service management operations which is represented by step 4 in Fig. 7. Implementation Artifacts are defined per Node Type; they are not bound to specific Node Templates. Therefore, the Implementation Artifact asks the container for the Node Template the current operation call is associated with, and if there are any policies registered for this Node Template under the current

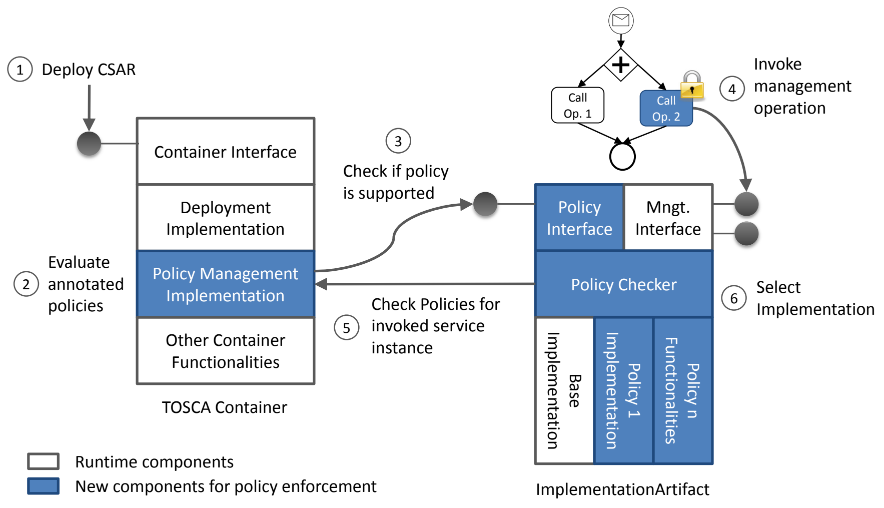

1 Deploy CSAR
Container Interface
Deployment Implementation
Policy Management Implementation
Other Container Functionalities
TOSCA Container
2 Evaluate annotated policies
3 Check if policy is supported
5 Check Policies for invoked service instance
Call Op. 1
Call Op. 2
4 Invoke management operation
Policy Interface
Mngt. Interface
Policy Checker
6 Select Implementation
Base Implementation
Policy 1 Implementation
Policy n Functionalities
ImplementationArtifact
Runtime components
New components for policy enforcement

Fig. 7. Implementation Artifact Based Policy Enforcement

Offering (step 5). Depending on the result, the corresponding implementation for the called service management operation is selected and executed (step 6).

5.2.2 Plan-Based Policy Enforcement: P-Approach

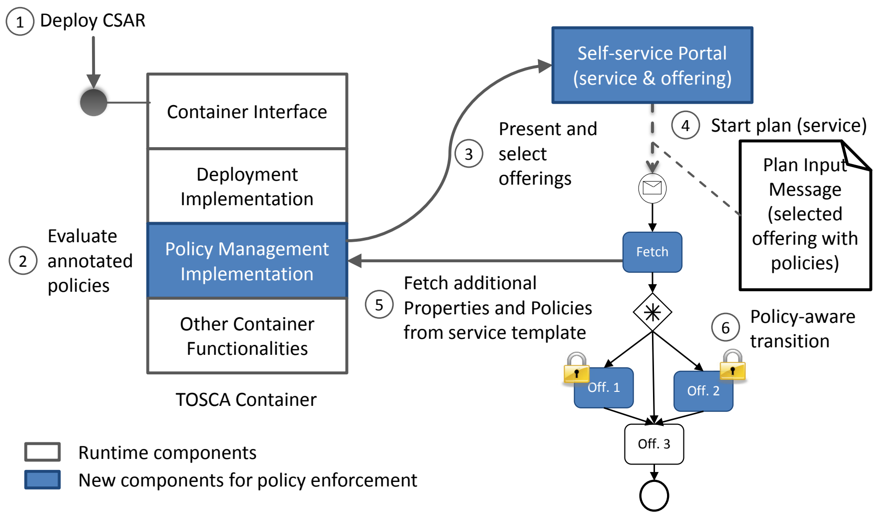
A policy-aware Management Plan is an imperative way to enforce policies. For each policy, the Plan triggers the appropriate steps. These steps are based on the semantics of each policy. Therefore, there is no general rule for writing enforcement instructions. For instance, when enforcing the Region Policy the Plan will insert a request for the specified region into the message it sends to the Cloud Provider. In the case of the Encrypted Database Policy, the Plan will execute additional configuration steps for enabling encryption when setting up the database.

The different steps of P-Approach are shown in Fig. 8. Step 1 is the deployment of a CSAR to the TOSCA Container using the container interface. Then in step 2 the policies annotated in the topology are evaluated and are stored and made available by the Policy Management Component. Step 3 is the presentation and offering of the installed service with all its different offerings in the self-service portal - a web-portal for the customer to select the desired service and offering for the individual security requirements. After the customer selected the service with a concrete offering a message is generated and sent to the Build Plan (input message). This message starts a new workflow instance that processes the build plan (step 4).

The Build Plan then determines which policies should be enforced, it reads the selected Offering from its input message and checks the Policy Management Implementation to determine which policies this Offering includes and what their specific configuration will be (step 5). Then, the Build Plan is executed and based on the activated policies different branches in the Build Plan are executed (step 6).

For the Plan to determine which policies should be enforced, it reads the selected Offering from its input message and checks the Policy Management Implementation to determine which policies this Offering includes and what their specific configuration has to be.

The Cloud security officer has to ensure that the plan conforms to the security annotations given in the topology. One way to check is compliance checking of processes as presented by Schleicher et al. [18].



1 Deploy CSAR
Container Interface
Deployment Implementation
2 Evaluate annotated policies
Policy Management Implementation
Other Container Functionalities
TOSCA Container
Runtime components
New components for policy enforcement
Self-service Portal (service & offering)
3 Present and select offerings
4 Start plan (service)
Plan Input Message (selected offering with policies)
Fetch
5 Fetch additional Properties and Policies from service template
6 Policy-aware transition
Off. 1
Off. 2
Off. 3

Fig. 8. Plan Based Policy Enforcement

5.3 Discussion

In the previous sections, two different approaches were presented to implement policies. These approaches allow the enforcement of policies in a TOSCA environment. However, it should by carefully considered which approach is used for realizing which policy.

Regarding reusability TOSCA allows NodeTemplates or NodeTypes to be reused in different topologies. Similarly, policies implemented with the IA-Approach can be reused very easily as they are implemented locally on the IA level. However, Build Plans are designed globally, i.e. with the whole topology in mind. Hence, the P-Approach is service specific and can not be reused for different Service Topology.

On the other hand, in more complex scenarios, it might be necessary to monitor the whole service on a global level in order to determine if enforcement actions have to be triggered. However, usually IAs are only aware of the local state they are in. This limits their capability to make global decisions concerning policy enforcement that depend on information from other IAs of the topology. This global control can be achieved with the P-Approach. The management plans can be designed to gather the information for policy enforcement from all required components. Based on this information the plans can decide when enforcement actions have to be performed. So the P-Approach is on an other abstraction level and has global knowledge, whereas the IA-Approach works on a detailed level considering propertied of one concrete component.

6 Conclusion

In this paper we first introduced TOSCA a new standard by OASIS for defining Cloud Services. In order to allow the definition of security policies for Cloud Services, we introduce in this paper a formal policy definition based on a taxonomy defining the stage, layer, and effect of policies. Multiple policies are combined into an offering together with a formal TOSCA Cloud Service definition. The customer can now chose such an offering that fits his requirements. We then present a method for policy-aware Cloud Service provisioning consisting of a security-enhanced Cloud Service lifecycle and two approaches (P-Approach and IA-Approach) for processing policies during the service deployment in a TOSCA runtime. We concluded by giving a discussion about the differences between the two approaches.

Acknowledgements. This work was partially funded by the BMWi Trusted Cloud project CloudCycle (01MD11023) and the BMWi IT2Green project Migrate! (01ME11055).

References

1. Beisiegel, M., Booz, D., Colyer, A., Hildebrand, H., Marino, J., Tam, K.: SCA – service component architecture (March 2007)
2. Binz, T., Breiter, G., Leymann, F., Spatzier, T.: Portable Cloud Services Using TOSCA. IEEE Internet Computing 16(03), 80–85 (2012)
3. Breitenbücher, U., Binz, T., Kopp, O., Leymann, F., Schumm, D.: Vino4TOSCA: A visual notation for application topologies based on TOSCA. In: Meersman, R., et al. (eds.) OTM 2012, Part I. LNCS, vol. 7565, pp. 416–424. Springer, Heidelberg (2012)
4. Breitenbücher, U., Binz, T., Kopp, O., Leymann, F., Wieland, M.: Policy-aware provisioning of cloud applications. In: Conference on Emerging Security Information, Systems and Technologies. IARIA (2013)
5. Garbani, J., Mendel, T., Radcliffe, E.: The writing on IT's complexity wall (2010), Forrester Research
6. Garlan, D., Monroe, R., Wile, D.: Acme: an architecture description interchange language. In: Conference of the Centre for Advanced Studies on Collaborative Research. IBM Press (1997)
7. Leymann, F.: Cloud computing. IT – Information Technology 53(4) (2011)
8. Leymann, F., Fehling, C., Mietzner, R., Nowak, A., Dustdar, S.: Moving applications to the cloud: an approach based on application model enrichment. Int. J. Cooperative Inf. Syst. 20(3), 307–356 (2011)

9. Machiraju, V., Dekhil, M., Wurster, K., Garg, P.K., Griss, M.L., Holland, J.: Towards generic application auto-discovery. In: Hong, J.W.K., Weihmayer, R. (eds.) Network Operations and Management Symposium. IEEE (2000)
10. Mell, P., Grance, T.: The NIST definition of cloud computing. Recommendations of the National Institute of Standards and Technology Special Publication 800-145, 7 (2011)
11. Mietzner, R.: A method and implementation to define and provision variable composite applications, and its usage in cloud computing. Ph.D. thesis, Universität Stuttgart (2010)
12. Niehues, P., Kunz, T., Posiadlo, L.: Das CloudCycle-Ökosystem. Tech. rep., CloudCycle (2013)
13. Nowak, A., Binz, T., Fehling, C., Kopp, O., Leymann, F., Wagner, S.: Pattern-driven green adaptation of process-based applications and their runtime infrastructure. Computing, 463–487 (February 2012)
14. OASIS: OASIS Topology and Orchestration Specification for Cloud Applications (TOSCA) Version 1.0 Committee Specification 02 (2013), `http://docs.oasis-open.org/tosca/TOSCA/v1.0/cs02/TOSCA-v1.0-cs02.html`
15. OASIS: Topology and Orchestration Specification for Cloud Applications (TOSCA) Primer Version 1.0 (January 2013), `http://docs.oasis-open.org/tosca/tosca-primer/v1.0/tosca-primer-v1.0.html`
16. Object Management Group: Unified modeling language 2.1.2 super-structure specification. Specification Version 2.1.2, Object Management Group (November 2007)
17. Oppenheimer, D., Ganapathi, A., Patterson, D.A.: Why do internet services fail, and what can be done about it? In: USENIX Symposium on Internet Technologies and Systems (2003)
18. Schleicher, D., Leymann, F., Schneider, P., Schumm, D., Wolf, T.: An Approach to Combine Data-Related and Control-Flow-Related Compliance Rules. In: Conference on Service Oriented Computing & Applications. IEEE (December 2011)
19. Sunyaev, A., Schneider, S.: Cloud services certification. Commun. ACM 56(2), 33–36 (2013)
20. Takabi, H., Joshi, J., Ahn, G.J.: Securecloud: Towards a comprehensive security framework for cloud computing environments. In: Computer Software and Applications Conference Workshops (2010)
21. Unger, T., Mietzner, R., Leymann, F.: Customer-defined service level agreements for composite applications. Enterp. Inf. Syst. 3(3), 369–391 (2009)
22. Waizenegger, T., Wieland, M., Breitenbücher, U.: Towards a policy-framework for provisioning and management of cloud services. In: Conference on Emerging Security Information, Systems and Technologies. IARIA (2013)

All links were last followed on 2013-05-29.

SLA-Based Reputation Life Cycle

Kahina Hamadache

Department of European Projects, Singular Logic S.A. Athens, Greece
khamadache@singularlogic.eu

Abstract. Establishing and managing reputation has become a critical issue for Cloud Computing. In this paper we investigate this issue and propose our conception for the management of Reputation in Cloud Computing. This conception is based on the extensive use of SLA through the whole life cycle of the reputation. With this objective in mind we focus on defining and describing the close relations that exist between the establishment of reputation and the management of SLA. In this perspective we identify key aspects and mechanisms to develop a truly efficient Reputation management based on the use of SLAs.

Keywords: Service Level Agreement, Reputation Management, Service-Oriented and Cloud Computing, Reputation Life Cycle.

1 Introduction

Cloud Computing advocates the outsourcing of more and more complex services, providing a way to virtualize entire ecosystems. Given this trend, companies are facing the choice of migrating their system to the cloud or keeping their infrastructure. But with this new paradigm come new issues to tackle.

Entrusting an unknown or unevaluated entity in providing a reliable, secure, legally sound and efficient service is clearly impossible. Besides, the cloud computing paradigm implies that even if you deal with a service provider, you cannot know or control if the provider will not call services of other third parties. In this perspective, and as intensively suggested by the recent researches, one of the main issue to tackle for the adoption of this paradigm is the management of trust and reputation inside and outside Clouds. Given this inevitable requirement for a better and faster adoption of Cloud Computing, numerous researches have been conducted in this direction. However, there is no consensus on how trust and reputation has to be handled.

Service Level Agreement is a term that refers to the general idea that someone or something that wants to use a service offered by a provider may want, or need, to sign a legal contract with this provider in order to ensure that a certain level of performance (according to some parameters) will be delivered. Thus, Service Level Agreement is really a legal contract binding the consumer and provider to a set of duties, responsibilities and penalties/bonuses in case of under/over-performance. However, and as we will demonstrate in this paper, the conception of SLA for service-oriented computing, goes beyond the simple notion of legal contract, it can be used as a real basis for the collaboration by describing meaningful characteristics and be used to drive and support reputation mechanisms.

This paper is organized as follows: we will first introduce a state of the art on SLA-Based reputation. In the third section we present our SLA-based reputation life cycle. Finally we conclude by discussing our research and by giving some perspectives of our future work.

2 Related Work

This research focuses on the management of Reputation for Cloud Computing by relying on SLA. Hence, the investigation we had to conduct involved looking into the life cycle of Cloud Computing, Management of Reputation and use of SLA.

2.1 Existing Cloud Computing Life Cycles

Due to its complex multi-faceted nature, Cloud Computing has to be described with a more complex life cycle than "usual" service-oriented computing. Conway and Curry [3] proposed a cloud life cycle that can be used for the migration and the ongoing management of public and also for cloud-based services. Their cloud life cycle is divided into four phases that are further divided into nine steps. The four phases and the nine steps are:

- *Architect*: Investigate, Identify, Implementation Strategy and Business design;
- *Engage*: Select, Negotiate;
- *Operate*: Operational Roll-out, Manage the Supply Chain;
- *Refresh*: Review.

This nine-stepped process gives an interesting overview of the cloud life cycle, as it provides an organization and milestones helping the consideration of the cloud.
In [5] the authors described a Service Cloud Life Cycle; it consists of six main stages and 3 looped activities:

- *Deployment*: collection of all information about a service in a service description;
- *User Requirements*: Service Requirements stage;
- *Matchmaking*: Identification of Services fulfilling user requirements;
- *Negotiation*: Service Negotiation;
- *Execution*: Service Consumption
  - *Monitoring, Analyzing, Adjusting*: are looped through during the execution;
- *Ending*: costs for the service execution are billed, and the service is rated by the consumer.

This work is interesting in the perspective that it describes relatively precise life cycle since the deployment stage until the rating task.

Just with these two examples the main aspects of the cloud life appears clearly: a first phase consisting of the service design, deployment, user requirement and service selection; a second phase relying on the negotiation; a third phase consisting of the service execution/operation and a final phase with billing and rating. However, if we consider these works in the perspective of the reputation management, they don't give information about how it could be handled and processed throughout the life cycle.

2.2 Reputation Management

Reputation Management in massively distributed systems has been a particularly hot topic for many years. With the rapid emergence of Peer-to-Peer computing and Service-Oriented Computing, the need for trust and reputation mechanisms has been considered of major importance. Indeed, relying on third parties that you don't always know to support a part of your business (even a small part) is far from being easy. Still, this need was not completely identified by all the players of the market. Within a few years, Cloud Computing has grown tremendously, and almost every large company is considering this technology for a part of its business. But with the possibility to virtualize every component of your system, and then completely virtualize your business, the question isn't any more whether or not you need to have access to Reputation information, the question is how reliable is this information, what has been used to estimate this reputation, how does it live through the Cloud life.

In [9] the authors are presenting an overview of the Reputation and Trust management in service-oriented computing, from the perspective of Service Selection. Among the different remarks they make on the reputation and trust management, they notice that still few work exists on the decentralized management of trust, mainly because of the inherent complexity of such system, and probably because of the need for such system to build a credibility model on top of the reputation model in order to evaluate the credibility of the reputation source, meaning that Reputation information becomes harder to trust due to the potential lack of information on the provenance of feedbacks, leading to a great uncertainty.

In [1] the authors proposed a trust model for Cloud Computing based on the usage of SLA. They describe the requirements and benefits of using SLA for trust modeling in cloud environment, provide a high level architecture capturing major functionalities required, and provide a protocol for the trust model. Their model consists of SLA agents, cloud consumer module and cloud services directory. The SLA agent is the core module of the architecture as it groups the consumers in classes based on their needs, designs SLA metrics, negotiates with cloud providers, selects the providers based on non-functional requirements such as QoS, and monitors the activities for the consumers and the SLA parameters. Cloud consumer module requests the external execution of one or more services. Cloud services directory is the one where the service providers can advertise their services and consumers seek to find the providers who meet their functional requirements such as database providers, hardware providers, application providers etc. The authors have proposed only the model and no implementation or evaluation has been developed or described. Every module will have to be evaluated for its functionality and the effectiveness and finally the overall model will have to be evaluated for its effectiveness.

2.3 Existing SLA Life Cycles and Uses

Service Level Agreement (SLA) is a formal contract negotiated between a service consumer and a service provider for the service. It outlines the relationship between parties to understand each other's needs, preferences and expectations. It should include how to perform future service delivery including QoS levels required, performance-tracking techniques, performance reports, managing problems and conflicts, security and termination. During service delivery, SLA requires real-time verification. If violation occurs, appropriate action (e.g., penalty, conflict resolution) should be taken. Ron et al [2] defined SLA life cycle in three high level phases: the creation phase, operation phase and removal phase. Sun Microsystems Internet Data center Group defined a more detailed SLA life cycle in six steps:

1. *Discover service providers*: where service providers are located according to consumer's requirements.
2. *Define SLA*: includes definition of services, parties, penalty policies and QoS parameters. In this step it is possible for parties to negotiate to reach an agreement.
3. *Establish agreement*: an SLA template is established and filled in by specific agreement, and parties are starting to commit to the agreement.
4. *Monitor SLA violation*: the provider's delivery performance is measured against to the contract.
5. *Terminate SLA*: SLA terminates due to timeout or any party's violation.
6. *Enforce penalties for SLA violation*: if there is any party violating contract terms, the corresponding penalty clauses are invoked and executed.

This life cycle provides some insight on how the SLA will be handled during the use of a service. However, it does not define the entanglements between SLA life and actual service steps.

Indeed, in addition to traditional use of SLA, many research activities are extending the reach of SLA to use them as the base for more and more tasks inside the service and cloud computing paradigms. In [4] an approach to manage Cost and Business Risk Awareness within SLA is proposed. This work is particularly interesting as it demonstrates the potential use of SLA to help monitoring the Business constraints.

SLA appears to be a more and more active topic in recent research on Cloud Computing. This trend is natural. Indeed, Cloud Users are requesting guarantees before diving into the Cloud, and the only way they have to obtain these guarantees is to sign an SLA with the providers. In the following section we will present how our conception of the relations between SLA and Reputation can lead to the use of SLA to drive the creation of accurate Reputation information, simplifying the life of consumers and providers as they can refer to a single document for the management of their relations.

3 Our Approach

As it is our main focus for this article we will make an in-depth investigation of SLA through the whole life cycle of service computing and reputation. Before diving into the life cycle however, we have to more precisely define the concept of SLA that we adopt for our work. Indeed, if the notion of Service Level Agreement can seem self-descriptive and relatively unambiguous, it is in fact necessary to clarify what is an SLA, how it has to be described, what can/could/should be described in SLA.

3.1 Service Level Agreement for Reputation Management

Service Level Agreement is everything except just a simple legal contract. Yes it is a legal contract; in fact, it is the only piece of information indicating what a service has to provide a consumer. Without SLA, service-oriented computing is more or less like a jungle, you can try to eat an unknown fruit, but you cannot have any insight about the consequences. Signing an SLA is a bit like having the list of excipients and consequences on an exotic fruit: you're informed beforehand, but in some cases you don't understand completely the effects. However in the case of SLAs, you can discuss with the different trees you have at your disposal in order to find the fruit that suits you best. In addition, if one of your co-adventurers in the jungle has already tasted such a fruit and has shared his opinion and a "feedback" on the effects of eating one of them, it becomes a bit simpler to choose one fruit or another. An efficient way for those adventurers to evaluate the fruits is simply to comment the effects announced on the fruits according to the expectations and real experience. In the service-oriented computing it corresponds to the evaluation of SLA parameters. And the fact of sharing and dispatching this evaluation become the reputation of the fruits/service.

Exotic fruits have obviously nothing to do with service-oriented computing. However, the analogy made here serves to illustrate the simple fact that SLAs are powerful supports for the reputation. They describe what is expected, while reputation gives information about what was expected and what was finally delivered. Besides, as we will see in the remainder of the section, SLAs are useful throughout the whole life cycle of the service.

Our conception of SLAs relies on some clear assumptions that are required to correctly consider the life cycle itself. One of the basic needs for the construction of SLAs is to ensure that all parties are using the same accurate and understandable terminology. To this extent, we advocate the fact that SLAs have to be represented according to a shared ontology, an ontology of SLAs. In addition, SLAs should be the base of the feedbacks that will be collected and will later become part of the reputation. Thus, Feedbacks should rely on and extend the SLA ontology.

With these precisions given, we can dive into the main part of this section and consider the life cycle of service-oriented computing and identify the key points of SLA and Reputation.

3.2 SLA-Based Reputation Life Cycle Main Phases

The design of an SLA-based Reputation Life Cycle for Cloud Computing requires the exploration of most of the steps of the Cloud from the Reputation perspective, and the consideration of the use of SLA in this perspective. Our consideration of the Cloud Computing Life Cycle is organized in 4 major phases as depicted on Fig. 1.

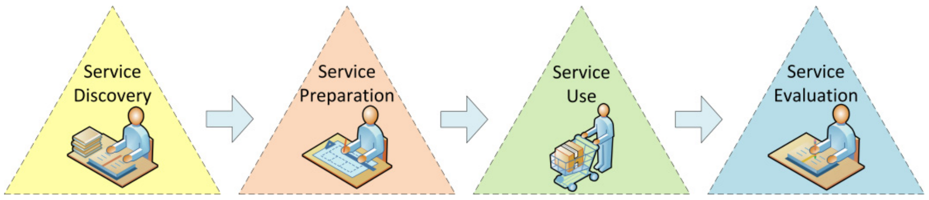


Fig. 1. Cloud Computing Life Cycle Overview

The first phase of this general conception is the *Service Discovery*. This initial phase starts at the very beginning of the Cloud Service life; when consumer and provider aren't known of each other and when the Cloud Service does not exist yet. For these phases, we have to consider that there is no direct relation between the service provider and the service consumer. During this phase, the service provider is building the service and publishing it on a repository. From the consumer perspective, this phase includes several steps, from the service requirement description and SLA drafting, to the service search and selection.

The second main step of the Cloud Service life cycle is the *Service Preparation*. This phase is used by the service consumer and provider to negotiate the terms of the service level agreement. Once these terms have been negotiated and agreed upon, service provider and consumer have to effectively setup their environment for the consumption of the service. On the provider part it may include customization or configuration of the service or/and environment. On the consumer side it requires the integration of the service into the rest of its system. Only when these steps have been completed can the actual use of the service occur. Our first consideration for the overall representation of the Cloud Service Life Cycle was to design a 3-phased cycle based on a simple "*past-present-future*" conception, which in the perspective of Service-Oriented Computing would be mapped as something like "before the use of the service", "during the use of the service" and after the use of the service". However, even if this conception is a valid one, we argue it has to be a bit more detailed to be both relevant and accurate. Indeed, the *service discovery* phase is asynchronously done by provider and consumer; they don't have to know each other or to communicate to operate. It is only at the end of this phase that the communication between them will start as they will undergo the *service preparation* phase.

Once the *service preparation* is completed on each side and the parties are ready, the actual *service use* (or consumption) can take place. During this complex phase, many different elements, entities and action will be involved, spanning from the actual deployment of the service until the monitoring and reporting of SLA violations. More details on this phase will be provided in the reminder of this article.

The fourth phase we have defined is related to the end of use of the service. Once the consumer wants to put an end to its service consumption, or if the ending date of the SLA has been reached. Important actions have to be taken in this regarding the feedback provided by the different entities that where involved in the Service Use.

Now that we have presented the general view of our conception of Service-Oriented Computing we will go into details for each phase and identify the key points for reputation and SLA.

3.3 Service Discovery

Fig. 2 describes the different steps in the first phase of the life cycle.

Service Creation: the very first element in the life of the service is the creation of the service itself, according to a wish of the provider to offer such service to its future clients. This step contains several sub-steps that won't be detailed in this paper as they correspond to service development steps and can vary according to different elements. However, it is important to notice that even at this step we have to consider the reputation and SLAs. We have to consider the reputation at this step, because the development of the service can imply the use of external services and will then induce that a part of the service reputation will be linked to the reputation of these external services. At this stage we have to take SLAs into account for three reasons: first, as we will see in the following phases, we argue that reputation of a service has to be closely related to its respect of SLAs, thus the consideration of external services implies the consideration of the SLAs that have been signed with them in the past; secondly we need to consider the currently signed SLAs with these external service to be able to estimate the boundaries of the envisaged service. Finally we need to consider SLA because when developing the service, depending on its intended use, the service provider will make development choices that will give him limits for the use of its service and then limits in some of the SLA parameters that he will be able to agree on.

Service Publication is the second step, it occurs when the service has been created and is deployed to be used by consumers.

Service Indexing immediately follows the publication, it is done at the initiative of the provider and the service is indexed and referenced in a service registry held by a third party. As it will be evoked later, this kind of registry can be useful not only to simply index the service, but also potentially to aggregate and provide additional data on the service and can then be seen as an advertisement support. At this point in the process, the job of the provider is done and he has to "wait" for consumers to contact him in order to use its service.

From the consumer perspective the steps are completely different. The ***Service Description*** is the first step in the consumer process in using a service. For this step the future consumer will describe the service is looking for in term of functionality. Most probably he will also want to start at least drafting the SLA he's expecting to sign with the service provider he wants to find.

Once the consumer has finished describing the service he's looking for he will effectively make the ***Service Search*** and contact registries to find the adequate service matching his description. In practice this step can be ensured by a Service Broker that will manage the search on behalf of the consumer. In such a case it can be essential for the consumer to draft the SLA as it will simplify the exchanges between the parties and fasten the general process of looking for a service.

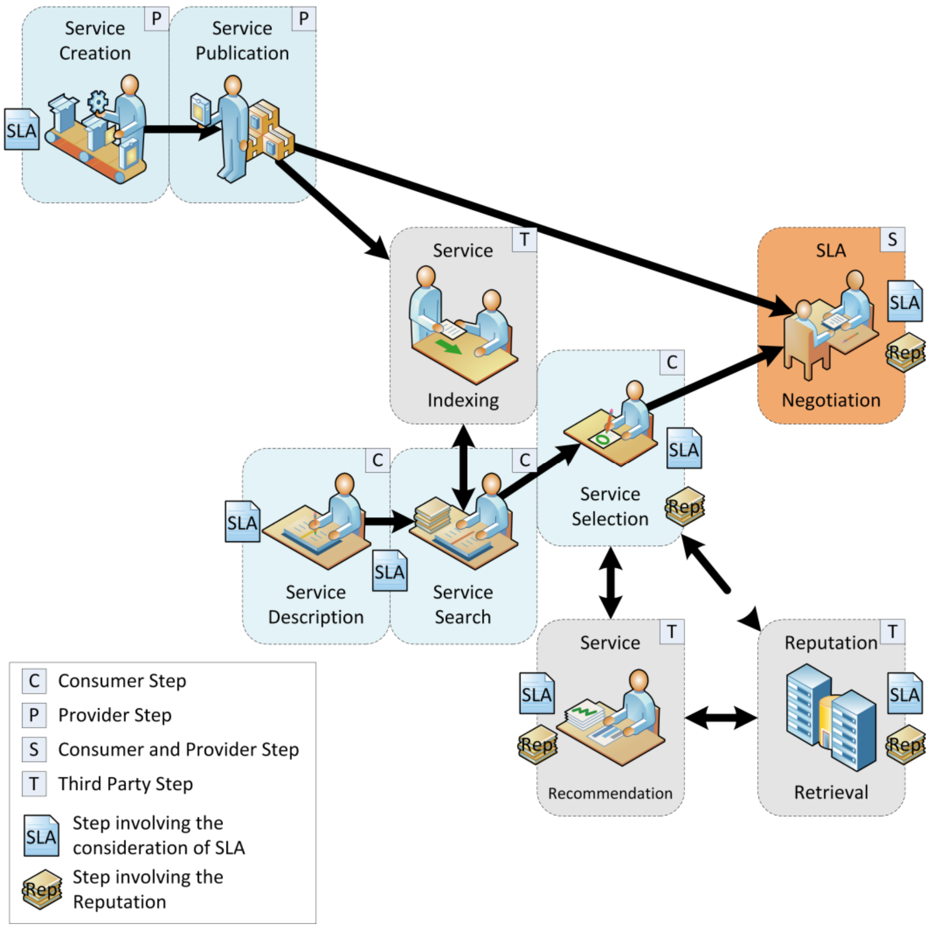


Fig. 2. Cloud Service Discovery Phase

In return registries will provide to the consumer a list of services that match the description he has provided. This will give the consumer the opportunity to choose the service he wants to use, thus he will perform a ***Service Selection***. The selection process has to be done according to the differences between services and the preferences of the consumer. As it is rarely possible to be able to have an opinion or previous experiences of use of all the services proposed and thus make an educated choice, the service selection step is often linked to the ***Service Recommendation***. The recommendation of a service can take several forms and involve different entities of the ecosystem. In our conception we consider that the service recommendation has to reflect past experiences of use of the service. Thus the recommendation has to be influenced by past experience reports, usage feedbacks and be adapted to the final needs of the user. Thus, in our conception, the service recommendation has to be fed by experiences/reputation registries, and provide customized information to the user. Hence, Service Recommendation implies the ***Reputation Retrieval*** from a reputation repository. Such recommendation can be collected directly by the user or can pass through a Service Broker in order to simplify the selection. On the selection and recommendation steps it is important to consider reputation of the service according to past SLAs and the expected SLA for this use, as it will drive a part of the selection process.

3.4 Service Preparation

Once the selection of the service is done and the consumer knows with which provider he wants to deal with, he will contact this provider and initiate the ***SLA Negotiation***. This step links together the first and the second phases of the life cycle. As depicted on Fig. 3.

During the SLA Negotiation, consumer and provider discuss and exchange about the different terms of the SLA on which they want to agree. Given the fact that past experience can provide hints about the expectable SLA, it is of prime importance to accurately consider the reputation of the service (and service provider). This step includes multiples interactions, mainly between the consumer and provider, but also with third parties that will be involved in the service such as service auditors, contractors, service brokers, etc. In the process of the SLA Negotiation between the consumer and the provider, ***Contractor SLA Renegotiation*** can also be involved. For this optional and specific step the service provider may want to reconsider the SLA he has signed with its contractors in order to be able to agree to some of the parameters the consumer is willing to have a specific level of performance. This renegotiation will then potentially imply a customization of the service. After the SLA negotiation between consumer and provider, two outcomes are possible: no agreement on SLA parameters is reached and one of the parties breaks the negotiation. In such a case the consumer will go back the Service Selection step of the Discovery Phase. The second possible outcome is where an agreement is reached by the two parties and an SLA is signed, this corresponds to the ***SLA Signing*** milestone. After reaching an agreement and signing the SLA, the provider may have to undergo an extra two extra steps: customization and redeployment of the service.

The ***Service Customization*** step is done by the service provider, when after signing an SLA with a customer he has to tailor the service that he will deliver in order to match consumer's expectations and ensure the respect of the terms of the SLA. Depending on the level of customization needed, the provider may also go through a ***Service Deployment*** phase. During the Service Customization of the provider, he will also communicate with the consumer to notify him if changes in the use of the service arise. This will drive the consumer for his next step: the ***Service Integration***. Indeed, while issues regarding provider's technical challenges on service development, adaptation and management and consumers difficulties regarding service discovery, selection are often discussed or at least mentioned in researches, we also have to consider the simple fact that the consumer has to integrate its consumption of the service within its own system.

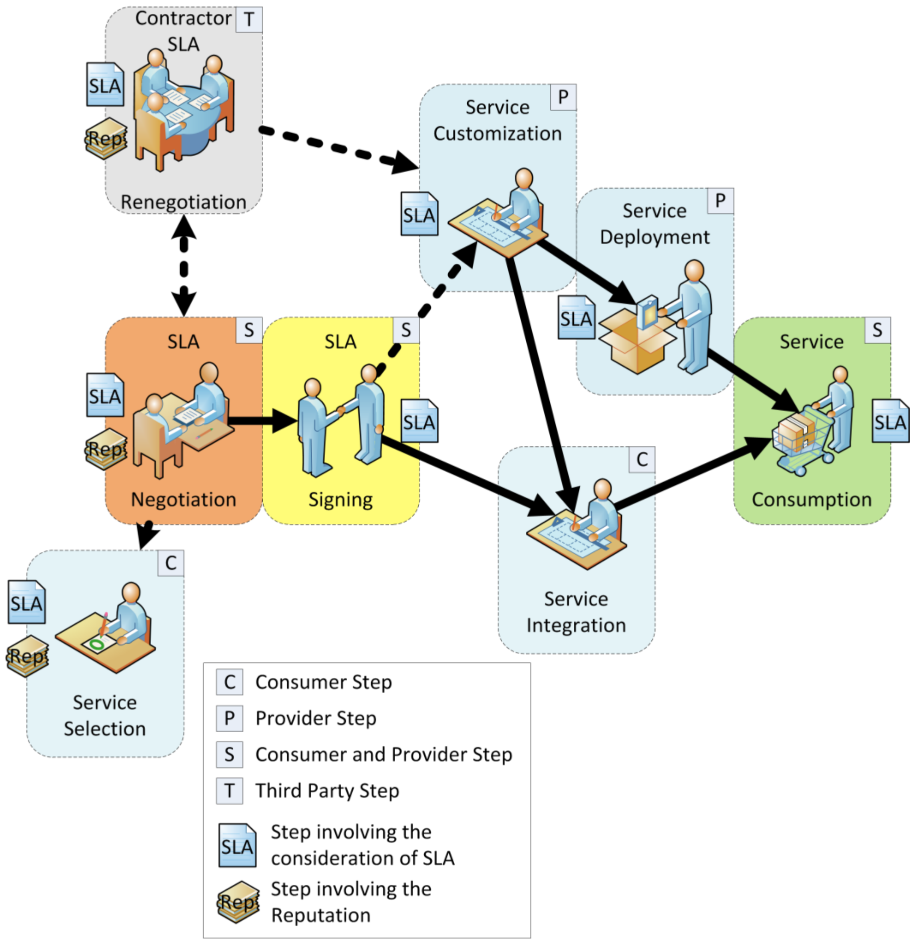


Fig. 3. Cloud Service Preparation Phase

3.5 Service Use

Upon completion of the service integration and service deployment, consumer can finally start the main activity of the life cycle: ***Service Consumption***. The beginning of this activity marks the end of the Service Preparation Phase and the start of the *Service Use Phase* as illustrated on Fig. 4.

Service Consuming itself is the simple fact for the consumer to call the service of the provider. The service use phase includes much more interactions between the parties.

In addition to the simple deployment of a service, the provider has to ensure the efficiency and well-being of the service by supporting the monitoring of his service. This can simply be called ***Service Monitoring*** and has to be considered as a continuous activity during the whole use of the service. This activity allows the service provider to keep its awareness on the state of his service and detect potential issues or needs. The monitoring of the service is closely related to the ***Service Adaptation***. Indeed, while the monitoring task has in charge to collect information regarding the state of the service, the service adaptation allows triggering specific management activities in case a need for change is detected. In such a case, the service adaptation that will occur may take several forms: it can be an adaptation of the execution environment, an adaptation of the contractors (providers) of the provider or an adaptation of the underlying infrastructure used to support the service deployment. In any case, this adaptation has to preserve or improve the respect of the SLA. Once adaptation is done, the provider may have to go through a ***Service Deployment*** step in order to take into account some of the adaptations made. In an ideal world, it would be possible to only have a monitoring of the service from the provider. However, an ideal world takes time to build, and it would be unreasonable to consider that we already live and work in such a place. Thus, in order to be able to somehow trust a provider, we need to ensure that trust by auditing the provided service. This task, the ***Service Auditing*** can be performed either by the service consumer itself, but in order to gather more reliable information and ensure the objectivity of the audit it is preferable to let this activity be performed by a third party entrusted by both consumer and provider. Typically, the identity and the method of auditing can be parts of SLA terms.

Then, the Service Auditing will consist in monitoring and evaluating the respect of performance indicators and other specific parameters of the service. Hence in our conception it would be almost natural to consider that the most important parameters to consider (but not the only ones) for the audit are the one of the agreed SLA. Such a mechanism is naturally suited to handle the task off detecting SLA Violation. Given this possibility, the Service Auditor can, in addition to the ***Service Reporting*** that would generate reports to be used in the last phase of the process, handle the ***SLA Violation Reporting*** activity that will be communicated to the provider (and the consumer) in order to alert him of the need for adaptation of the service. Both Service Auditing (consumer side) and Service Monitoring (provider side) possess common concerns: *Performance Monitoring* to ensure the performance of the service (CPU availability, average response time, network speed, etc.), Security Monitoring (encryption, protocol, access control, etc.), Business Monitoring (when possible and upon applicability of the concept) and Legal Monitoring (for example ensuring that data do not transit in some countries that would not respect the same privacy rights as the one desired by the consumer). Even if there are common items between the two activities, there are also differences. Service Monitoring will also incorporate monitoring and awareness of external resources (external to the service, for example resources dedicated to another service in use) in order to be able to optimize resource usage. SLA Violation and the different monitoring activities have a direct impact on the reputation of the service and its provider, as they will quantitatively and qualitatively reflect the respect of the SLA.

It is also noticeable that using a service can be based on a pay-on-use model, requiring the consumer to have a ***Cost Monitoring*** activity.

Finally, we have to mention the possibility for one party or the other (or both) to ask for an ***SLA Modification***. Indeed, the provider may ask for a modification of SLA parameters because he isn't able to provide the agreed level (with the risk of losing the trust of the consumer) or on the contrary because he wants to improve the SLA. The consumer on his side may want to modify the SLA for different reasons (for instance if one of the parameters that they did not agreed upon is disturbing the consumer's business, or if the requirements of the consumer have evolved).

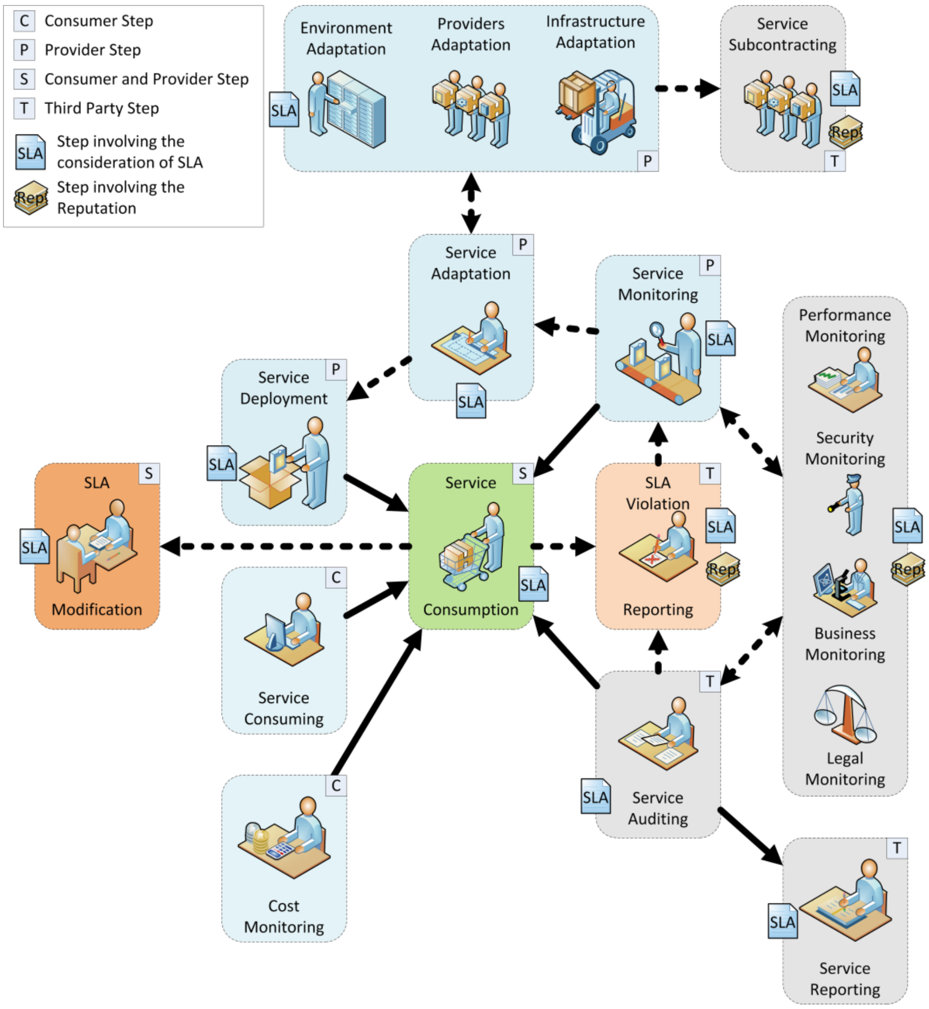


Fig. 4. Cloud Service Use Phase

3.6 Service Evaluation

The Service Use phase is followed by the Service Evaluation Phase (Fig. 5), the last of the process. After the end of the service consumption, there is a need for the entities that were involved to provide feedbacks on the use of the service. In this perspective both service provider and consumer undergo a ***Feedback Creation*** step that will represent their general evaluation of the service consumption for their respective perspectives. In addition to this evaluation, the Service Monitoring and Service Auditing activities will produce reports through the ***Monitoring Reporting*** and ***Audit Reporting*** steps. The main difference between feedbacks and reports is to be found in the fact that reports is only based on the measured aspects of the service consumption, while feedbacks can include, in addition with information on measured aspects, information on not-measured aspects (whether they are positive or negative feedbacks).

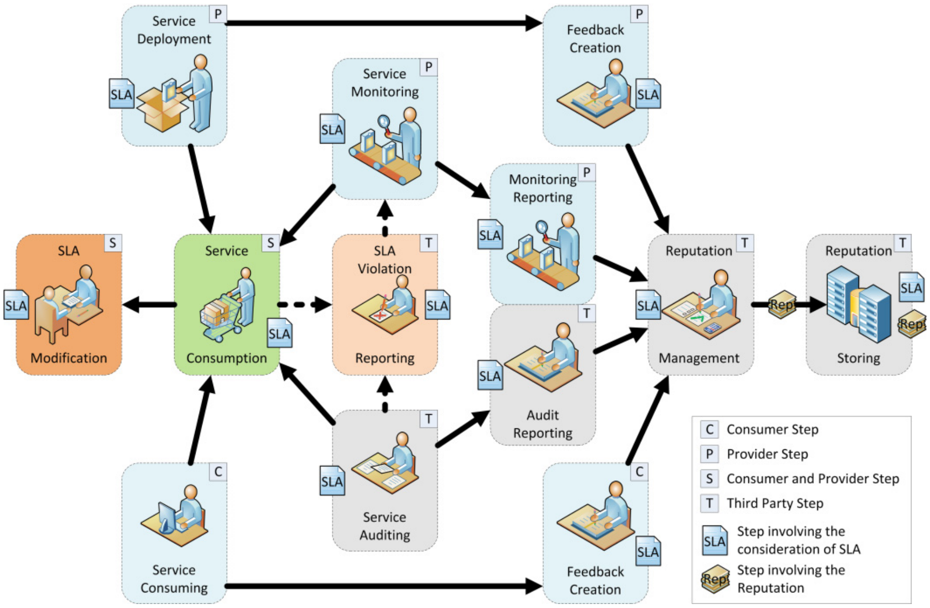


Fig. 5. Cloud Service Evaluation Phase

All those reports and feedbacks have to be the source of the reputation, and consequently have to be sent to a ***Reputation Management*** system in order for it to infer information about the reputation of the participating entities. Once processing of this information is done, it can proceed to the ***Reputation Storing*** step which will allow retrieving reputation information on a later use.

On this final phase of the Service life, we can see that *Reputation* will be processed and stored for later use. Indeed, as we have seen in the initial phases, reputation will be used during the service search, discovery, selection and SLA negotiation steps.

3.7 SLA-Based Reputation Life Cycle

Now that we have seen the details of service-oriented computing in the light of SLA and Reputation, we can extract a minimalistic life cycle for the SLA-Base life-cycle. As depicted by Fig. 6, the core life cycle behind our research is simple. Starting from the SLA signed by the parties, it constitutes the basis of the Feedbacks that will be given afterwards. Thus we can say that SLAs are characterizing feedbacks. On the feedbacks have been created they are aggregated and processed to feed the Reputation of the service. While we could think that the reputation is the final “form” of the influence of SLA, we have, in fact, to consider the fact that the Reputation of a Service drives a part of the establishment of future SLA. Indeed, if you now that a service performs poorly on a given SLA parameters, it is probable that you will try to reach an agreement on this parameter to avoid too many “surprises” during the use of the service.

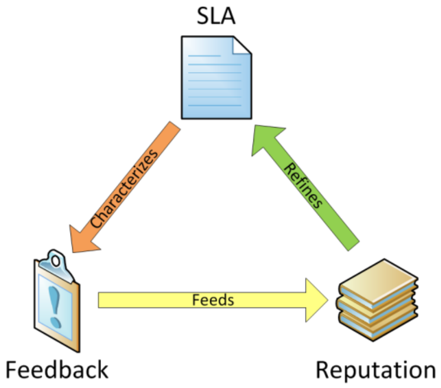


Fig. 6. SLA-Based Reputation Core

Table 1 summarizes the different key points that we have identified along our analysis of the Reputation life-cycle. The driving role of the SLA appears clearly and will help us in our future work to determiner actions to undertake.

Table 1. SLA and Reputation Key Points

Phase/Step	Entity	Element	Key Point
Service Discovery			
Service Creation	Provider, Subcontractor	Reputation	• Inheritance of Reputation from subcontracted services
		SLA	• Transitivity of SLA (SLAs signed with subcontracted services fix boundaries *de facto* to the establishment of the SLA)
		Reputation & SLA	• SLAs previously signed with external (subcontracted) services have to be considered for the reputation and the design of the SLA of the service.
Service Description	Consumer	SLA	• Drafting of SLA

Table 1. (*Continued*)

	Service Recommendation	Consumer, Broker	Reputation	• Reputation retrieval, filtering, customization
			Reputation & SLA	• Reputation has to be considered according to the intended SLA.
Service Preparation				
	SLA Negotiation	Provider, Subcontractor	SLA	• Consider previously signed SLAs • May renegotiate SLA with sub-contractors
		Consumer	Reputation & SLA	• Has to consider reputation of SLA parameters in the negotiation.
		Third Party	SLA	• Take part to the negotiation, for instance for the monitoring activity
	SLA Signing	Consumer, Provider, Third Party	SLA	• Sign the SLA, creating the legal contact
	Service Customization	Provider	SLA	• Tailor the service and its adaptation features according to the parameters of the SLA.
Service Use				
	Service Consumption	All parties	SLA	• Has to be done in respect of SLA
	Service Monitoring	Provider	SLA	• Verify the respect of the SLA and the respect of provider-specific elements
	Service Auditing	Consumer, Third Party	SLA	• Verify the respect of the SLA parameters.
	Service Adaptation	Provider, Subcontractors	SLA	• Ensure the respect of the SLA • Adapt the service execution environment according to the threats to the SLA and internal needs of the provider. • Adapt/Renegotiate with subcontractors to ensure the SLA
	Service Deployment	Provider	SLA	• Service has to be deployed in respect of the SLA parameters
	Service Reporting	Consumer, Third Party	Reputation & SLA	• Create audit reports on the respect of SLA, that will be used later to evaluate the service reputation.
	SLA Violation Reporting	Any party	Reputation, SLA	• The violation of SLA has a direct impact on the Reputation of a service and its provider.

Table 1. (*Continued*)

	SLA Modification	Consumer, Provider	Reputation, SLA	• Any request for the modification of the SLA implies a modification of the relation between the parties, and then a modification of the reputation. It will also be the start of the life of a new SLA
Service Evaluation				
	Feedback Creation	Consumer, Provider	Reputation, SLA	• When creating a feedback for the use of the service, it has to be done by taking into account the SLA. • The reputations of both consumer and providers are directly related to the feedback provided.
	Monitoring Reporting	Provider	Reputation, SLA	• Reports generated by the monitoring are done according to the SLA • Those reports will influence the Reputation.
	Audit Reporting	Consumer, Third Party	Reputation, SLA	• Reports generated by the auditing are done on the basis of the SLA • Those reports will influence the Reputation.
	Reputation Management	Reputation Manager	Reputation, SLA	• The reputation management has to be done in accordance with the SLA as it provides a basis against which the service evaluation has been done.
	Reputation Storing	Reputation Manager	Reputation, SLA	• The storage of feedbacks has to be done with a reference to the SLA

4 Discussion

In this paper we propose our conception of an SLA-Based reputation management for service-oriented and cloud computing. This conception relies on the simple principle that SLA can drive many activities during the whole life of the service, and more particularly can serve as a reference and a base for the production of feedbacks that will then feed the Reputation of the service. Management of Reputation is a hard task in open environments where any entity can enter the system and offer a service with more or less goodwill. With the new challenges and opportunities offered by Cloud Computing, it is more than ever needed to develop a robust method and mechanisms allowing a reliable collaboration between involved parties. In this perspective, our approach proposes to reuse the established process of SLA negotiation and signing. This reuse of SLA has to be completed by building a semantically coherent and meaningful ontology (inspired by work already done on this matter [7] and with the perspective of a complete Cloud ontology [8]) to help the design, understanding and negotiation of SLAs. This ontology would also ensure the coherence and the common ground on which SLAs are built and then it will ensure the coherence of Feedbacks, and finally Reputation.

One clear potential issue in our conception is a need for a relatively complex a highly descriptive SLA. This issue is a “natural” problem of SLA and some really promising researches are already trying to solve it ([6], [10], [11]) by simplifying or guiding the design and development of SLAs.

Many researches in the domain of Trustworthiness of services are considering primarily the Trust of an entity in another. From our point of view, the concept of trust is either insufficient, or not correctly employed. Indeed, the Reputation mechanism we rely on is using feedbacks from the different entities implied in the Cloud ecosystem the build the Reputation of entities. Then, the trust of one entity in another could be described as a sum of the feedback it provided on that second entity. And finally reputation could be considered as the consolidation of all trust of all entities of the system regarding the concerned entity. In this conception the trust of on particular entity in another becomes a simple part of the global reputation. That’s the reason why we haven’t explored more deeply the representation of trust; it is implicitly “included” in the Reputation.

5 Conclusion

Management of Reputation for Service-Oriented Computing has been a hot subject for many years now. The fast emergence of the Cloud Computing has generated new challenges and requires an even deeper consideration of this reputation as it may be one of the only ways to make a difference between two providers offering the same service at the same cost. Moreover, in a wider and wider network of entities collaborating more and more freely, it becomes unavoidable to be able to get information about providers in order to avoid bad surprises. In addition, we argue that providers are not the only ones that need to be evaluated. Service consumers for instance are not “external” to the system, they also need to do their part and have to correctly use services. Given this consideration, it becomes natural that reputation has to be envisaged as a characteristic of services themselves, but also a characteristic of providers of these services (some elements of the reputation have no direct relation with the service, but with the management of the service; for instance service scalability may partially be a result of the service design, but in many case it will more probably be a result of the capacity of the service provider to correctly balance load and respond to the evolution of service use). Reputation is also a characteristic of consumer of the service (as evoked, the way the service is integrated and used within the system of the consumer can lead to various results and may induce a lack of efficiency compared with the actual performance of the service). The case of third parties is a bit different, as it involves the consideration of separate SLAs. However, it is possible that in the future, third parties involved in the service life cycle will be integrated more deeply into the SLA.

In the end, SLAs have to be considered during the whole life of the service, as multiple entities, steps and activities can use them to calibrate, customize and optimize service delivery, service management, monitoring, reporting and even subcontracting. Given the natural benefits offered by SLA in the Reputation life cycle, it is clear for us that we will continue working on this perspective. The future of our research will concentrate on the description of an ontology for SLA, Feedbacks and Reputation, allowing the seamless understanding of the information, from the beginning of the service discovery, until the very last evaluation steps.

Acknowledgements. The research presented in this paper is supported by the European Union within the FP7 Marie Curie Initial Training Network "RELATE".

References

1. Alhamad, M., Dillon, T., Chang, E.: SLA-Based Trust Model for Cloud Computing. In: Proceedings of the International Conference on Network-Based Information Systems, Barcelona, Spain, September 14-16, pp. 321–324 (2010)
2. Ron, S., Aliko, P.: Service level agreements. Internet NG. Internet NG project (1999-2001), `http://ing.ctit.utwente.nl/WU2/`
3. Conway, G., Curry, E.: Managing Cloud Computing: A Life Cycle Approach. In: 2nd International Conference on Cloud Computing and Services Science (CLOSER 2012), Porto, pp. 198–207 (2012)
4. Jiang, M., Byrne, J., Molka, K., Armstrong, D., Djemame, K., Kirkham, T.: Cost and Risk Aware Support for Cloud SLAs. In: Proceedings of the International Conference on Cloud Computing and Services Science (CLOSER 2013), Aachen, Germany, May 8-10, pp. 207–212 (2013)
5. Moltkau, B., Thoss, Y., Schill, A.: Managing the Cloud Service Lifecycle from the user's View. In: Proceedings of the International Conference on Cloud Computing and Services Science (CLOSER 2013), Aachen, Germany, May 8-10, pp. 215–219 (2013)
6. Spillner, J., Illgen, S., Schill, A.: Engineering Service Level Agreements A Constrained-domain and Transformation Approach. In: Proceedings of the International Conference on Cloud Computing and Services Science (CLOSER 2013), Aachen, Germany, May 8-10, pp. 395–405 (2013)
7. Patel, P., Ranabahu, A., Sheth, A.: Service level agreement in cloud computing. In: Cloud Workshops at OOPSLA (2009)
8. Youseff, L., Butrico, M., Da Silva, D.: Toward a unified ontology of cloud computing. In: Grid Computing Environments Workshop, GCE 2008, pp. 1–10. IEEE (2008)
9. Wang, Y., Vassileva, J.: A review on trust and reputation for web service selection. In: 27th International Conference on Distributed Computing Systems Workshops, ICDCSW 2007, p. 25. IEEE (2007)
10. Keller, A., Ludwig, H.: The WSLA framework: Specifying and monitoring service level agreements for web services. Journal of Network and Systems Management 11(1), 57–81 (2003)
11. Rady, M.: Parameters for Service Level Agreements Generation in Cloud Computing. In: Castano, S., Vassiliadis, P., Lakshmanan, L.V.S., Lee, M.L. (eds.) ER 2012 Workshops 2012. LNCS, vol. 7518, pp. 13–22. Springer, Heidelberg (2012)

Towards a Self-protecting Cloud

Anthony Sulistio[1] and Christoph Reich[2]

[1] Dept. of Service Management & Business Processes, HPC Center Stuttgart (HLRS), Germany
[2] Cloud Research Lab, Hochschule Furtwangen University, Germany
`sulistio@hlrs.de, rch@hs-furtwangen.de`

Abstract. Small and Medium-sized Enterprises (SMEs) are the backbone of a nation's economy. In order to remain competitive and become a market leader in this globalization era, SMEs need to look into innovative IT solutions. One of these IT solutions is cloud computing, where it delivers Infrastructure, Platform, and Software as a Service (IaaS, PaaS, and SaaS) on a simple pay-per-use basis. Nevertheless, enterprises are sceptical about adopting the cloud, because of the risks involved in outsourcing services into the cloud.

To reduce the barrier for SMEs, this paper proposes a solution towards a two step service migration approach, 1) pre-defined Service-Level Agreement templates and 2) risk analyses, combined with a *self-protecting* cloud service that monitors, for example, the pre-defined privacy regulations and privacy protection measures.

Keywords: self-protecting, cloud, privacy.

1 Introduction

Cloud computing enables SMEs to avoid over-provisioning of IT infrastructure and training personnel to become experts in administration of IT infrastructures. Further, SMEs can take advantage of cloud computing if the IT capacity needs to be increased on the fly. This increase is typically needed for services which are only requested for a certain period of time. By leveraging cloud computing, SMEs can also reduce time to market for new products and services.

Although cloud computing provides the aforementioned advantages, SMEs are quite slow in adopting cloud solutions into their existing production system and/or migrating their applications to the cloud. Barriers towards the cloud are complexity of their services, integration problems, cloud specific Service Level Agreements as well as security and privacy concerns [1][2]. SMEs do not have the necessary resources and expertise in cloud migration and choosing which cloud technologies are most suitable for their needs [3]. Moreover, questions have been raised about the privacy of data stored in the cloud and how cloud providers could guarantee them, as stated in [4]. In [5], some of the identified privacy issues for cloud computing are data access, data breaches and data recovery. Businesses are responsible for the data that will be outsourced, and therefore must know the type of data to be outsourced (e.g., personal data, confidential data), the legal regimes their outsourcing providers are subject to, the risk of losing data and the quality of the outsourced service during the decision of outsourcing a service into the cloud. Gathering this information can be difficult and time-consuming, if the providers do not make public all the situations in which personal data is made available to government or law enforcement bodies (e.g., with respect to the US Patriot Act). The lack of clarity may prompt some companies to bring data back under their direct control.

To address these aforementioned problems, this paper contributes by introducing the following approaches. First, to reduce the barrier for service migration into the cloud a two stage migration strategy is introduced with a) pre-defined Service-Level Agreement (SLA) templates and b) a risk analyses. Second a *self-protecting* cloud service monitors the specified privacy issues and privacy protection measures.

The rest of this paper is organized as follows. Section 2 mentions the related work. Section 3 describes the cloud migration strategy in details, whereas Section 4 explains the self-protecting cloud service. Finally, Section 5 concludes the paper and provides future works.

2 Related Work

This section first starts with the privacy-related standards, regulations, and policies. Then, this section discusses papers stating important issues of privacy if services are outsourced, and presents an overview of a privacy preserving in cloud computing.

There are several legislations for data protection that need to be considered, which depend on the data domain, and the countries of the cloud customer and the cloud provider are based on. The US takes a sector-based approach (e.g. Children's Online Privacy Protection Act (COPPA) or Health Insurance Portability and Accountability Act (HIPPA)), whereas in Asian countries like Singapore, Hong Kong, and Malaysia case-based emerging laws are relevant. Other countries like Canada and Argentina have their own special data protections laws. There are many similarities among these laws due to the OECD guidelines [6] on these issues.

The longest established and strictest data protection laws in effect are the one enacted by the European Union (EU). The Data Protection Directive 95/46/EC [7] applies throughout the EU. Currently at the time of the writing, the European Commission works on a proposal for a General Data Protection Regulation [8], that is likely to take until 2016. One of the key themes of the proposed regulation is accountability (i.e., taking responsibility for your data processing), the right to be forgotten, notification of data breaches to the regulators and data protection officers for SMEs that employ more than 250 people.

According to Sotto et al. [9], storing data in the cloud may elicit various federal and state privacy and data security law requirements, such as US HIPPA and EU Data Protection Directive. For example, Articles 25 and 26 of the EU Data

Protection Directive prohibit transfers of personal data to countries outside of European Economic Area (EEA), unless these countries have an adequate level of data protection [10]. Thus, privacy and data security laws present a significant challenge for cloud providers to comply with. As a result, the cloud providers are unable to provide SMEs with an assurance.

Betge-Brezetz et al. [11] proposes a solution where cloud users enabled to control their data storage, processing and movement in the cloud. The approach is based on a paradigm of sticking policies to data and encapsulating sensitive information in a Privacy Data Envelope (PDE) structure before uploading it in the cloud.

A general discussion about privacy issues can be found in [12], where the main privacy issues in data outsourcing are addressed, ranging from data confidentiality to data utility. It is also illustrated which main research directions are being investigated for providing effective data protection to externally stored data and as well as how to enable querying on such data.

Recently, there have been several publications discussing data privacy in relation to cloud storage. Joseph et al. [13] discuss key privacy issues in the context of cloud computing, and analyze the various works being done to solve the issues in privacy, in order to ensure privacy of outsourced data on cloud storage. Anil et al. [14] survey and compare different types of techniques used to enhance the security of data stored in cloud environment.

There are several different approaches addressing the privacy-preservation problem. Sayi et al. [15] tackle the issue by proposing a vertical fragmentation to a relation, where the fragment that is assigned to the cloud server contains maximum data without violating privacy. Privacy is expressed in terms of a set of confidentiality constraints and represented as a graph where the nodes are the attributes and the links represent paired confidentiality. With the graph coloring approach, the amount of data stored and/or the workload at the data owner such that there is no violation of data confidentiality is minimized.

On the other hand, a privacy-preserving access control mechanism is proposed by Jiazhu et al. [16] to provide a guaranteed secure access to outsourced data while preserving privacy of access control policies and users IDs.

Wang et al. [17] states requirements for secure third party auditor: 1) the auditor should be able to efficiently audit the cloud data storage without demanding the local copy of data 2) the auditing process should bring in no new vulnerabilities towards user data privacy.

3 A Cloud Migration Strategy

From the related work complex issues regarding data privacy and protection arise, which significantly hinder the cloud adoption rate by SMEs. In order to reduce the barrier for adopting and migrating applications to the cloud, a two-stage migration strategy is proposed. In the first stage the customer can choose from domain specific Service-Level Agreement (SLA) templates, and then perform a risk analysis in the second stage.

The first stage of the migration strategy is to allow the SMEs to choose a pre-defined domain specific SLA template that is suitable for their applications, such as in the area of e-Learning, Product Lifecycle Management, e-Science, SaaS, etc. For example, if a user selects the e-Science criteria, a specific template for e-Science applications like Computational Fluid Dynamics (CFD) and Finite Element Method (FEM) is available to choose, as shown in Figure 1. This SLA template will eventually correspond to an image template with specific software packages or libraries and configurations when running on the cloud.

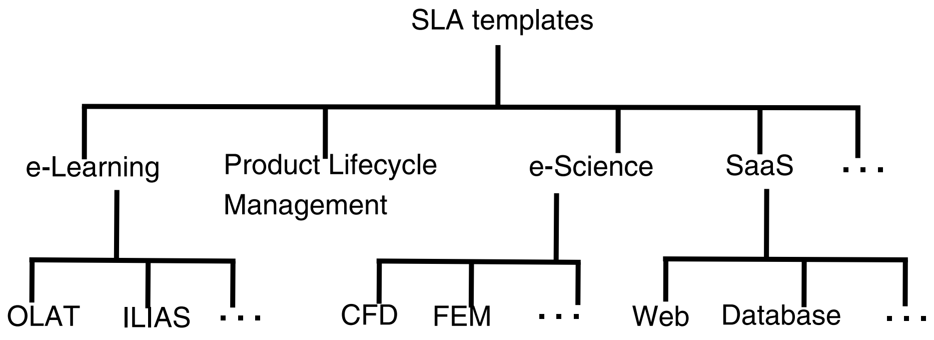


Fig. 1. Pre-defined SLA templates

After choosing the SLA template, a comparison table of various cloud providers will be displayed, as shown in Table 1. This table shows how well this application will run and what risks are considered. The SLA templates are living documents that constantly need to be updated and improved. Thus, for each criterion, an automated performance evaluation tool needs to be used across all cloud providers. Moreover, to avoid vendor lock-in, cloud providers that provide interoperability will be considered.

The second stage of the migration strategy aims to provide a detailed risk analysis of the chosen SLA template, by providing a *what-if* scenario. The risk analysis is based on existing security catalogs (e.g., ISO 27001), and considering several factors like factual knowledge (e.g., the location of the cloud infrastructure),

Table 1. An example of a comparison table for various cloud providers

Risk Categories	Amazon Web Services	Flexiscale	Microsoft Azure	...
Total cost / hour	4 Euros	4 Euros	5 Euros	
Security	Good	Satisfactory	Satisfactory	
Privacy	Good	Good	Good	
Best Practices (ITIL)	Level 3	Level 5	Level 2	
Interoperability	Satisfactory	Good	Good	
...				

customer surveys or questionnaires (e.g., Gartner and 451 Group), cloud samples (such as measurement of software versions, and security incidents), and best IT practice fulfillment like Information Technology Infrastructure Library (ITIL) [18] stated in levels (Level 0 = no to Level 5 = ITIL certified). Moreover, the user can modify the input SLA parameters (shown in Table 1). This approach will greatly facilitate the migration to the cloud because it allows an overview of the total costs, functionalities and associated risks to the SMEs.

The data privacy risk is expressed in three levels (i.e., Good, Medium, Bad) and evaluated according to historical data (e.g., data breach incidences) while considering used cryptographic methods for data transfer and storage. If the suitable provider (the one with the best fitting SLA with a minimum risk) is found, a privacy self-protecting service will be deployed on the chosen cloud provider to protect the data privacy.

The next step is to protect the migrated service during runtime by self-protection.

4 A Self-protecting Cloud Service

After a cloud provider has been selected, privacy needs to be monitored and the appropriate actions shall be performed when privacy incidences occur. This can be achieved by a *privacy self-protecting* cloud service with a control system that performs independent compliance checks of external and internal regulations, as shown in Figure 2. The control system leverages the MAPE-K (Monitor, Analyze, Plan, Execute, and Knowledge) reference model [19]. The key characteristics of autonomic computing are self-healing, self-configuring, self-optimizing and self-protecting [20]. The focus in this work is on the self-protection of a system, where systems safeguard the private or confidential information.

In the MAPE-K Loop reference model, the privacy related criteria are monitored during the Monitor and Analyse phases, using a knowledge base containing various information, such as data access rules, etc. and the automated privacy measures (like anomaly metric) are implemented in the Plan and Execute phases.

The *Privacy API* module contains a sensor and an executor, as illustrated in Figure 2. The sensor provides an input of sensor data for the Monitor phase in the control system, whereas the executor enables security actions in order to protect the customer application. For this purpose, an interconnected artificial neural network of the self-protecting *Service* module is installed to detect a potential security breach, where the same customer is running several services in the same cloud environment. Moreover, through an interconnected network the anomaly detection rate will be improved over time as more users using the service.

Through a web browser or calling directly via web services of the control system's API, the knowledge base can be updated at any time, and the selected security parameters can be monitored visually, as depicted in Figure 2. Moreover, in this figure, the *Enterprise Integrator* module enables the integration of cloud services into the user's business infrastructure through a Virtual Private Network (VPN) tunnel. As a result, this approach is intended to ensure the integrity, availability and confidentiality of the data and functions of the cloud service.

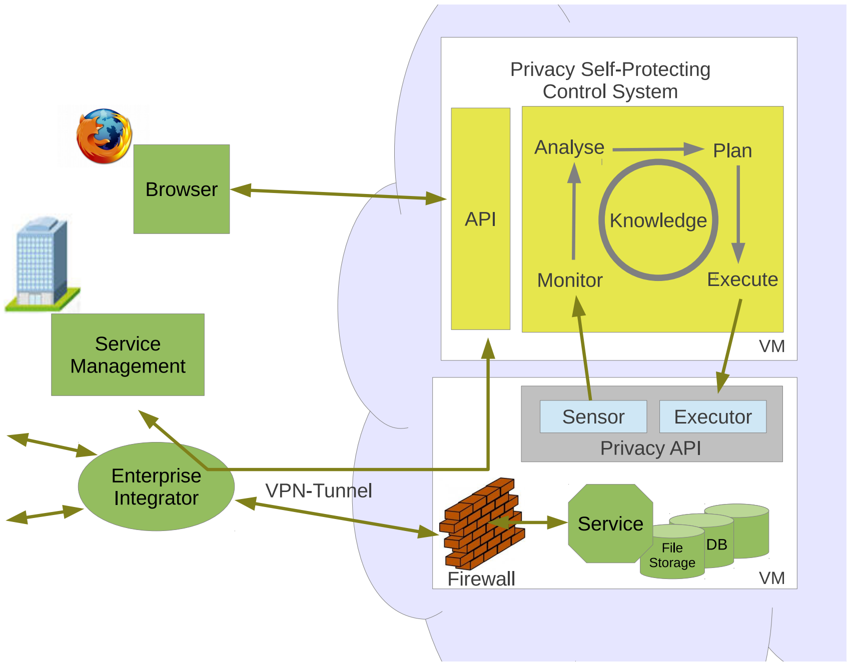


Fig. 2. The Architecture Overview

Towards Self Protecting Privacy

For private or confidential data safeguarding, companies define privacy policies that comprises various privacy rules, such as:

- only specific group has access to the data,
- gives alarm if data breaches occurred,
- data location only at predefined region,
- only encrypted data should be stored, and
- access of data only by a specific service.

Each of these rules specify on how various pieces of information need to be handled, and are individually defined by each cloud customer. The cloud's multi-tenancy character needs a scalable and effective implementation of these rules. It is essential to have in place a system for an automated enforcement of various privacy rules. The MAPE-K loop is therefore designed as following:

- *Knowledge:* The knowledge base contains the tenant-based privacy protection policies. Also, the knowledge about policy enforcement tools has to be known. It is intended to use existing tools (e.g., Intrusion Detection System or Intrusion Prevention System) and integrate them, if a policy rule implies it. Moreover, the knowledge base has information such as data access rules, log file formats, service configurations, cloud-specific facts (e.g.,

virtualization technology, firewall API), etc. which is needed for the analysis and planing phases.
- *Sensor/Actor:* Application Programming Interface (API) transformation between the tools and the phases (Analysis and Planning). For each existing tool, there will be one specific sensor/actor.
- *Analysis:* The data of the tools are analyzed and the appropriate plan is triggered. For example, a tool signals data breach and unauthorized access. As a result, it will trigger an alarm for data leakage.
- *Plan:* There are different executable plans, such as different notification levels (info, alarm), firewall reconfiguration (block traffic), data restore, etc.

5 Conclusion and Future Work

This paper proposes a solution towards a *self-protecting* cloud to reduce the barrier for cloud adoption by Small and Medium-sized Enterprises (SMEs). This can be done by having two approaches, i.e. by having a migration strategy with pre-defined Service-Level Agreement (SLA) templates, and a self-protecting cloud service that monitors the specified security criteria and security measures.

In order to find the "suitable" cloud environment for IT services, an SLA analysis of various cloud providers are evaluated and analyzed by using pre-defined SLA templates and service-specific SLA specifications. Moreover, a detailed risk analysis is used to determine an overview of the total costs, the risks and the long-term benefit to SMEs.

After a cloud provider has been selected, a *self-protecting* cloud service with a *self-securing* control system oversees safety criteria compliance and performs independent safety measurements. This approach is intended to ensure the integrity, availability and confidentiality of the data and functions of the cloud service.

As for future work, an implementation of the self-protecting cloud service will be done as a proof of concept.

References

1. CISCO Systems Inc.: Planning the migration of enterprise applications to the cloud, `http://www.cisco.com/en/US/services/ps2961/ps10364/ps10370/ps11104/Migration_of_Enterprise_Apps_to_Cloud_White_Paper.pdf`
2. Jansen, W.A.: Cloud hooks: Security and privacy issues in cloud computing. In: HICSS, pp. 1–10. IEEE Computer Society (2011)
3. MacInnes, B.: Leading the SME to cloud, `http://www.microscope.co.uk/feature/Leading-the-SME-to-cloud`
4. Ren, K., Wang, C., Wang, Q.: Security challenges for the public cloud. IEEE Internet Computing 16(1), 69–73 (2012)
5. Winkler, J.R.V.: Securing the Cloud: Cloud Computer Security Techniques and Tactics. Syngress Publishing (2011)
6. OECD: OECD Guidelines on the Protection of Privacy and Transborder Flows of Personal Data. OECD Publishing (February 2002)

7. Data Protection Directive 95/46/EC: Data Protection Directive 95/46/EC, `http://europa.eu/legislation_summaries/information_society/data_protection/l14012_en.htm`
8. European Commision: General data protection regulation, `http://ec.europa.eu/justice/data-protection/document/review2012/com_2012_11_en.pdf`
9. Sotto, L.J., Treacy, B.C., McLellan, M.L.: Privacy and Data Security Risks in Cloud Computing. In: Electronic Commerce & Law Report (February 2010)
10. Council Directive 2002/58/EC: On the protection of individuals with regard to the processing of personal data and on the free movement of such data. Official Journal L 281, pp. 31–50 (October 24, 1995)
11. Betge-Brezetz, S., Kamga, G.B., Ghorbel, M., Dupont, M.P.: Privacy control in the cloud based on multilevel policy enforcement. In: 2012 IEEE 1st International Conference on Cloud Networking (CLOUDNET), pp. 167–169 (2012)
12. Samarati, P., di Vimercati, S.D.C.: Data protection in outsourcing scenarios: issues and directions. In: Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security, ASIACCS 2010, pp. 1–14. ACM, New York (2010)
13. Joseph, N.M., Daniel, E., Vasanthi, N.A.: Article: Survey on privacy-preserving methods for storage in cloud computing. In: IJCA Proceedings on Amrita International Conference of Women in Computing - 2013 AICWIC(4), pp. 1–4. Foundation of Computer Science, New York (2013)
14. Anil, S.L., Thanka, R.: A survey on security of data outsourcing in cloud. International Journal of Scientific and Research Publications (IJSRP) 3 (2013)
15. Sayi, T., Krishna, R., Mukkamala, R., Baruah, P.K.: Data outsourcing in cloud environments: A privacy preserving approach. In: 2012 Ninth International Conference on Information Technology: New Generations (ITNG), pp. 361–366 (2012)
16. Jiazhu, D., Shuangyan, L., Hongxia, L.: A privacy-preserving access control in outsourced storage services. In: 2011 IEEE International Conference on Computer Science and Automation Engineering, CSAE, vol. 3, pp. 247–251 (2011)
17. Wang, C., Wang, Q., Ren, K., Lou, W.: Privacy-preserving public auditing for data storage security in cloud computing. In: Proceedings of the 29th Conference on Information Communications, INFOCOM 2010, pp. 525–533. IEEE Press, Piscataway (2010)
18. ITIL: ITIL - IT Infrastructure Library, `http://www.itil-officialsite.com/home/home.aspx`
19. Menascé, D.A., Kephart, J.O.: Guest editors' introduction: Autonomic computing. IEEE Internet Computing 11(1), 18–21 (2007)
20. IBM: An architectural blueprint for autonomic computing, `http://www-03.ibm.com/autonomic/pdfs/AC%20Blueprint%20White%20Paper%20V7.pdf`

Cloud Audits and Privacy Risks

Thomas Rübsamen and Christoph Reich

Cloud Research Lab,
Furtwangen University of Applied Science,
D-78120 Furtwangen, Germany
{thomas.ruebsamen,christoph.reich}@hs-furtwangen.de

Abstract. In cloud computing users are giving up control over resources such as storage. Lacking transparency of cloud services (e.g. data access and data lifecycle reports) is an important trust issue, that hinders a more wide-spread adoption of cloud computing. Giving the customer of cloud services more information about data usage, compliance test reports and accordance to best-practices make the cloud more transparent. Reporting about such verifications is the main objective of cloud audits and is performed by third party auditors (TPAs). However, public auditing by TPAs can introduce new privacy problems. In this paper, a survey of current cloud audit privacy problems is given and techniques are shown how they can be addressed. Also, requirements for a privacy-aware public audit system are discussed.

Keywords: Audit, Privacy, Cloud Computing, Transparency.

1 Introduction

Cloud computing as a paradigm is becoming more important for today's information technology. The shift from using traditional in-house datacenters towards on-demand provision of resources for delivering services has provided significantly more flexibility. However, a major aspect of adopting the cloud is giving up control of resources such as servers and network infrastructure. This also implies giving up control of data, which is stored in the cloud as well as services running in the cloud. The benefits provided are: increased flexibility, on-demand scalability and sometimes cost advantages.

Large companies like Amazon, Google and Microsoft have recognized the potential business opportunities and are providing well-established cloud services. Depending on the cloud service model, Infrastructure as a Service (IaaS), Platform as a Service (PaaS), or Software as a Service (SaaS) [1] cloud customers have different levels of control over their resource stack for providing and using their services. For instance, a customer does have more control of their resources in a IaaS compared to a SaaS resource. Nevertheless some parts of the cloud infrastructure always remain hidden from the customer. This leads to a lack of transparency of cloud services, which is in turn the most important reason for a lack of trust in cloud computing. Highly visible security and privacy incidents of major cloud providers also make this problem worse, e.g., as reported by CNN: "Google fires engineer for privacy breach" [2].

The Cloud Security Alliance (CSA) launched a new initiative to encourage transparency of security practices within cloud providers, called Security, Trust & Assurance Registry (STAR) [3]. Providers are encouraged to provide self assessment reports using STAR.

Some cloud providers address the transparency problem by providing tools such as Amazon's dashboard [4], but to increase trust cloud providers need to be able to prove compliance with external and internal regulations. Cloud audits are one instrument to provide cloud customers with more insight on cloud service delivery. They increase trust in cloud providers by proving the provider's compliance to regulatory requirements (SAS70, HIPAA, PCI) and industry best-practices (ISO27001, BSI Grundschutz, ITIL) regarding the implementation of privacy controls. Cloud audits may be characterized from different perspectives. They can be performed internally by the cloud provider or externally by a trusted third-party auditor. Additionally, audits may also be performed by the customer (e.g., public audit), by providing him with the necessary interface to conduct them.

However, cloud audit capabilities themselves can be problematic in multi-tenant environments such as the cloud. By providing an auditor access to cloud infrastructures, the customer's data might be at risk. In case of an untrustworthy auditor (third-party or internal) sensitive information is potentially at risk. The Privacy Rights Clearinghouse published such an incident, where an untrustworthy auditor used his access privileges to sell customer's credit card information obtained from the Amsterdam Hospitality Group [5]. Such an incident may also happen in cloud environments. Information about other tenants might also be leaked, when a customer is given the capability to audit the provider's services himself. Additional protection mechanisms beyond contractual terms are needed in cloud audit scenarios. Another problem arises from making the cloud more transparent in terms of data locality and audits of policies, which require certain types of data to be stored at distinct locations. However, tracking of data in the cloud is not the focus of this paper.

In this paper, we focus on the privacy implications of providing publicly available audit capabilities to cloud customers. In Section 2 we define transparency, privacy and cloud audit concepts, followed by Section 3 where we provide a survey on privacy issues, which stem from public cloud audits as well as some ways to address these issues. After that, requirements for a privacy-aware public audit system are discussed in Section 4. We end this paper with a conclusion in Section 6.

2 Transparency, Privacy and Cloud Audit

The dilemma is to provide the highest levels of transparency while maintaining strong privacy and being auditable by a third party. In this section, we describe the concepts of transparency, privacy and cloud audit.

2.1 Transparency

Transparency in the sense of visibility is an important issue in cloud computing. According to the NIST definition of cloud computing one essential characteristic of cloud computing is measured service [1]. Resources are monitored, controlled and reported on continuously. This provides some degree of transparency for the user and provider. However, this information might not be enough for showing *what* happens in the cloud. Additionally, technical (e.g. cloud architecture, configurations etc.) and operational information (information about security processes, incidence management, etc.) needs to be provided to make cloud services transparent. Also, the different cloud service delivery scenarios (IaaS, PaaS, SaaS) need to be considered. In IaaS the cloud user usually needs less information from the cloud provider, due to having full control about virtual machines at the operating system level. In PaaS and SaaS however, additional information needs to be provided by the provider because of the administration shift away from the customer.

2.2 Privacy

Privacy can be looked at from different angles. From a consumer's perspective it comprises the protection and appropriate use (according to the customer's expectations) of the customer's personal data. From an organization's point of view, privacy includes the application of laws, policies, standards and processes by which personally identifiable information (PII) is managed [6]. These aspects have to be considered carefully in cloud computing scenarios, where PII is processed and transferred.

Confidentiality is a concept, which focuses on the protection of data from unauthorized access. Usually, cryptographic mechanisms are used to achieve confidentiality. Confidentiality can be linked to privacy in the sense of protecting PII by using encryption techniques. However, in current cloud computing services, data is usually not encrypted to enable processing of data by the cloud provider. Therefore, other ways have to be found to protect customer's privacy.

2.3 Cloud Audit

Common cloud computing scenarios usually include at least one cloud service provider and a customer in the service provision chain. The introduction of a third party auditor (TPA) offloads time-consuming and cost-intensive tasks, as well as required knowledge to conduct audits from the customer. The entity, which performs audits is called cloud auditor. Generally, audits are used to "verify conformance to standards through a review of objective evidence" [7]. In cloud computing this means cloud services are examined regarding their performance, privacy, and security controls, which shall assure the cloud customer, that appropriate measures are in place. To enable the TPA to fulfill his role, public auditability becomes a requirement. The cloud provider has to provide auditors with interfaces, techniques and necessary documentation as well as any additionally needed information to conduct audits. The auditor can than leverage his expertise and conduct audits on behalf of the provider (for external audits) or the customer.

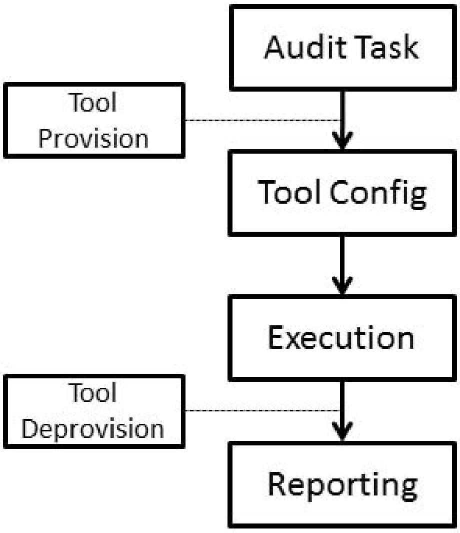


Fig. 1. Cloud Audit Process

Figure 1 depicts an automatic, tool-driven cloud audit process. Starting from a specific audit task, which describes a distinct objective, a tool is provisioned to perform the task (e.g., checking specific configuration files or parsing logs for certain events). The configuration of the tool specifies what has to be checked during execution. On the basis of the evidence collected by the tool, a report is generated and the tool is deprovisioned.

Auditors can be internal or external. External auditors are members of independent organizations specialized on performing audits. Internal auditors are usually contracted to the organization, whose entities are to be audited. They generally adhere to the same standards as external auditors with respect to performing an independent analysis.

3 Audit and Privacy

Cloud auditing needs to consider privacy of cloud customers. There can be several views on public cloud auditing scenarios, which will be shown in the following:

- Disclosure of customer's data
  Cloud customers store information in the cloud. Examples for this are cloud storage services, which are used to store files, database as a service, which

stores more structured data and data stored in SaaS services (e.g., customer relationship information).

- Disclosure of metadata about customer's data
  This kind of metadata includes data access and data usage logs, location data and data tracking information.
- Disclosure of metadata about customer
  This kind of metadata includes information about customer's usage profiles of cloud services.
- Disclosure of security relevant information
  Cloud customers may build virtual infrastructures. Auditing capabilities for such infrastructures may be provided by the cloud service provider as well as a third party. It is important to design them in a way, that no potentially harmful information is disclosed.

In this section, we provide an overview about recent research conducted in the area of privacy with respect to audits. We also give a short overview of how audits increase trust in cloud computing infrastructures.

3.1 Operational and Reliability Audit

By moving to the cloud, customer's (also owners of data) are placing control of their data into the hands of the cloud service provider. Two big arising issues are reliability and availability. Threats to these goals are for instance (silent) data corruption and (unintentional) deletion. Data corruption can have multiple causes like hardware faults in the networking or storage subsystems, or software bugs during processing of data. Deletion of data can be caused by careless provider personnel as well as software bugs. Risks evolving around these problems are for instance high costs caused by downtimes. On the provider side there is a risk of potential reputation loss. To mitigate these risks and provide the customer with an adequate level of assurance, that his data are safe with the cloud provider and stored correctly, audits can be used.

However, actually checking the correctness and retrievability of data is not a trivial task. Simply downloading data from the cloud and checking whether or not its integrity was harmed or parts are missing is not a feasible solution in cloud environments. Having to download terabytes of data and calculating checksums on that data would place a heavy burden on the customer or auditor, financially as well as in terms of resources needed.

Therefore, new ways of reliability auditing have to be found for cloud environments. There already exist several approaches, which aim at assuring data integrity and retrievability in the cloud while removing the need to download and check huge chunks of data. Most of the approaches focus on a third-party auditor (TPA) performing audits on cloud infrastructures.

Proof of data possession (PDP) [8] is a concept that uses homomorphic verifiability tags (HVTs) to prove that a remote party is in possession of a file.

This is done by generating metadata about the file prior to upload and verifying possession using a challenge-response protocol later on.

Proofs of retrievability (POR) [9] is a concept to enable an entity to verify the intactness of remotely stored data without requiring to retrieve large files. Thereby the server proves with a high probability to a data owner, that a file can be retrieved even in the case of some parts being corrupted. Most of the presented approaches leverage some kind of POR to enable reliability audits of cloud storage services [10].

In [11] Wang et al. propose a network architecture for secure data storage in the cloud. They acknowledge the previously described availability and privacy problems in current cloud storage systems. In their solution, a trusted TPA conducts audits on behalf of the data owner. They assume, that there is no incentive for the TPA to violate the privacy of the data owner. However, by proposing a solution, where the TPA is denied access to content, data leakage to the TPA shall be prohibited. Therefore, they propose the requirement, that a TPA must not know the contents of data, which is audited. Further goals of their approach are the support of dynamic data updating, batch auditing, and minimization of auditing overhead (e.g., network bandwidth). They propose cryptographic techniques like using homomorphic authenticators, and Merkle hash trees [12] to fulfil those requirements.

Boyang et al. take the concept of public auditability one step further and propose a system, where shared data stored in the cloud can be verified while preserving the identity privacy of each signer of a data block [13]. By using ring signatures to construct homomorphic authenticators the TPA is able to verify data, while not leaking identity information.

Furthermore there exist projects, that include TPAs and public auditability principles to enable secure and trustworthy cloud storage systems [14–17].

Performance auditing and the privacy issues introduced by it have been investigated to a much lesser degree. Large cloud provider such as Amazon, Google or Microsoft usually provide their customers with specialized interfaces (e.g., Amazon CloudWatch) for extracting performance monitoring information. Such information includes, but is not limited to, CPU utilization, network I/O statistics. The degree to which such information is published (e.g., on a per VM basis or details about the actual cloud infrastructure performance) is chosen by the cloud provider. Privacy problems may be linked strongly to the number of details published and the multi-tenant nature of cloud environments. Performance counters published to one customer might be used to deduce information about tenants using the same shared resources.

TPAs auditing cloud performance might need elevated privileges on the examined services (e.g., for measuring loading and saving times in a SaaS scenario). A TPA also has to assess the accounting system of cloud providers. This is usually tied very closely to the collection of usage logs and performance counters. To make a statement about the correctness of a provider's accounting processes, the TPA needs access to such information. However, without proper measures to protect the customer's privacy (e.g., proper level anonymization) a TPA can easily extract cloud usage profiles (e.g., service interaction and communication) from this information. One thing to consider is the reversal of anonymization by combining multiple sources of anonymized data.

Operational and reliability audits are used to assess a providers procedures, systems, records and activities in order to test the adequacy of controls in place.

3.2 Regulatory Compliance Audit

Another form of audits is for regulatory compliance. Some businesses require cloud service providers to be compliant to or certified against certain regulations. Prime examples of such industries are healthcare, where sensitive medical information is processed, or finance, where sensitive financial data about individual subjects are processed. Typical examples are SAS70 [18] reports or HIPAA [19], which define how such data may be processed. Furthermore, there exist ISO27001 [20] which addresses information security management and CSA CloudAudit [21], which combines several of the previously mentioned audit frameworks to address cloud specific issues. However, these kinds of audits usually contain a lot of non-automatable audit tasks such as questionnaires or interviewing experts at the provider.

Reports generated by these frameworks are often not made available to the public. Amazon for example only releases their SAS70 report, when a customer contacts the Amazon support and requests it specifically. Regarding the privacy of other customers, these reports might reveal security flaws in the provider's processes, which could be exploited.

Regulatory compliance audits are used to assess a providers compliance in order to:

– Test the adequacy of controls in place
– Verify that a provider complies with established policy
– Verify that a provider complies with operational procedures (e.g., COBIT)

3.3 Security and Information Privacy Audit

Cloud security audits are supposed to uncover flaws and vulnerabilities in cloud infrastructures and service delivery chains, for instance: reveal unauthorized access to services and data, destruction of data and denial of service (DoS). The goal of this kind of auditing is to assure an appropriate level of protection especially by following industry best-practices.

However, information that is used during a security audit usually contains highly sensitive data, such as access logs and performed actions by customers. Special care has to be taken when providing such information in a public audit system.

Security audits are used to asses a provider's security issues in order to:

– Detect breaches or potential breaches of compliance
– Detect badly configured services

Information privacy audits are used to assess a providers compliance especially with respect to customer data managing in order to:

- Verify that a provider complies with established data policy
- Verify that a provider complies with privacy acts

4 Proposed System Requirements for a Public Cloud Audit System

In this section, we propose a privacy-respecting system for public cloud audits and address the most important requirements.

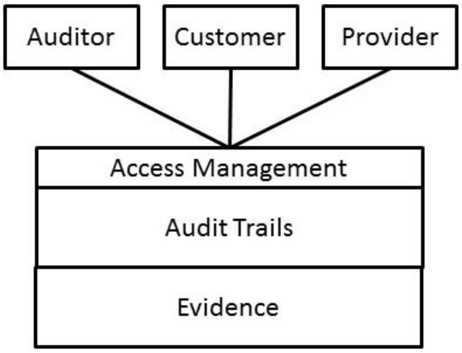


Fig. 2. Privacy-respecting Views on Cloud Audit Reports

Figure 2 depicts a high-level scenario for such a system. The foundation comprises of an evidence base. Evidence is any information needed to provide proof for specific audit tasks. This may include logs, configurations, data provenance information, timestamps, checksums and any other kind of data and metadata, which might be useful in constructing audit trails. Audit trails provide reliable proof about a certain audit task. However, as previously described audit trails might leak PII. Therefore, different views on audit reports must be provided to different stakeholders. This is done using the access management layer, which, depending on the actor requesting the audit report, provides reports with different levels of detail.

Sample Scenario: For example, consider the following scenario:

> A task for auditing the data life-cycle events of a customer's data in the cloud is requested. The sources of the relevant information are the cloud management system (CMS), which tracks high-level events such as provision/deprovision of cloud storage and the storage backend, which provides detailed information about data access, location and usage. A report is generated as a result depicting the audit result.

Mapped to the approach described, three different views on this report are provided by the audit system:

- Customer view:
  The customer view is the most detailed. It provides information about the CMS events, actions performed on the storage, such as who accessed what including timestamps, retrievability checks results.
- Auditor view:
  The auditor's view is less detailed. According to the checks defined in the audit request, the auditor is provided with more high-level results. Evidence accompanying the report is anonymized as needed.
- Provider view:
  The provider's view on the audit report contains mostly information stemming from the CMS.

Cloud Audit System Requirments: From this we derive requirements for the cloud audit system:

- Interfaces: public audit interfaces may differ among providers which complicates interoperability. In complex service provision chains, this hinders efficient auditing (see Section 5).
- Formats: differing data formats for the same information among providers also hinders interoperability. Also, tools used to extract information for specific parts of the audit trail usually use proprietary formats.
- Data collection: auditing requires the collection of data across all architectural levels. However, no more information than actually needed to provide proof for the audit task at hand shall be collected.
- Specific audit tasks: audit tasks shall focus on specific tasks. This shall enable reducing the amount of data needed to provide proof for the task.
- Dynamics: cloud environments are very dynamic. Therefore, a dynamic mechanism for collecting relevant evidence is needed. The audit system must support a mechanism to react address dynamic infrastructures.

5 Audit Challenges in Cloud Service Delivery Chains

Complex cloud service provision scenarios, where multiple service providers are chained for service composition, introduce new audit challenges. Typically, these chains are hidden from the customer. However, the customer's data are transferred along the service provision chain. For example a SaaS service provider may use an IaaS service provider's infrastructure to deliver its service. The IaaS service may be chosen depending on pricing and performance parameters and can also be exchanged without the customer's notice. This is a rather simple example for a cloud service delivery chain. More complex scenarios are thinkable, when multiple SaaS, PaaS and IaaS providers are involved.

Auditing a service and the subsequent services it depends on, introduces the following challenges:

- Audit of each involved service: each service involved in the service provision chain needs to audited.
- Audit of data transfer between services: the communication between services along the provision chain needs to be audited.
- Regulatory compliance: some parts of the provision chain may be located abroad, placing them under different jurisdictions.

6 Conclusion

In this paper we presented public cloud audit as a possible solution to increase trust in cloud computing by providing a proof and higher cloud transparency. We thereby focused on privacy concerns which arise in public cloud audit scenarios and addressed them in our proposed solution. By providing cloud stakeholders with different access views on cloud audit reports and therefore privacy of cloud customers can be protected.

In our future work, we will refine the high-level requirements of this system. Additionally, more complex service delivery chains, which involve multiple cloud service providers will be analyzed in detail to make public cloud audits practicable and service delivery chains auditable.

Acknowledgment. The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement no: 317550 (A4Cloud).

References

1. Mell, P., Grance, T.: The NIST Definition of Cloud Computing. Technical report, National Institute of Standards and Technology, Information Technology Laboratory (2011)
2. CNN: Google fires engineer for privacy breach (September 2010), `http://edition.cnn.com/2010/TECH/web/09/15/google.privacy.firing/`
3. Cloud Security Alliance (CSA): Security, Trust & Assurance Registry (STAR), `https://cloudsecurityalliance.org/star/`
4. Amazon: Amazon's service health dashboard, `http://status.aws.amazon.com/`
5. Privacy Rights Clearinghouse: Amsterdam Hospitality Group, `https://www.privacyrights.org/data-breach-asc?title=amsterdam`
6. Pearson, S., Benameur, A.: Privacy, security and trust issues arising from cloud computing. In: 2010 IEEE Second International Conference on Cloud Computing Technology and Science (CloudCom), pp. 693–702 (2010)
7. Badger, L., Bohn, R., Chu, S., Hogan, M., Liu, F., Kaufmann, V., Mao, J., Messina, J., Mills, K., Sokol, A., Tong, J., Whiteside, F., Leaf, D.: US Government Cloud Computing Technology Roadmap Volume II Release 1.0 (Draft) - Useful Information for Cloud Adopters. Technical report, National Institute of Standards and Technology, Information Technology Laboratory (2011)

8. Ateniese, G., Burns, R., Curtmola, R., Herring, J., Kissner, L., Peterson, Z., Song, D.: Provable data possession at untrusted stores. Cryptology ePrint Archive, Report 2007/202 (2007), http://eprint.iacr.org/
9. Bowers, K.D., Juels, A., Oprea, A.: Proofs of retrievability: theory and implementation. In: Proceedings of the 2009 ACM Workshop on Cloud Computing Security, CCSW 2009, pp. 43–54. ACM, New York (2009)
10. Juels, A., Kaliski Jr., B.S.: Pors: proofs of retrievability for large files. In: Proceedings of the 14th ACM Conference on Computer and Communications Security, CCS 2007, pp. 584–597. ACM, New York (2007)
11. Wang, C., Ren, K., Lou, W., Li, J.: Toward publicly auditable secure cloud data storage services. IEEE Network 24(4), 19–24 (2010)
12. Merkle, R.C.: Protocols for public key cryptosystems. In: IEEE Symposium on Security and Privacy, pp. 122–134 (1980)
13. Wang, B., Li, B., Li, H.: Oruta: Privacy-preserving public auditing for shared data in the cloud. In: 2012 IEEE 5th International Conference on Cloud Computing (CLOUD), pp. 295–302 (2012)
14. Li, L., Xu, L., Li, J., Zhang, C.: Study on the third-party audit in cloud storage service. In: 2011 International Conference on Cloud and Service Computing (CSC), pp. 220–227 (2011)
15. Patel, H., Patel, D.: A review of approaches to achieve data storage correctness in cloud computing using trusted third party auditor. In: 2012 International Symposium on Cloud and Services Computing (ISCOS), pp. 84–87 (2012)
16. Zhu, Y., Hu, H., Ahn, G.J., Yau, S.S.: Efficient audit service outsourcing for data integrity in clouds. J. Syst. Softw. 85(5), 1083–1095 (2012)
17. Shah, M.A., Swaminathan, R., Baker, M.: Privacy-preserving audit and extraction of digital contents. Cryptology eprint archive, report 2008/186 (2008)
18. SAS70: SAS70, http://sas70.com/sas70_overview.html
19. U.S. Government Printing Office: HIPAA, http://www.gpo.gov/fdsys/pkg/PLAW-104publ191/html/PLAW-104publ191.htm
20. ISO: ISO27001:2005, http://www.iso.org/iso/catalogue_detail?csnumber=42103
21. Cloud Security Alliance (CSA): CloudAudit A6 Cloud Security Alliance, http://cloudaudit.org/

On the Relationship between the Different Methods to Address Privacy Issues in the Cloud

Siani Pearson

Security and Cloud Lab, HP Labs, Bristol, UK
Siani.Pearson@hp.com

Abstract. In conjunction with regulation, information security technology is expected to play a critical role in enforcing the right for privacy and data protection. The role of security in privacy by design is discussed in this paper, as well as the relationship of these to accountability. The focus within these discussions is on technological methods to support privacy and data protection in cloud scenarios.

Keywords: Accountability, Cloud Computing, Design for Privacy, Security.

1 Introduction

Privacy in cloud business environments can be a difficult issue to tackle because of the underlying complexity across multiple dimensions and the interdisciplinary nature of the problem. For example, location matters from a legal point of view but processing flows are dynamic, global and fragmented: there are restrictions about how information can be sent and accessed across boundaries, but in cloud computing data can flow along chains of service providers both horizontally between Software and a Service (SaaS) providers and vertically, down to infrastructure providers, where the information can be fragmented and duplicated across databases, files and servers in different jurisdictions. However, data controllers still have the responsibility to ensure that the service providers are meeting regulatory obligations.

In this paper the overarching means of addressing privacy issues in cloud computing are analysed, with a focus on privacy by design, security and accountability. The relationship between such notions is a complex one that has not been sufficiently elucidated to date. This paper examines that relationship, including the following issues:

- to what extent is information security an integral part of privacy by design?
- what is the relationship of accountability to privacy by design?
- how does this apply in the cloud context?

The importance and timeliness of this analysis is underpinned both by technological and business changes embodied within the adoption of cloud computing that need to be deployed in such a way as to reduce privacy risk, as well as ongoing global regulatory changes. Notably, problems with the 1995 EU Data Protection Directive [1] as a harmonisation measure and in relation to new technologies including cloud computing have led the European Commission (EC) in January 2012 to publish a draft of replacement General Data Protection Regulation that is currently being discussed and revised [2], in which accountability features and privacy by design take greater precedence. Amongst other things, this imposes new obligations and liabilities for data processors, new requirements on data breach notification and stricter rules on international data transfers. It also empowers National Regulatory authorities to impose significantly higher fines.

The structure of this paper is as follows: section 2 highlights some key privacy challenges for the cloud and general categories of mechanism by which these may be addressed; section 3 and 4 then consider some of the key relationships between these, notably to what extent information security is an integral part of privacy by design, and the relationship of accountability to privacy by design, and how this applies in a cloud context. Finally, conclusions are given.

2 Cloud Privacy Issues: Do We Have All the Answers?

In this section key cloud-related terminology is introduced, cloud privacy issues are discussed and generic approaches to tackle these problems are introduced. What makes data processing in the cloud challenging is the rapidly expanding scale of cloud services, the pervasive role they will play in the future business and personal life, the complexity of the supply chain and the ability of advanced data mining techniques to draw inferences about data subjects from the large datasets under their control. The 'data-centric' nature of cloud computing creates a tension between service suppliers who perceive that the data they hold could be a strategic business resource and their customers who are increasingly aware of risks posed by the perceived lack of control over data in the cloud. The actual level of control in the cloud can be very variable. Transparency, remediation, clarification of responsibilities and maintenance of obligations within the supply chain are all key issues.

2.1 Cloud Computing

A definition of cloud computing that is commonly accepted is provided by the United States National Institute of Standards and Technologies (NIST): "*Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction*." [3]

There are different layers of cloud services that refer to different types of service model, each offering discrete capabilities. The service offered may be the delivery of computing resources such as storage and computing power, where the customer rents virtual machines from the service provider rather than buying hardware and software to provide these resources in the customer's own data centre (this is known as Infrastructure as a Service, or IaaS), the delivery of a solution stack for software developers (Platform as a Service, or PaaS) or the delivery of software applications available on demand and paid for on a per-use basis (Software as a Service, or SaaS). These can be layered and combined in different ways. In addition there are several deployment models for cloud computing, of which the main ones are the following:

- **Private:** a cloud infrastructure operated solely for a single organisation, being accessible only within a private network and being managed by the organisation or a third party
- **Shared:** a cloud that is open to use by selected organisations
- **Public:** a publicly accessible and shared cloud infrastructure. In such an offering the stored data of different customers will usually be logically segregated
- **Hybrid:** a composition of two or more clouds that remain separate but between which there can be data and application portability

2.2 Cloud Privacy Challenges

There are a number of privacy-related challenges in the cloud, that include aspects such as whether data handling is compliant with laws and regulations, whether data is safe across all the cloud and under control throughout its lifecycle, whether data is handled based upon users' expectations, and whether appropriate use and obligations are ensured along the processing/supply chain. In this section further elucidation is given about some of these issues, with emphasis on aspects that are exacerbated or specific to cloud.

Table 1. Cloud Features and Key Related Privacy Issues

Cloud features	Key related issues
Multi-tenancy	Data of co-tenants may be revealed in investigations, isolation failure, proper deletion of data and virtual storage devices
Complex, dynamically changing environment; data flows tend to be global and dynamic	Ensuring appropriate data protection, overlapping responsibilities in data management, unauthorized secondary usage, vendor demise, lack of transparency
Data duplication and proliferation; Difficult to know geographic location and which specific servers or storage devices will be used	Exacerbation of trans-border data flow compliance issues, detecting and determining who is at fault if privacy breaches occur
Easy and enhanced data access from multiple locations	Data access from remote geographic locations subject to different legislative regimes, subpoenas, access by foreign governments, 'idiot with a credit card'

Cloud Features and Privacy Problems. Cloud vulnerabilities are varied and are categorised in material including [4]. Table 1 highlights some cloud features and associated potential issues, many of which are at the governance level. Data duplication and proliferation (and its autonomic aspect) creates problems in terms of compliance:

Amazon for example creates up to three copies in different data centres when storing data. In addition, public cloud providers make it very easy to open an account and begin using cloud services, and that ease of use creates the risk that individuals in an enterprise will use cloud services on their own initiative, without due consideration of the risks and due governance process. There are also fears, among users, about increased access to data by foreign governments and other parties. Other issues include data lifecycle management across chains of suppliers, including data discovery and destruction, and legal risks that include security obligations, international transfers and the processing of sensitive data. For example, difficulties exist if users want to end a service, get their data deleted or export their data to another provider. Often, it is unclear who the data controller is and which parties have what responsibilities. More detailed analysis is given for example in [5,6]. In particular, loss of control and transparency (in the sense of insufficient information, thus making the task more difficult of selecting a suitable service from the vast choice of cloud offerings) are highlighted as key issues by the Article 29 Working Party [7].

A Challenge in the Enterprise Security Life Cycle. All enterprises operate a security lifecycle something like the following: assess risk associated with IT; shape investment, controls and policy choices; applications and technical procurement; work hard to configure and patch the infrastructure environment; monitor events to catch incidents and support forensics; carry out audits to see if the controls are mitigating risks. Organisations struggle to operate this cycle effectively because: technology is always changing; threats and attacks evolve faster; the cycle consists of many silos of stakeholders that have very different perspectives and expertise and do not speak the same language. For example, legal or human resources employees involved in people policies are very different from a network security expert configuring firewalls. With cloud this situation is going to get even worse, not just because there are new architectures but because the supply chain of services breaks up the activities of the security lifecycle even further (as shown in Figure 1) [8].

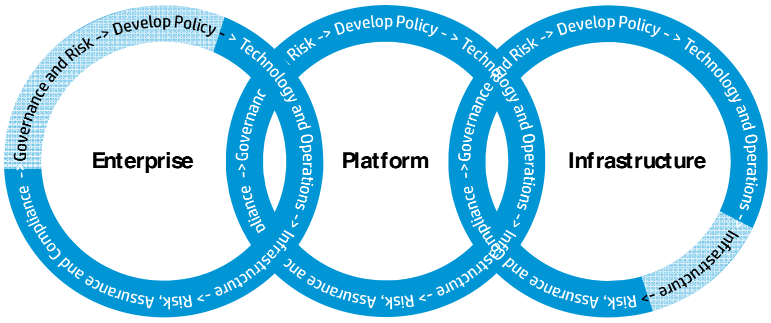

Enterprise
Platform
Infrastructure
Governance and Risk -> Develop Policy -> Technology and Operations -> Infrastructure -> Risk, Assurance and Compliance -> Governance

Fig. 1. The Need for Accountability and Transparency in the Cloud Service Provision Chain

For instance, if the enterprise buys Customer Resource Management from a SaaS provider that in turn uses an Amazon Web Service for IaaS, then the people judging risks and forming policy now have to rely on and influence the investment and monitoring choices of the SaaS provider, and are also dependent on the configuration and infrastructure purchases of the IaaS. Hence there is a need for data stewardship and accountability along the service provision chain.

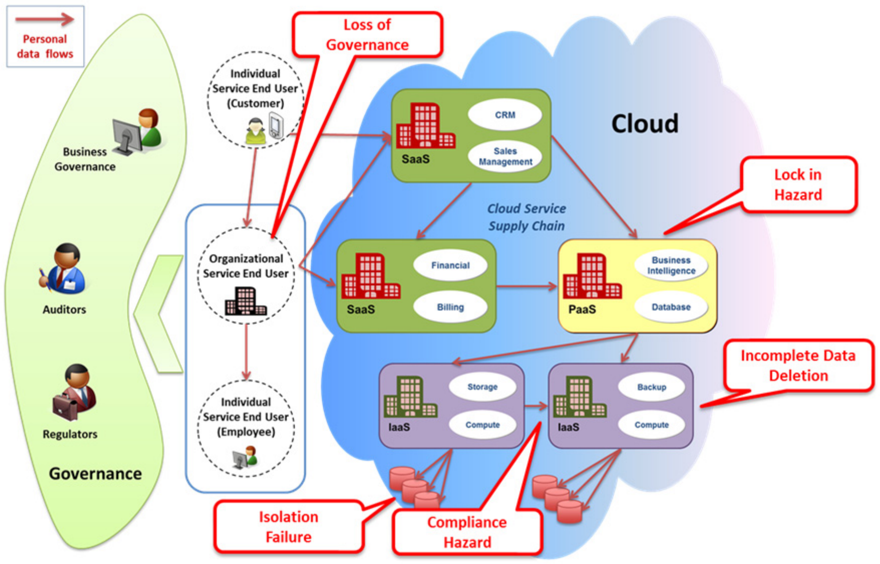


Fig. 2. Ramifications of Cloud Failures

Issues from one Cloud Service Provider (CSP) may have ramifications further up the chain, for example in terms of loss of governance. This is illustrated in Figure 2, which shows an example cloud ecosystem. Loss of governance may arise in cloud computing for example as the client cedes control to the CSP, but Service Level Agreements (SLAs) may not offer commitment to provide such services on the part of the CSP, thus giving a gap in security. There are many ways in which there can be data loss or leakage involving IaaS, PaaS and SaaS providers: for example, unauthorized parties might gain access to sensitive data due to insufficient authentication and authorization controls, or data might be stored in servers in India without appropriate governance mechanisms in place, causing a compliance hazard (for other examples see for instance [5]). Security and privacy threats of data breaches are the most severe types for cloud computing [9]. Unfortunately, some of the measures (e.g. data encryption) that can address data breaches may exacerbate data loss (e.g. if the encryption key is lost all encrypted data will be lost too). Data can be exposed to different types of security and privacy concern, and these include internal cloud facing security issues such as security attacks exploiting vulnerabilities of virtualization mechanisms and monitoring virtualized environments giving information about data usage by neighbouring users. Data breaches need to be addressed within specific provisions of

SLAs that clarify the respective commitments of the CSP and the client. An analysis of cloud failures has identified further threats (i.e. hardware failure, natural disaster, service closure, cloud-related malware and inadequate infrastructure planning) specific to cloud computing [10].

Security and Privacy Responsibilities in the Cloud. Security and privacy requirements in the cloud will vary widely from one use case to the next, and be heavily dependent upon risks and responsibilities of actors in those use cases, which again depend upon a combination of the service and deployment models used. For example, an internal private cloud can potentially offer an organisation greater oversight and authority over security and privacy, and better limit the types of tenants that share platform resources, reducing exposure in the event of a failure or configuration error in a control [11].

The NIST Cloud Computing Reference Architecture identifies the main roles in cloud computing [12]. Overall, SLAs define the respective responsibilities, although certain responsibilities are set by law, as discussed further below.

In terms of service models, the security that the consumer is responsible for will vary: the lower down the stack the cloud provider stops, the more security the consumer is responsible for implementing and managing [5]. In IaaS and PaaS, a great deal of orchestration, configuration and software development is performed by the customer, so much of the responsibility cannot be transferred to the CSP. Although the cloud provider bears most of the responsibilities for SaaS (normally being responsible for operational security processes i.e. user and access management, including identity management, authentication and compliance with data protection law), the virtual machine that contains licensed software and works with sensitive data places many more responsibilities on the consumer that builds and manages it. There is also potential for user responsibility to be outsourced to third parties who sell speciality security services, such as configuration management or firewall rule analysis.

2.3 Solutions

The following means can be used in combination in order to address the challenges above:

- *General standards and practices for operating in the cloud.* Several already exist, for example those provided by ENISA [4], CSA [5] and NIST [11]. The need for provision of consolidated European cloud standards has been highlighted and is currently being addressed within recent EC planning, as considered further in section 4.
- *Privacy by design.* Organisations need to think upfront about the impact and risks they create, and balance innovation with the expectations of individuals. A number of different techniques are available that can be used in combination to achieve this, and to provide data minimization, including anonymisation.

- *Security*. In the context of privacy, security relates to the protection of personal information. Many security controls are available in the cloud context: see for example the Cloud Controls Matrix mapping carried out by the CSA [13].
- *Accountability*. Broadly speaking this term is used in the sense here of corporate data governance related to personal data, including consideration of responsibility and risks upfront. Regulators and individuals expect organizations to act as a responsible steward for the data that is provided to them, and such organizations need to do more to live up to their promises and ensure responsible behavior, which can be achieved via an accountability-based approach.

In the following sections the relationship between these means for addressing privacy in the cloud is analysed. In this paper, the focus is on the way in which these categorisations relate and can be used in combination rather than the mechanisms and controls themselves (including encryption) that are used within them to help achieve privacy in the cloud. For an analysis of the latter, see for example [6,48], and the accountability tools being developed within the EU Cloud Accountability Project [14].

3 To What Extent Is Information Security an Integral Part of Privacy by Design?

First, the relationship between security and privacy is considered, before an analysis is provided about the role of information security in privacy by design.

3.1 The Relationship between Privacy and Security

At the broadest level (and particularly from a European standpoint), *privacy* is a fundamental human right, enshrined in the United Nations Universal Declaration of Human Rights (1948) and subsequently in the European Convention on Human Rights and national constitutions and charters of rights. There are various forms of privacy, ranging from 'the right to be let alone' [15], 'control of information about ourselves' [16], 'the rights and obligations of individuals and organisations with respect to the collection, use, disclosure, and retention of personally identifiable information' [17], focus on the harms that arise from privacy violations [18] and contextual integrity [19].

For organisations, privacy entails the application of laws, policies, standards and processes by which personal information is managed. The fair information practices (FIP) developed in US in 1970s [20] and later adopted and declared as principles by the Organisation for Economic Co-operation and Development (OECD) and the Council of Europe [21] form the basis for most data protection and privacy laws around the world. This framework can enable sharing of personal information across participating jurisdictions without the need for individual contracts. It imposes requirements on organisations including data collection, subject access rights and data flow restrictions. In Europe, the European Data Protection Directive 95/46/EC (and its supporting country legislation) implements these FIP principles, along with some additional requirements including transborder data flow restrictions. Other privacy-related restrictions may also be imposed (e.g. on cookie usage by the recent EU ePrivacy Directive). Legislation similar to the European Data Protection Directive has been, and continues to be, enacted in many other countries. In contrast, the US does not have a comprehensive regime of data protection but instead has a variety of sector-based or state level legislation and places few if any restrictions on transborder data flow. For further details, see [5].

Security mechanisms protect data by maintaining its confidentiality, integrity and availability; associated functionalities may also be provided, notably: authentication, access controls, data retention, storage, backup, incident response and recovery. Confidentiality is sometimes confused with privacy, but is "The property that information is not made available or disclosed to unauthorized individuals, entities or processes" [22].

Privacy differs from security, in that it relates to handling mechanisms for personal information, although security is one element of that. For example, the FIP can be broadly described as follows, where security is one of the principles:

1. *Data collection limitation:* data should be collected legally with the consent of the data subject where appropriate and should be limited to the data that is needed.
2. *Data quality:* data should be relevant and kept accurate.
3. *Purpose specification:* the purpose should be stated at time of data collection.
4. *Use limitation:* personal data should not be used for other purposes without the consent of the individual.
5. *Security:* personal data should be protected by a reasonable degree of security (i.e. safeguards against such risks as loss or unauthorised access, destruction, use, modification or disclosure of data).
6. *Openness:* individuals should be able to find out what personal data is held and how it is used by an organisation.
7. *Individual participation:* an individual should be able to obtain details of all information about them held by a data controller and challenge it if incorrect.
8. *Accountability:* the data controller should be accountable for complying with these principles.

Legal requirements differ depending upon whether the organisation is a data controller and/or a data processor in a given situation:

- A *data controller* (DC) is an entity (which could be a person, public authority, agency or other body) which alone, jointly or in common with others determines the purposes for which and the manner in which any item of personal information is processed, and this is legally responsible for ensuring compliance requirements are met.
- A *data processor* (DP) is an entity which processes personal information on behalf and upon instructions of the data controller. Contractual agreements may add additional responsibilities or constraints with respect to privacy, although data protection laws stipulate that the organisation that is transferring personal information to a third party for processing remains responsible for the personal information.

The DC must implement appropriate technical and organizational measures to protect personal data against accidental or unlawful destruction or accidental loss, alteration, unauthorized disclosure or access, in particular where the processing involves the transmission of data over a network, and against all other unlawful forms of processing. Such measures need to ensure a level of security appropriate to the risks represented by the processing and the nature of the data to be protected. If processing is carried out on behalf of a DC, the DC must choose a DP that provides sufficient guarantees in respect of the technical security measures and organizational measures governing the processing to be carried out, and must ensure compliance with those measures. A written contract or legal act is needed to bind the DP to the DC and should stipulate that the DP will act only on instructions from the DC [1].

Clarification of DC and DP responsibilities is key in cases where personal information is collected and used within cloud scenarios. The provider of cloud services can be a DP and/or a sole or joint DC. For analysis of how this is likely to change with respect to the forthcoming EU Regulation, see [23,24]. The proposed EU Regulation [2] increases the responsibility and accountability of DCs and DPs. The DC should implement appropriate procedures to ensure that the data processing carried out by the Cloud Service Provider (CSP) complies with the Regulation - but it is difficult for a business customer - especially a Small/Medium Enterprise (SME) - to influence the structure of cloud services, particularly for IaaS services. There can be multiple DPs in some scenarios (see [25] for example, which defines an accountability framework for mobile environments). Contracts and SLAs define respective responsibilities, but some of the responsibility cannot be transferred to the CSP, both in terms of security responsibilities and legal responsibilities.

Cloud providers may be constrained by the levels of security they can offer for different types of cloud. It may be difficult for a service provider to determine if the level offered is appropriate if it does not know what type of data may be stored in the cloud by the customer. Furthermore, security levels need to be enhanced to win business in certain industry sectors (e.g. financial services and health).

3.2 The Role of Security in Privacy by Design

Privacy by Design refers to the philosophy and approach of embedding privacy into design specifications [26-28]. It applies to products, services and business processes. The main elements are:

1. Recognition that privacy concerns must be addressed
2. Application of basic principles expressing universal spheres of privacy protection
3. Early mitigation of privacy concerns when developing information technologies and systems, across the entire information life cycle
4. Need for qualified privacy input; and
5. Adoption and integration of privacy-enhancing technologies (PETs).

In essence, companies should build in privacy protections at every stage in developing products, and these should include reasonable security for consumer data, limited collection and retention of that data, as well as reasonable procedures to promote data accuracy. Various companies have produced detailed privacy design guidelines [29] and methodologies about how to integrate privacy considerations and engineering into the development process [30]. The process can include building privacy into technical solutions by including privacy-enhancing features or through privacy solutions that manage the data from the code level up, as described further below.

'Privacy by policy' is the standard current means of protecting privacy rights through laws and organisational privacy policies, which must be enforced. Privacy by policy mechanisms focus on provision of notice, choice, security safeguards, access and accountability (via audits and privacy policy management technology). Often, mechanisms are required to obtain and record consent. The 'privacy by policy' approach is central to the current legislative approach, although there is another approach to privacy protection, which is 'privacy by architecture' [31], which relies on technology to provide anonymity. The latter is often viewed as too expensive or restrictive. Although in privacy by policy the elements can more easily be broken down, it is possible (and preferable) to enhance that approach to cover a hybrid approach with privacy by architecture.

Privacy settings are an important aspect of online privacy. *Privacy by default* is a software design concept that is presently being promoted by a number of data protection authorities, including the EC. Privacy by default would prohibit the collection, display, or sharing of any personal data without explicit consent from the customer. More detailed definitions often include the requirement that privacy settings that limit the sharing of personal data be turned on by default. Notable examples of this approach include MS Internet Explorer 6 and above, in which privacy settings blocked cookies, and also Internet Explorer 10, in which the default setting is that the Do Not Track (DNT) flag is set to "on". Privacy by default is not really a security issue, although there are good settings from a privacy point of view related to security aspects, e.g. restricting access to personal data via a 'deny by default' access control policy.

Privacy enhancing technologies (PETs) are solutions whose specific purpose is to help consumers and companies to protect their privacy [32]. These include those technologies that permit developers, solution providers and service companies to add privacy enhancements to their solutions. For example, PETs include anonymisers and pseudonymisers (e.g. anonymous Web and email access), history-clearing tools, pop-up blockers, anti spam, anti spyware, cookie managers, secure file deletion and software for firewalls.

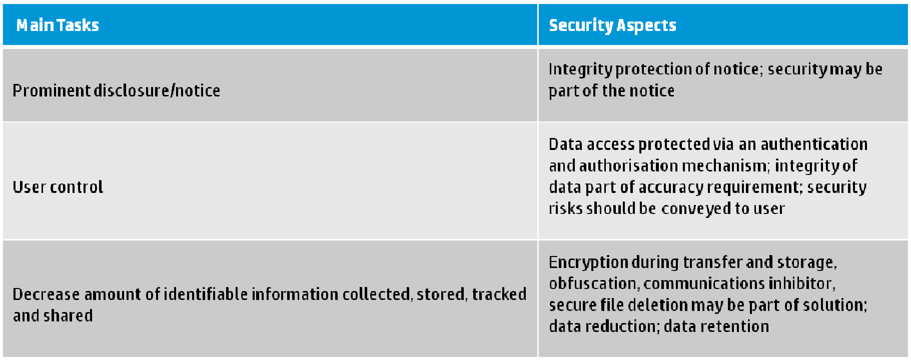
Privacy aware technologies (PATs) are standard non-privacy related solutions that include features that enable users to protect their privacy, e.g. passwords, file access security, communication inhibitor, encryption. The OECD privacy principles may be used to guide good system design, for example by addressing the following issues during the design process:

- *Collection limitation*: Investigate what data systems are collecting automatically; Determine what data you really need and collect only that
- *Data quality*: Keep data up to date; Use data for relevant purposes
- *Purpose specification*: Work out why you are collecting data and explain it in your policy

- *Use limitation*: Keep track of purpose for which data was collected; obtain consent for other uses; Mechanisms may be needed for obtaining and recording consent
- *Security safeguards*: If you collect it, you need to secure it
- *Openness*: Users need to be informed of all data collection, including implicit collection using cookies, behavioral tracking, etc.
- *Individual participation*: Work out how you are going to handle access, correction and purging of data
- *Accountability*: Be proactive about developing policies, procedures, and software to comply with these principles

Security aspects can be involved within privacy by design tasks, as illustrated in Table 2.

Table 2. Privacy by Design



Main Tasks	Security Aspects
Prominent disclosure/notice	Integrity protection of notice; security may be part of the notice
User control	Data access protected via an authentication and authorisation mechanism; integrity of data part of accuracy requirement; security risks should be conveyed to user
Decrease amount of identifiable information collected, stored, tracked and shared	Encryption during transfer and storage, obfuscation, communications inhibitor, secure file deletion may be part of solution; data reduction; data retention

There can be security aspects within several stages of the data lifecycle (Collection-Storage-Processing and Transfer-Archival-Destruction), including: secure storage, transfer, retention and disposal (inc. physical security, encryption); disclosure to authorised, authenticated parties; data protection plan of third parties, inc. confidentiality and security requirements in vendor management; data loss prevention; risk assessment; compliance and auditing; securing backup; disaster recovery. Security aspects may also be involved within different phases of the privacy design lifecycle:

- *Initiation*: setting high level recommendations
- *Planning*: Describing privacy requirements in detail
- *Execution*: Identifying problems with proposed privacy solutions; Considering alternatives; Documentation
- *Closure*: Audit; Change control; Considering privacy during backup; Fault repair; Business continuity; Disaster recovery, etc.
- *Decommission*: Ensuring secure deletion and disposal of personal data

In conclusion, from the above analysis it can be seen that information security is an integral part of privacy by design, although the latter includes a number of other considerations and techniques.

4 What Is the Relationship of Accountability to Privacy by Design?

First the notion of accountability is explained, and then its relationship to design for privacy and to external standards is considered. While regulatory frameworks involving accountability provide a foundation for data protection in the cloud, none are specifically designed with cloud computing in mind.

4.1 The Concept of Accountability

The concept of accountability is used in various different communities in a slightly different sense, and there is no commonly agreed definition. In particular, in data protection regulation since the 1980s, accountability has been used in the sense that the DC is responsible for complying with particular data protection legislation and, in most cases, is required to establish systems and processes which aim at ensuring such compliance. The Article 29 Data Protection Working Party [33], the European Data Protection Supervisor [23], as well as the data protection and privacy regulators at the 31st International Conference of Data Protection and Privacy Commissioners [34] have all recently paid special attention to the principle of accountability. In IT governance, accountability is used in the sense that the information security management system of an organisation is meant to generate assurance, transparency and responsibility in support of control and trust. For corporate governance, accountability is viewed as an organisational privacy management program. There are also other types of usage coming from social science and computer science. For example, the privacy-oriented definition of accountability given in ISO standard 29100 [35] expresses accountability in terms of the practices associated with it in organisations: *"Accountability: document policies, procedures and practices, assign the duty to implement privacy policies to specified individuals in the organisation, provide suitable training, inform about privacy breaches, give access to effective sanctions and procedures for compensations in case of privacy breaches."* For further discussion of the concept, see for example [36,37]. Overall, accountability can be thought of in terms of defining governance to comply in a responsible manner with internal and external criteria, ensuring implementation of appropriate actions, explaining and justifying those actions and remedying any failure to act properly [38].

The scope of accountability can cover a range of diverse aspects, including politically sensitive areas such as intrusive Government surveillance and the responsibilities of organizations to contribute fairly to their local societies via appropriate payment of taxes. Some of these, like Sarbanes-Oxley requirements, do not have a strong connection to privacy. The issue of accountability in relation to intrusive governmental surveillance for national security purposes is more general than cloud computing, although two secret mass surveillance programmes recently revealed (ie. PRISM and Tempora) have a connection to cloud computing in that information collected by certain US-based cloud companies about EU citizens was made available to the US and UK security services. Accountability controls that centre on enforcement of private contracts and domestic data protection legislation would not provide effective protection against such activities; instead, the relevant sphere of governance is such cases seems to be in application of the principle of legality, proportionality and judicial and parliamentary accountability, potentially combined with technical measures to help make scrutiny of social media accountable [39]. Here we focus on the reduced scope of corporate accountability for handling personal data in the cloud, and use accountability in the following sense: *Accountability for an organisation consists of accepting responsibility for the stewardship of personal and/or confidential data with which it is entrusted in a cloud environment, for processing, storing, sharing, deleting and otherwise using the data according to contractual and legal requirements from the time it is collected until when the data is destroyed (including onward transfer to and from third parties). It involves committing to legal and ethical obligations, policies, procedures and mechanisms, explaining and demonstrating ethical implementation to internal and external stakeholders and remedying any failure to act properly* [37, 38].

Accountability can be provided within an organisation, by means of the organisation identifying risks, having appropriate policies that mitigate risks, mechanisms for enforcement internally and for monitoring that these are effective within the enterprise, and for internal and external validation of this. In addition, provision of transparency and redress to customers and end users is also very important. Technology can be used for example to strengthen the enforcement and monitoring of policies, to support design for privacy to help provide assurance and transparency and enforce privacy obligations along the service provision chain.

These elements of risk assessment, transparency and redress are captured within the core elements of implementing an accountability project within an organisation specified within the Galway/Paris projects [43]. In terms of risk assessment, this involves on-going risk assessment and mitigation relating to new products or processes, as well as regular risk assessment and validation of the accountability programme itself. This analysis influenced the similar guidance for a privacy management programme provided by the Privacy Commissioners of Canada, Alberta and British Columbia [44], which again included privacy risk assessment mechanisms as well as on-going assessment and revision of the programme controls.

Risk assessment (a core security process) is particularly important for accountability because it is a central part of the process used to determine and demonstrate that the policies (whether reflected in corporate privacy and security policies or in contractual obligations) that are signed up to and implemented by the organisation (that is taking an accountability-based approach) are appropriate to the context. The type of procedures and mechanisms vary according to the risks represented by the processing and the nature of the data [4,40,41]. In the cloud context, as considered in section 2.2, the risks also depend upon the cloud service and deployment models used [4,6]. Further research to provide risk assessment mechanisms in relation to cloud service provision is being carried out within the Cloud Accountability Project [14].

Existing organisational risk assessment processes need to be enhanced to meet the requirements above, or else supplemented with separate privacy-specific risk assessment. Privacy impact assessments are already being rolled out as part of a process to encourage privacy by design [42]: in November 2007 the UK Information Commissioners Office (ICO) (an organisation responsible for regulating and enforcing access to and use of personal information), launched a Privacy Impact Assessment (PIA) [42] process (incorporating privacy by design) to help organisations assess the impact of their operations on personal privacy. This process assesses the privacy requirements of new and existing systems; it is primarily intended for use in public sector risk management, but is increasingly seen to be of value to private sector businesses that process personal data. Similar methodologies exist and can have legal status in Australia, Canada and the USA [42]. The methodology aims to combat the slow take-up to design in privacy protections from first principles at the enterprise level. Usage is increasingly being encouraged and even mandated in certain circumstances by regulators [42]. Data impact assessment may also become an obligation for some high risk contexts within the forthcoming EU regulation [cf. Article 33: 2].

4.2 The Relationship between Accountability and Design for Privacy

A major driver for an accountability-based approach is to provide an incentive for organizations to 'do the right thing', in terms of decreasing regulatory complexity, easing transborder data flow restrictions while avoiding increased privacy harm, encouraging best practice and using strong punishment as a deterrent. For example, in response to the seemingly insufficient reflection of EU data protection principles and obligations in concrete measures and practices used by organisations, the Article 29 Working Party advocated in its Opinion on the principle of accountability [33] that such a general principle could help move data protection 'from theory to practice', as well as provide a means for assisting data protection authorities in their supervision and assessment tasks. Organisations are allowed increased control over aspects of compliance (i.e. which tools and mechanisms to use in order to achieve compliance), but at the expense of having to demonstrate on an ongoing basis that these mechanisms are appropriate for their business context, and operationally work as expected. For example, the Article 29 Working Party extended a similar notion of accountability to that contained within the OECD guidelines (as considered above in Section 3) with a requirement for DCs to be able to demonstrate compliance to supervisory authorities upon request [33]. Although some specific measures would have to be implemented for most processing operations, for reasons of scalability and flexibility, the suitability of measures needs to be determined on a case-by-case basis, with particular reference to the type of data and to the risks involved.

It is important to state that accountability should not be seen as an alternative to privacy [45]. Instead, as shown in Figure 3, accountability and privacy by design are complementary, in that the latter provides mechanisms and controls that allow implementation of principles and standards, whereas accountability makes organizations responsible for providing an appropriate implementation for their business context, and addresses what happens in case of failure (i.e. if the account is not provided, is not adequate, if the organisation's obligations are not met e.g. there is a data breach, etc.). Privacy by Design may to some extent incorporate corporate accountability mechanisms [46], in that a privacy management program can be seen as bridging between accountability and privacy by design.

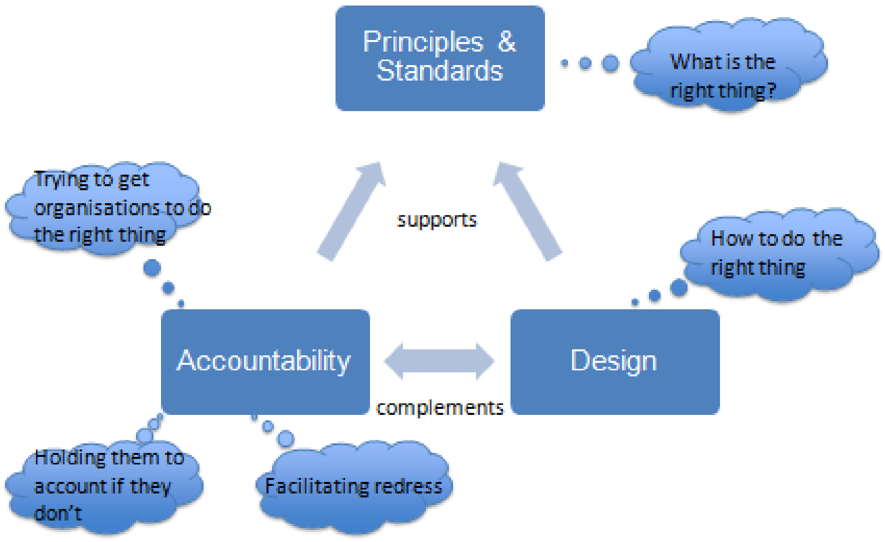

Principles & Standards
What is the right thing?
supports
Trying to get organisations to do the right thing
How to do the right thing
Accountability
complements
Design
Holding them to account if they don't
Facilitating redress

Fig. 3. Accountability Context

At the top of Figure 3, standards are external criteria against which the organization needs to measure themselves and demonstrate compliance against where relevant to the business context. Some of these are technical best practice standard, as considered further within the following section. Others are social principles that are reflected in legislation, such as the OECD privacy principles discussed in section 3. Accountability should underpin the principles and standards set by society rather than undermining them; such principles should be subject to change via a democratic process. Additional constraints may come from other legal and stakeholder requirements. Together, these set societal, regulatory and contractual obligations that organizations need to meet, and for which they are accountable to certain other parties, such as business partners and regulators.

In Section 3 it has been considered how these principles may be reflected in privacy by design, via PATs and PETs. There are a range of different privacy and security controls that could be used, including encryption, that suit different contexts (see for example [47,48]). Implementation and configuration choices also need to be decided upon.

Accountability is concerned with governance mechanisms related to punishment, remediation, transparency, providing a trustworthy account, holding to account, data breach detection and notification, etc. Correspondingly, accountability mechanisms and tools do not focus on providing privacy and security tools *per se*, but rather on formation of appropriate organizational policies, detection of violation of obligations, notification, remediation in complex environments, increased transparency without compromising privacy, provision of a trustworthy account, improved verification and assurance, etc. Accountability should involve checking and proving that data stewardship is in place along the service provision chain, which involves showing that appropriate privacy and security 'design' controls are being used. Hence, accountability should not be a replacement for certain other procedures, including privacy controls, but should be used to complement these [42] and an accountability-based approach should not be used to justify the abandonment of privacy rights and principles.

Although organisations can select from accountability mechanisms tools in order to meet their context, the choice of such tools needs to be justified to external parties. It would be a mistake to have this reliant upon self-certification or weak certification processes. As Bennett points out [p45: 45], due to resource issues regulators will need to rely upon surrogates, including private sector agents, to be agents of accountability, and it is important within this process that they are able to have a strong influence over the acceptability of different third party accountability mechanisms. This can be achieved via independent testing of practices, provision of evidence that is taken into account, including auditing against the ISO 27001 series and associated cloud security standards.

4.3 The Role of Standards

From the definition of accountability given above, it can be seen that standards and best practice are a reference point when organizational policies are formed as part of an accountability-based approach, and the latter (and the internal organizational practices) need to be justified against those. This extends also across to justification of the selection and usage of appropriate cloud service providers with appropriate controls in place.

Examples of such standards in the cloud space include organisational security guidance for the cloud [4,5], guidance to UK organisations on the use of cloud computing [49] and higher-level enterprise risk management guidelines for executives to enable them to identify, monitor, and mitigate or accept the risks that come with using cloud computing [50].

In addition, competing architectural standards are being developed, with big cloud vendors pushing their own mutually incompatible *de facto* standards. Limitations include: differences between common hypervisors; gaps in standard APIs for management functions; lack of commonly agreed data formats; issues with machine-to-machine interoperability of web services. The lack of standards makes it difficult to establish security frameworks for heterogeneous environments and forces people for the moment to rely on common security best practice. As there is no standardised communication between and within cloud providers and no standardized data export format, it is difficult to migrate from one cloud provider to another or bring back data and process it in-house.

The EC is driving a number of initiatives around harmonization across the member states for cloud. These reflect concerns about trust in cloud computing and include standards and mechanisms for interoperability and data portability, security, cloud and compliance. New requirements are coming through for cloud providers, e.g. with regard to breach notification and cyber incident notification, penalties are increasing, and business environments are getting more complex. The draft data protection legislation [2] has already been discussed above. In addition, Neelie Kroes (the Vice-President of the European Commission responsible for the Digital Agenda) has launched a *European Cloud Computing Strategy* aiming at

- more clarity and knowledge about the applicable legal framework (includes development of model contracts & BCRs)
- making it easier to verify compliance with the legal framework (e.g. through standards and certification)
- developing it further (e.g. through a European Cloud Partnership to drive innovation and growth from the public sector).

A Key Action within this cloud strategy is to cut through the jungle of standards, in particular building upon NIST standardisation, ongoing work within the ETSI cloud group and ENISA/CSA voluntary cloud certification schemes.

In February 2013 the European Commission published a *cybersecurity strategy* alongside a *draft directive on network and information security*. Once implemented, cloud service providers and many others will all be covered by a range of data security obligations including adopting risk management practices and reporting major security incidents. The EC expects the directive to be adopted in 2015. As part of this strategy, the EC will set up in June this year a platform on network and information security bringing together relevant public and private stakeholders, to identify good cybersecurity practices across the value chain and create favourable market conditions for development and adoption of secure IT solutions.

In summary, the relationship between accountability and privacy by design is both complementary and at times closely interlinked, such as when privacy impact assessment is deployed as a key part of organizational privacy management programmes.

5 Conclusions

There is a complex and interesting relationship between privacy, security and accountability. For example, privacy relates to personal information only, whereas security and confidentiality can relate to all information, but consideration of the security of personal information is an essential aspect within the process of privacy by design. Accountability has a broader scope than privacy but is often used specifically in the context of privacy governance. In this sense, accountability can be viewed as being complementary to privacy by design, in that it can help support the implementation of privacy principles within organisations.

Cloud computing creates new dynamics to such analysis in that there is an additional role of cloud provider, and indeed there could be several such parties. Top barriers in providing cloud computing services include lack of customer trust and regulatory complexity in global business environments: cloud consumers want data processors to respect their obligations and policies and be compliant (especially as they may be legally liable). A combination of privacy by design and accountability can help provide solutions to such issues.

Further work on development of such solutions is being carried out within the Cloud Accountability project [14], which is an integrated research project under the EU Framework 7 programme addressing data governance problems and developing accountability solutions for the cloud.

Acknowledgements. The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement n° 317550. Thanks also to Simon Shiu who provided me with a version of Figure 1 and some associated insights, as well as colleagues within the A4Cloud project for feedback on related materials.

References

1. European Commission (EC): Directive 95/46/EC of the European Parliament and of the Council of 24 October 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data (1995)
2. European Commission: Proposal for a Directive of the European Parliament and of the Council on the protection of individuals with regard to the processing of personal data by competent authorities for the purposes of prevention, investigation, detection or prosecution of criminal offences or the execution of criminal penalties, and the free movement of such data (January 2012)
3. Mell, P., Grance, T.: The NIST Definition of Cloud Computing. NIST Special Publication 800-145 (September 2011)
4. Catteddu, D., Hogben, G. (eds.): Cloud Computing: Benefits, Risks and Recommendations for Information Security. ENISA Report (November 2009)
5. Cloud Security Alliance (CSA): Security Guidance for Critical Area of Cloud Computing V3.0 (2011)
6. Pearson, S.: Privacy, Security and Trust in Cloud Computing. In: Pearson, S., Yee, G. (eds.) Privacy and Security for Cloud Computing, Computer Communications and Networks. Springer (2012)
7. European DG of Justice: Article 29 Working Party, Opinion 05/12 on Cloud Computing (2012)
8. Baldwin, A., Pym, D., Shiu, S.: Enterprise Information Risk Management: Dealing with Cloud Computing. In: Pearson, S., Yee, G. (eds.) Privacy and Security for Cloud Computing, Computer Communications and Networks. Springer (2012)
9. CSA: The Notorious Nine Cloud Computing Top Threats in 2013. Top Threats Working Group (2013)
10. Ko, R.K.L., Lee, S.S.G., Rajan, V.: Understanding Cloud Failures. IEEE Spectrum 49(12), 84 (2012)
11. Jansen, W., Grance, T.: Guidelines on Security and Privacy in Public Cloud Computing, Special Publication 800-144, NIST (December 2011)
12. Liu, F., et al.: NIST Cloud Computing Reference Architecture. NIST Special Publication 500-292 (September 2011)
13. CSA: Cloud Controls Matrix, v1.4, `https://cloudsecurityalliance.org/research/ccm/`
14. Cloud Accountability Project (A4Cloud), `http://www.a4cloud.eu`
15. Warren, S., Brandeis, L.: The Right to Privacy. 4 Harvard Law Review 193 (1890)

16. Westin, A.: Privacy and Freedom. Atheneum, New York (1967)
17. American Institute of Certified Public Accountants (AICPA) and CICA: Generally Accepted Privacy Principles (August 2009)
18. Solove, D.J.: A Taxonomy of Privacy. University of Pennyslavania Law Review 154(3), 477 (2006)
19. Nissenbaum, H.: Privacy as Contextual Integrity. Washington Law Review, 101–139 (2004)
20. Privacy Protection Study Commission: Personal Privacy in an Information Society, United Statues Privacy Protection Study Commission Fair Information Practices (1977)
21. Organisation for Economic Co-operation and Development (OECD): Guidelines for the Protection of Personal Data and Transborder Data Flows (1980)
22. ISO/IEC 27001: Information technology – Security techniques – Information security management systems – Requirements (2005)
23. EDPS: Opinion of the European Data Protection Supervisor on the Commission's Communication on "Unleashing the potential of Cloud Computing in Europe" (2012)
24. EDPS: Responsibility in the Cloud should not be up in the air". Article EDPS/12/15 (2012), `http://europa.eu/rapid/press-release_EDPS-12-15_en.htm`
25. GSMA Mobile and Privacy: Accountability Framework for the implementation of the GSMA Privacy Design Guidelines for Mobile App Development (February 2012)
26. Cavoukian, A.: Privacy by Design: Origins, Meaning, and Prospects for Assuring Privacy and Trust in the Information Era. In: Yee, G. (ed.) Privacy Protection Measures andTechnologies in Business Organisations: Aspects and Standards, pp. 170–208. IGI Global (2012)
27. UK Information Commissioners Office (ICO): Privacy by Design. Report (2008)
28. Federal Trade Commission (FTC): Protecting Consumer Privacy in an Age of Rapid Change: Recommendations for Business and PolicyMakers. FTC Report (March 2012)
29. Microsoft Corporation: Privacy Guidelines for Developing Software Products and Services, Version 2.1a (2007)
30. Cannon, J.C.: Privacy: What Developers and IT Professionals Should Know. Addison Wesley (2004)
31. Spiekermann, S., Cranor, L.F.: Engineering Privacy. IEEE Transactions on Software Engineering 35(1), 67–82 (2009)
32. Shen, Y., Pearson, S.: Privacy Enhancing Technologies: A Review. HP Labs External Technical Report, HPL-2011-113 (June 2011)
33. European DG of Justice: Article 29 Working Party. Opinion 3/2010 on the principle of accountability (WP 173) (July 2010)
34. ICDPP: 31st International Conference of Data Protection and Privacy 'Data protection authorities from over 50 countries approve the "Madrid Resolution" on international privacy standards' (2009), `http://www.gov.im/lib/docs/odps//madridresolutionpressreleasenov0.pdf`
35. ISO/IEC 29100: Information technology – Security techniques – Privacy framework. Technical report, ISO JTC 1/SC 27
36. Papanikolaou, N., Pearson, S.: A Cross-Disciplinary Review of the Concept of Accountability. In: Proceedings of the DIMACS/BIC/A4Cloud/CSA International Workshop on Trustworthiness, Accountability and Forensics in the Cloud (TAFC) (May 2013)
37. Pearson, S., Charlesworth, A.: Accountability as a Way Forward for Privacy Protection in the Cloud. In: Jaatun, M.G., Zhao, G., Rong, C. (eds.) Cloud Computing. LNCS, vol. 5931, pp. 131–144. Springer, Heidelberg (2009)

38. Catteddu, D., et al.: Towards a Model of Accountability for Cloud Computing Services. In: Proceedings of the DIMACS/BIC/A4Cloud/CSA International Workshop on Trustworthiness, Accountability and Forensics in the Cloud (TAFC) (May 2013)
39. Omand, D., Bartlett, J., Miller, C.: DEMOS Report (2012), `http://www.demos.co.uk/files/_Intelligence_-_web.pdf?1335197327`
40. CNIL: Methodology for Privacy Risk Management (2012), `http://www.cnil.fr/fileadmin/documents/en/CNIL-ManagingPrivacyRisks-Methodology.pdf`
41. Castelluccia, C., Druschel, P., Hübner, S., et al.: Privacy, Accountability and Trust - Challenges and Opportunities. ENISA (2011)
42. Tancock, D., Pearson, S., Charlesworth, A.: Analysis of Privacy Impact Assessments within Major Jurisdictions. In: Proc. PST 2010, Ottawa, Canada. IEEE (August 2010)
43. Center for Information Policy Leadership (CIPL): Demonstrating and Measuring Accountability: A Discussion Document. Accountability Phase II –The Paris Project (2010)
44. Office of the Information and Privacy Commissioner of Alberta, Office of the Privacy Commissioner of Canada, Office of the Information and Privacy Commissioner for British Colombia: Getting Accountability Right with a Privacy Management Program (April 2012)
45. Bennett, C.J.: The Accountability Approach to Privacy and Data Protection: Assumptions and Caveats. In: Guagnin, D., et al. (eds.) Managing Privacy through Accountability, pp. 33–48. MacMillan (2012)
46. Cavoukian, A., Taylor, S., Abrams, M.: Privacy by Design: Essential for Organisational Accountability and Strong Business Practices. Identity in the Information Society 3(2), 405–413 (2010)
47. Camenisch, J., Fischer-Hubner, S., Rannenberg, K. (eds.): Privacy and Identity Management for Life. Springer (2011)
48. Mowbray, M., Pearson, S.: Protecting Personal Information in Cloud Computing. In: Meersman, R., et al. (eds.) OTM 2012, Part II. LNCS, vol. 7566, pp. 475–491. Springer, Heidelberg (2012)
49. Information Commissioner's Office (ICO): Guidance on the Use of Cloud Computing (2012)
50. Horwath, C.: Enterprise Risk Management for Cloud Computing, COSO (June 2012)

Machine-Readable Privacy Certificates for Services

Marco Anisetti[1], Claudio A. Ardagna[1], Michele Bezzi[2], Ernesto Damiani[1], and Antonino Sabetta[2]

[1] Università degli Studi di Milano, Dipartimento di Informatica, Italy
[2] SAP Product Security Research, Sophia-Antipolis, France

Abstract. Privacy-aware processing of personal data on the web of services requires managing a number of issues arising both from the technical and the legal domain. Several approaches have been proposed to matching privacy requirements (on the clients side) and privacy guarantees (on the service provider side). Still, the assurance of effective data protection (when possible) relies on substantial human effort and exposes organizations to significant (non-)compliance risks. In this paper we put forward the idea that a privacy certification scheme producing and managing machine-readable artifacts in the form of privacy certificates can play an important role towards the solution of this problem. Digital privacy certificates represent the reasons why a privacy property holds for a service and describe the privacy measures supporting it. Also, privacy certificates can be used to automatically select services whose certificates match the client policies (privacy requirements).

Our proposal relies on an evolution of the conceptual model developed in the Assert4Soa project and on a certificate format specifically tailored to represent privacy properties. To validate our approach, we present a worked-out instance showing how privacy property *Retention-based unlinkability* can be certified for a banking financial service.

Keywords: privacy, certification, testing.

1 Introduction

The success of the Web as a platform for the provisioning of services and the huge amount of personal information disseminated, collected, and managed through the network comes with important concerns for the privacy of users' data. The growing awareness of users, on one hand, and the increasing regulatory pressure coming from governments, on the other, are imposing new requirements and constraints on businesses that need to handle sensitive data to carry out their services.

As part of the effort of ensuring compliance to the new data protection laws and regulations, several solutions have been proposed defining different privacy-aware languages that help users in defining which of their data can be used, by whom, when, and for which purposes [1,2,3,4,5,6]. These solutions constitute an important step towards increasing the availability of practical data protection technology, leveraging automated processing of privacy policies. To this end, by capturing the data protection requirements as explicit policies, they do address a key facet of the problem. However, their applicability remains limited unless the data protection *guarantees* offered by providers are expressed similarly in a format that can be processed automatically. Initiatives such as *EuroPriSe* [7] and *Trust-E* [8] represent an initial move in the direction of explicitly representing the data protection measures put in place by a service (or by a software product in general), but they have significant limitations. Firstly, they take an *all-or-nothing* approach to compliance (a product is either compliant with their certification schema or not), which does not allow reasoning about privacy assurance based on richer, finer-grained client requirements. Secondly, these schemes rely on a format that, although structured, is essentially based on a natural language description meant for human consumption, but that is not suitable for automated processing. Finally, the evaluation process followed to assess the correctness and adequacy of the privacy protection measures declared in the certificates is not described in detail, which makes the evaluation itself somewhat opaque to the client.

Unfortunately, even the adoption of such first-generation privacy certification schemes is quite an exception; most frequently, service providers release just a text document (typically a web page in their website) describing what data they handle and what protection measures they put in place to protect those data. Such a description, expressed in natural language, requires an understanding of the data protection problem and solution spaces that users cannot be expected to have. This means that it is extremely difficult for users to determine if the protection measures offered by a service adequately cover their needs. Furthermore, the privacy statements associated to services are usually self-declarations by the service provider, which are difficult (if not impossible) for the client to check.

To address these problems, we believe that *i)* the privacy protection statements should be expressed in an explicit, machine-readable format so that the matching of privacy measures (offered by candidate services) with the corresponding client policies (privacy requirements) can be automated; and *ii)* the statements by service providers should be checked and endorsed by a third party that users trust (e.g., a recognized certification entity). Addressing these two aspects would unlock new scenarios that are not possible today: users could discover services based on their data protection guarantees, determine whether a service fulfills a particular privacy policy, and compare similar services based on the extent to which each of them targets a specific data protection goal.

Machine-readable certification of *privacy* statements serves the interests of both clients and providers. As for clients, explicit privacy certificates mean improved *transparency.* Companies (especially SMEs) may have not the adequate resources to assess the "quality" of offered services, especially from the point of view of security an privacy. The opportunity of having security and privacy features described in a structured and machine-readable artifact, as in a digital security and privacy certificate, can support users to make meaningful comparisons, which may also be (partly) automated and supported by tools, similarly to what is described in [9]. On the other hand, privacy certificates are an effective means for service providers to demonstrate compliance with data protection regulations and customer requirements. Although legal compliance with privacy an data protection regulations are mandatory nowadays, organizations often struggle to deal with the large diversity of regulation across geographies and sectors. For example, EU data protection directives often differ from US privacy regulatory framework, not to mention that the EU directive can be differently implemented in the 27 EU member states or sector specific regulations (e.g., HIPAA). Privacy certifications can provide a "stamp of approval" of a trusted, expert third-party attesting the adherence to specific legal and privacy frameworks, ultimately supporting the users to adopt service-based solutions that are provably compliant to national and sector specific regulations.

Over the last three years, the Assert4Soa project [10] has investigated ways of realizing a novel, light-weight approach to security certification of services, according to which finer-grained security properties of applications and services are evaluated by independent third parties and can be expressed in machine-readable artifacts (called Asserts). Among other results, the project defined a conceptual model and a certificate representation, which provide a concrete structure to represent security certification artifacts. Also, a reference architecture has been defined to support the processing of certificates, the discovery of services based on their security properties, and the automated matching of client requirements with services having corresponding certified properties.

In this paper we present an evolution of the Assert4Soa conceptual model and certificate format that is specifically tailored to represent privacy properties.

The remainder of this paper is organized as follows. Section 2 illustrates the motivation of our work and a reference scenario. Section 3 presents our certification model for privacy, and Section 4 illustrates its application using concrete examples of certificates and focusing on privacy property retention-based unlinkability. Section 5 discusses related work and, finally, Section 6 concludes the paper.

2 Motivating Example

We consider a scenario in which a client is searching for a privacy-aware IFX-based[1] financial service that addresses its privacy policies (e.g., any personal information provided to the service stays confidential or is deleted after a given period of time). This scenario involves *i)* a client accessing the IFX-based financial service with a set of privacy requirements, *ii)* a service provider implementing

[1] Interactive Financial eXchange (IFX) Standard (`http://www.ifxforum.org/standards/`), a financial messaging protocol initially defined in the 1997 by financial industry and technology leaders. IFX aims to exchange data electronically to accomplish a variety of transactions between (unknown and) distributed entities. The IFX standard supports many financial and security functionalities, and integrates them in a service-oriented architecture.

an IFX-based service exposed as a SOAP-based service on the Web, and *iii)* a certification authority certifying the privacy properties of web services.

In particular, we consider an IFX-based service implementing a *Deposit and Withdrawal* service that enables clients to make deposits (operation *CreditAdd*), possibly via cheque, and withdrawals (operation *DebitAdd*) in/from their bank account, using a reverse ATM. This service puts strong requirements on security and privacy of the clients, such as, confidentiality of the messaging exchange, integrity of data, authenticity of the involved parties, privacy of data in the cheques, and introduces the need of a security- and privacy-oriented certification scheme.

In this paper, we focus on the certification of privacy property *retention-based unlinkability*, meaning that the service is certified to maintain the client's personal data following the client's requirements specified through a retention-based privacy policy. Let us consider the scenario in which a client deposit a cheque using the reverse ATM connected to our *Deposit and Withdrawal* service. The reverse ATM scans the cheque and allows the client to specify a retention period for the cheque scan when stored at the *Deposit and Withdrawal* service storage.[2] We note that if the retention period is not specified a default one will be used by the service provider. The cheque scan is sent as a parameter of the request to operation *CreditAdd* of the *Deposit and Withdrawal* service, via the SOAP with attachment standard (`http://www.w3.org/TR/soap12-af/`), and the retention period specified by the client is associated as an annotation to it. Additional parameters of operation *CreditAdd* (e.g., amount and identity token) are associated with a default retention period possibly dictated by the legislation of the country in which the reverse ATM resides.

3 Privacy-Assert

We propose the concept of digital privacy certificate for services (p-Assert). p-Asserts are machine-readable, signed statements, bound to services, that certify the privacy properties guaranteed by a service. As in current certification schemes, the assessment of the property is performed by an independent third party (e.g., a certification authority), who issues (and signs) the p-Assert. The certification is based on an evaluation of the service characteristics (e.g., using formal methods or testing), which can be represented in the p-Assert. Differently from existing schemes (e.g., EuroPriSe [7] or Common Criteria [11]), p-Asserts are represented as (signed) XML documents, a format suitable for automated reasoning and processing.

In the following, we present the structure and main features of p-Asserts, which extend the digital security certificates (Assert) introduced in [12]. As in the original Assert, each p-Assert includes three main parts (see Figure 1).

[2] The cheque scan can provide additional information of interest like the cheque transfers and bounces, signatures, dates, which may be sensitive from a privacy point of view.

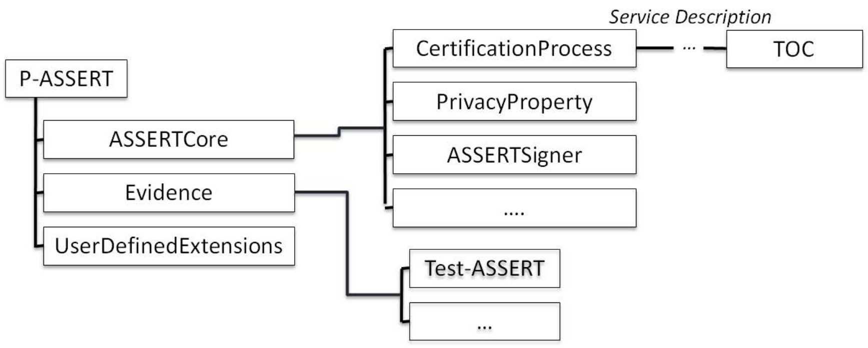


Fig. 1. p-Assert: high-level structure

– A *Core* part that provides information about the certified entity (service description), the specification of the privacy property of the certified entity, and additional contextual information such as the certification authority, signature, and certification process.
– An *Evidence* part that describes the details of the evaluation performed by the certification authority (see Section 3.2) and supports the certification claims, such as a structured description of the test suites executed as part of the evaluation process (see [13] for details).
– An additional part reserved for extensions, that can be used, e.g., to cover domain-specific concerns or to provide additional information on top of the content of the *Core* and/or the *Evidence*.

More in detail, in the core part, the service description contains the Target of Certification (TOC) describing the service being certified, and the Target of Evaluation (TOE) describing the part of the Target of Certification that is evaluated and the rationale for protecting the assets that are identified. More sophisticated models can be present in the *evidence* section of the p-Assert to describe the evaluation performed, e.g., Symbolic Transition System model for the generation of test cases (see Section 3.2).

Note that, traditional security (as Common Criteria) or privacy (as EuroPrise) certification schemes do not make a clear distinction between the system that is being certified and the aspects of the system that are subject to evaluation, limiting the description to the TOE in natural language. However, this distinction becomes more relevant whenever we want to use the certificates in service-based systems, because services can be easily composed of multiple, external services, and it should be clear which part is evaluated. Similarly, to allow for machine-readability, the service description also provides a list of assets, which will be explicitly referred in the different parts of the certificates. These assets replace the natural language description of assets-to-be protected in today certification schemes.

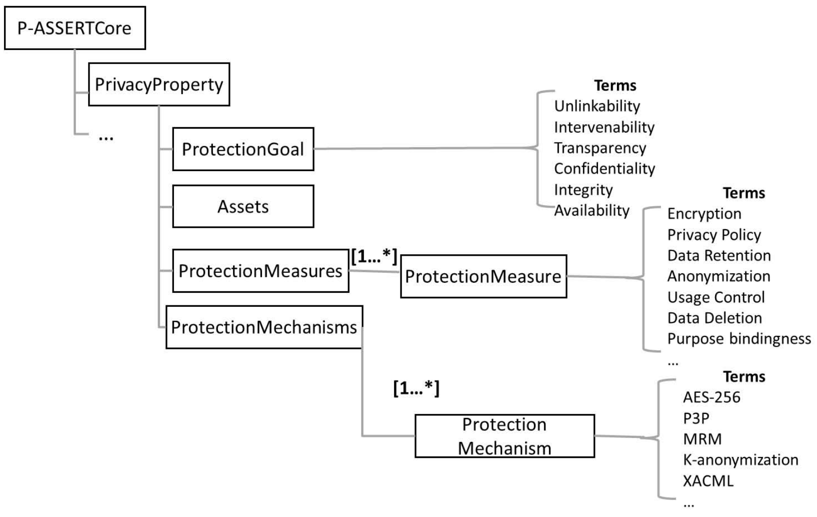


Fig. 2. *PrivacyProperty*: main elements

The privacy property specification element contains a multi-level description of the privacy property at different abstraction levels. We discuss this element in detail in Section 3.1. The evaluation specific portion of the certificate defines the representation of the details and results of the service evaluation process needed to support the certified property, describing, for example, the models used to generate the test cases and the tests performed on the system. We will describe this part in Section 3.2.

The proposed structure of the p-Assert is based on the the Assert model (which targets security properties), the main difference is in the description of the property. In the next sub-section, we will present the privacy property specification element, we refer the reader to [12,13], for a complete analysis of the remaining part of the Assert.

3.1 Privacy Property

Privacy properties need to be specified at different levels of abstraction, from more abstract concepts to fine-grained representations. The advantage is two-fold: first, it allows end-users with different levels of expertise to understand the privacy features of the service, increasing transparency; second, it permits users (being machines or human beings) to search for services that match their privacy requirements at different levels of complexity. Accordingly, we propose different elements, from abstract concepts to the technical implementation mechanisms, to describe the property, as illustrated in Figure 2. A privacy property for a certain asset (*Assets* element to the service description) is described with a three layer structure, as in [14,15], in terms of *Protection Goal*, *Protection Measures* and

Protection Mechanisms. All the other elements are common with the previously introduced ASSERT representation for security certificate (see [13] for details).

Regarding the most abstract layer, Privacy by Design principles [16] constitute a natural starting point to express privacy principles. On the other hand, they mix regulative and engineering criteria, making them unsuitable for describing technical features only. Recently, in analogy with the "classical" data security protection goals of *confidentiality*, *integrity*, and *availability*, three additional *data protection goals* (and corresponding protection measures) have been proposed [14,15].

Transparency means that "the collection and processing operations of data and its use can be planned, reproduced, checked and evaluated with reasonable efforts." It can be supported by measures such a clear privacy policy, breach notification, and so on.

Unlinkability ensures that personal data cannot be linked across domains or used for a different purpose than originally intended [15]. It can be supported by measures like: limited retention period, data erasure, anonymization, data minimization, separation of contexts by different identifiers.

Intervenability is the "ability to intervene" for data subjects, operators and supervisory data protection authorities to apply corrective measures if necessary. For example, it includes the right to rectification and deletion of data for the data subject. It can be supported by measures such as mechanisms for handling data subject's correction requests, *break-the-glass* policies, and the like.

These protection goals provide a high-level description of the privacy properties, but they do not give any information how these goals can be achieved. The P-ASSERT contains a list of *protection measures*, which are linked to a protection goal, indicating the necessary measures to reach the goal. For example, the *unlinkability* protection goal can be supported by *anonymization* and *data retention* measures. Measures are realized by specific protection mechanisms, describing the techniques or procedures used to realize a specific protection measure. For example, *anonymization* protection measure can be implemented by specific *k-anonymity* algorithms, with a set value of k.

More formally, a privacy property p is a pair ($\hat{p}$,A), where $p.\hat{p}$ is an protection goal and $p.A$ is a set of class attributes referring to specific characteristics of the privacy function implemented by the service (i.e., detailed description of protection measures and mechanisms). For instance, property p=(*confidentiality*,{`measure`=encryption,`algo`=DES,`key`=112bit,`ctx`=in transit}) describes a privacy property whose *protection goal* is confidentiality in transit, *protection measure* is encryption, and *protection mechanism* is DES encryption algorithm with key length of 112bits.

In some cases, a partial order can be defined over privacy properties based on attribute values, inducing a hierarchy $\mathcal{H}_P$ of properties as a pair ($\mathcal{P}$,$\preceq_P$), where $\mathcal{P}$ is the set of properties and $\preceq_P$ the partial order. Given two properties p_i and p_j, we write $p_i \preceq_P p_j$, if p_i is weaker than p_j (see [17] for a more detailed discussion on partial ordering of security properties). The hierarchy of privacy properties is fundamental for comparing different services from a privacy point of view, which is one of the most prominent functionality for a privacy-aware SOA infrastructure.

3.2 Evidence Representation in p-Assert

In this section, we describe a test-based certification scheme, that is, a process producing evidence-based proofs that a (white- and/or black-box) test carried out on the software has given a certain result, which in turn shows that a given high-level security property holds for that software [18]. The evidence in p-Assert contains test-based artifacts and details on how these artifacts support privacy property p defined in the core part of p-Assert. More in detail, it is divided into two main sections as follows.

Service model m: A Symbolic Transition System (STS) [19] that specifies service behavior and interactions as a finite state automaton. It is used for automatic generation of test cases. The service model specifies a label *Model* that describes its level of detail and assumes values in: *i) WSDL* when m models the Web Service Definition Language (WSDL) interface only, *ii) WSCL* when m models the client-server conversation in the Web Services Conversation Language (WSCL) document, and *iii) implementation* when m models the implementation of service operations. We note that the service model only describes those operations, called Most Important Operations (MIOs), that are needed to certify privacy property p and to maintain the correctness of the service model. A set of quantitative indexes v^m (e.g., number of states) is defined to calculate a quality measure for the model (the complete set of indexes is provided in [17]). As an example, a WSCL-based model for the *Deposit and Withdrawal* service described in Section 2 is depicted in Figure 3.

Figure 3 shows the WSCL conversation that allow the client to access operations *CreditAdd* and *DebitAdd* of the service. First the client has to log into the system. Operation *Signon* returns the variable *result*='ok' with a *token* in case of successful authentication, *result*='failure' otherwise. After a successful *Signon*, the client can call either operation *CreditAdd* or *DebitAdd*. *CreditAdd* takes as input variables *amount* (the amount of money to be deposited), *scan* (an optional parameter with a scan of the cheque used to transfer money and passed as an attachment to the SOAP message of the request), *token* (an authentication token returned by operation *Signon*), and *rp* (the retention policy attached to variable scan). *DebitAdd* takes as input variables *amount* (the amount of money to be withdrawn) and *token* only. Both operations return the *result* of the execution as output.

Test Evidence e: The testing artifacts proving a property for the certified service. Test Evidence e is composed of the set of test cases executed on the service, their category (i.e., functionality, robustness, penetration) and type (e.g., random input, equivalence partitioning), and a set of test attributes (e.g., the cardinality of the test set). It also specifies a set of test coverage metrics (e.g., branch coverage,

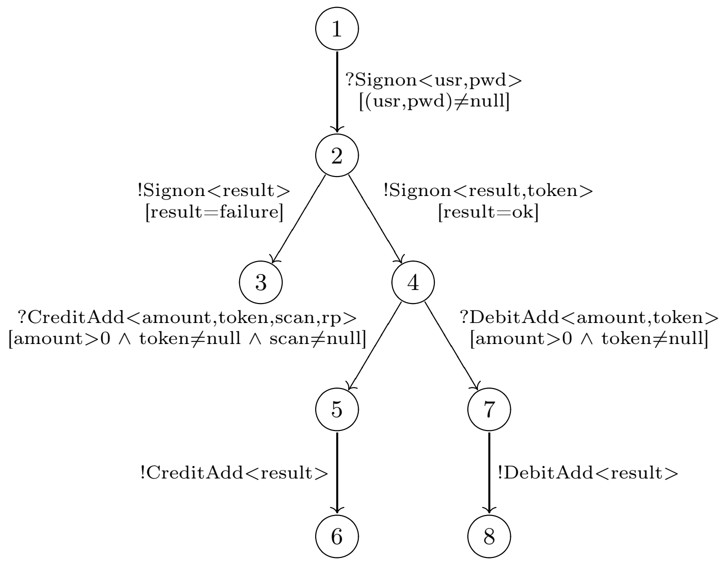


Fig. 3. WSCL-based model for deposit and withdrawal service

path coverage) measuring how much the test set covers the service model, and is complete and exhaustive. More formally $e = \{cat(e), type(e), ta(e), tc(e), tr(e), v^e\}$ where $cat(e)$ is the test category, $type(e)$ is the test type and $ta(e)$ are the related attributes; $tc(e)$ are the test cases while $tr(e)$ are the results of their execution. v^e_j is the set of test coverage metrics. The complete set of metrics is provided in [17] and can be used to calculate an aggregated quality metric.

Test evidence can support a variety of privacy-related properties. For instance a certification authority can award a privacy certificate $C(p,m,e)$ to a service s with privacy property $C.p = ($*Confidentiality*,$\{$`measure`=encryption,`algo`=*DES*, `key`=*112*,`ctx`=*in transit*$\})$, service model $C.m = \{$*WSCL*$,*\}$, and evidence $C.e =$ (*Functionality*,*Input Partitioning.Equivalence Partitioning*,$\{$`card`=130$\},*,*,*)$, where $C.e.$`card` is the cardinality of the test set and $*$ means any value. Certificate C provides the evidence proving that s holds the privacy property with the protection goal of confidentiality at communication (*in transit*) level, using a 112-bit *DES* algorithm. In this example, evidence is produced using a WSCL-based model and a set of 130 test cases with test category *Functionality* and test type *Input Partitioning.Equivalence Partitioning.*

4 Certification of Retention-Based Unlinkability

In the previous section, we briefly presented an example of P-ASSERT certificate for Confidentiality of data in transit supported by functionality test. In this section, we present a complete worked-out example showing how privacy property

Retention-based unlinkability can be certified. We assume that a simplified language is available permitting to specify the *retention period* of data contained in a service request.[3] After the retention period expires, the service provider must delete any reference to the data, assuring users that their data are no longer available for access.

To show the process of certifying a service for property *Retention-based unlinkability*, we consider the *Deposit and Withdrawal* service in Section 2 implementing a privacy retention mechanism similar to the one adopted by Microsoft Exchange Server 2010, which supports the definition and enforcement of retention tags and policies [20], but at filesystem level. We assume that each request performed by a client specifies a *sticky policy* [1] for each data item with the retention period. Sticky policies are data handling policies attached to the personal data they protect and regulate how personal data will be handled at the receiving parties (i.e., data controllers and processors). Users specify these policies to define restrictions on the retention period of their data, thus keeping a level of control on their information also after its release. Clearly, the retention period specified in the request by a user must comply with the requirements for retention defined by the service (see Section 2); if not specified a default retention period applies for the user request.

In the following, we consider the certification of a cheque-based *CreditAdd* only and assume that the retention period for a cheque scan attached to the request in a cheque-based deposit is directly specified by the client using the reverse ATM (see parameters *scan* and *rp* in Figure 3). We note that a retention period can be also specified for parameters *amount* and *token*, though not discussed in our scenario.

Suppose that the service supports a retention mechanism with frequency of control 1 second, minimum retention period 1 day, and maximum retention period 1 year. To this aim, the service implements a mechanism that periodically checks (every 1s) the retention period of each request and deletes all data for which the retention period is expired. In particular, the process implemented by the retention mechanism is composed of the following steps:

1. the client sends a *CreditAdd* request to a service annotated with a retention period *rp* for the cheque scan. The retention period is defined in seconds and automatically transformed in a precise date and time at the service side;
2. the service checks if the retention period complies with its requirements (e.g., minimum retention for cheque scan). If yes, the cheque scan is stored in the filesystem with the retention period; if not, the user request fails;
3. the service periodically controls the cheques' storage and flags all cheque scan for which the retention period is expired (i.e., the date and time in the retention policy are before the current date and time);
4. flagged cheques are deleted.

[3] This assumption is not restrictive since our solution can be easily adapted for working with any privacy policy language supporting retention, including classic P3P [6].

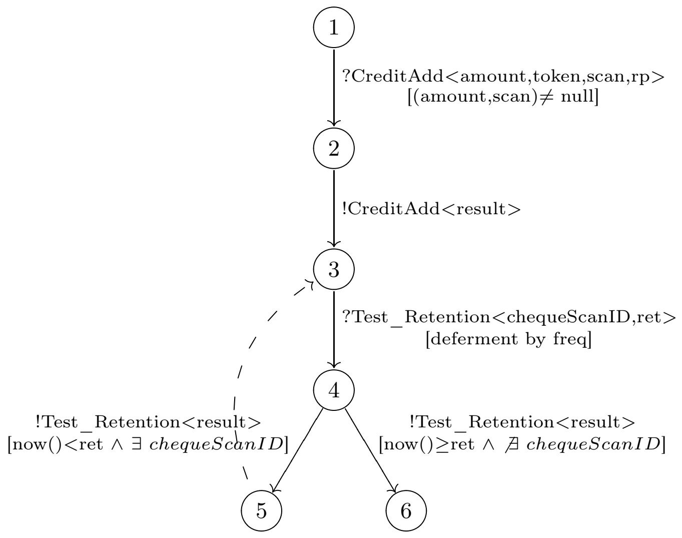

1
?CreditAdd<amount,token,scan,rp>
[(amount,scan)≠ null]
2
!CreditAdd<result>
3
?Test_Retention<chequeScanID,ret>
[deferment by freq]
4
!Test_Retention<result>
[now()<ret ∧ ∃ chequeScanID]
!Test_Retention<result>
[now()≥ret ∧ ∄ chequeScanID]
5
6

Fig. 4. WSCL-based Test model for Deposit and Withdrawal example

As soon as the service provider wants to certify this service for privacy property p=(*Unlinkability*,{`measure`=retention,`frequency`=1s, `min_retention`=1d, `max_retention`=1y}), the certification authority (CA) verifies the correct working of the above described retention mechanism by means of *deferred testing execution*. *Deferred testing execution* is a largely used test execution approach in which the executions of two consecutive test cases are deferred by a specific temporal delay. The delay of the deferment is usually due to the fact that the components that a test case exercises may not be ready for inspection by the time the test runs (e.g., due to instantiation of classes declared with deferred initialization stages). In our approach, execution deferment is used to test properties, like retention, that should become true (or false) after a given period of time. Our testing strategy relies on the model m of the service (see Figure 4), on test category *cat*, and test type *type* to produce the test model used for test case generation [17]. The test model allows deferred test execution via the definition of specific timing conditions on the STS-based service model (see Figure 4). In this specific case, to support the testing activities, the WSDL of the service is extended with a new operation *Test_Retention*(*chequeScanID*,*ret*). This operation, which is used at certification time only and then removed, provides the test code for evaluating the retention mechanism. It takes the name of the file representing the cheque scan (*chequeScanID*) in the service storage and the retention period *ret* (date/time) derived from the retention policy *rp* as input, and returns the result of the testing activity as output.

The *Test_Retention* operation checks the storage for cheque *chequeScanID*. The retention mechanism is correctly working and the *Test_Retention* returns true if: *i)* the cheque is in the storage and the retention period is not expired

Privacy Property: Unlinkability
Class Attributes: *measure*=retention, *frequency*=1s, *min_ retention*=1d, *max_ retention*=1y

$$TC1 = \begin{cases} I_1: & CreditAdd(amount, token, scan, rp) \in valid\ partitions \\ EO_1: & result = ok \\ PR_2: & deferment\ by\ freq \\ I_2: & Test\_Retention(scan.chequeScanID, ret)\ with\ now() < ret \\ EO_2: & EO_2: result = True\ \wedge\ scan.chequeScanID\ is\ found \end{cases}$$

$$TC2 = \begin{cases} I_1: & CreditAdd(amount, token, scan, rp) \in valid\ partitions \\ EO_1: & result = ok \\ PR_2: & deferment\ by\ freq \\ I_2: & Test\_Retention(scan.chequeScanID, ret)\ with\ now() \geq ret \\ EO_2: & result = True\ \wedge\ scan.chequeScanID\ is\ not\ found \end{cases}$$

$$TC3 = \begin{cases} I_1: & CreditAdd(amount, token, scan, rp)\ with\ amount, token, rp \in valid\ partitions\ \wedge \\ & \quad \wedge\ scan \in invalid\ partition \\ EO_1: & result = err \\ PR_2: & deferment\ by\ freq \\ I_2: & Test\_Retention(scan.chequeScanID, ret)\ with\ anytime \\ EO_2: & result = True\ \wedge\ scan.chequeScanID\ is\ not\ found \end{cases}$$

Fig. 5. Test cases for retention for *Deposit and Withdrawal* service

(while being less than one year) or is expired by less than 1s (frequency with which the retention is evaluated), *ii)* the cheque is not in the storage and the retention period is expired. It returns false in the other cases.

To test the retention-based privacy property, we first remove operation *Signon* because it is not a MIO and thus does not contribute to the generation of the test cases (see Figure 4). Also, for simplicity, we removed operation *DebitAdd* and focused on the certification of *CreditAdd*. As a consequence, the test model only involves operations *CreditAdd* and *Test_ Retention*. Since the retention test is a deferred testing we add deferring time conditions to the STS-based test model in such a way that the *Test_ Retention* can be executed before the retention time is expired ($now() < ret$) and after the retention time is expired ($now() \geq ret$). The test model in Figure 4 generates multiple calls to operation *Test_ Retention* with different deferment times (i.e., *deferment by freq*) for proving the correctness of the retention mechanism implemented by the service. Our model includes a cycle which iterates until the cheque scan is no longer stored by the service, that is, the retention control mechanism is proved to be correctly implemented for that specific scan. We note that operation *Test_ Retention* is iteratively executed according to the frequency used by the retention mechanism.

Some examples of test cases generated by the test model in Figure 4 are shown in Figure 5. TC1 and TC2 belong to the functionality test category and consider valid test types (all parameters are in their valid partitions). In general, the test model in Figure 4 will generate a set of TC1-like test cases, until the test time (indicated using *now()*) is greater or equal to the retention time *ret* (in the form of date/time) derived from the user's privacy policy. TC3 is a

Privacy Property: Unlinkability
Class Attributes: *measure*=retention, *frequency*=1s, *min_retention*=1d, *max_retention*=1y

$$TC4 = \begin{cases} I: & request = CreditAdd \ \wedge \ (1d < rp < 1y) \\ EO: & result = ok \end{cases}$$

$$TC5 = \begin{cases} I: & request = CreditAdd \ \wedge \ rp < 1d \\ EO: & result = error \end{cases}$$

$$TC6 = \begin{cases} I: & request = CreditAdd \ \wedge \ rp > 1y \\ EO: & result = error \end{cases}$$

Fig. 6. Test cases for verifying retention period boundary values

robustness test case that verifies whether the cheque scan is invalid (e.g. not correctly scanned), while the other parameters of the *CreditAdd* are valid. In this case the result of *CreditAdd* is an error and the cheque scan must be not saved or deleted immediately from the cheque scan storage; the operation may be maintained in the system log with the other parameters of the function call, depending on the legislation of the country in which the reverse ATM resides. The execution of *Test_Retention* with a wrong cheque scan must return a *scan not found* independently by the precise time in which it is executed.

The certification authority can also verify the correct support for minimum and maximum retention periods, using the additional test cases showed in Figure 6.

5 Related Work

Different languages and format for machine-readable privacy policy for web applications and services have been proposed. The XML-based P3P (Platform for Privacy Preferences Project) language and APPEL (A P3P Preference Exchange Language) [6] are used for describing privacy policies and privacy negotiations between a web site and users. The Enterprise Privacy Authorization Language (EPAL) [21] is based on the same concepts of the eXtensible Access Control Markup Language (XACML) [5], but it implements more privacy-specific conditions, such as purpose-based access control. More recently, the PrimeLife Privacy Language (PPL) [22] was defined as an extension of the XACML authorization language. PPL allows to handle complex privacy policies, including specifying secondary usage of the data (e.g., when an external data processor is involved) and privacy obligation handling, relying on XACML for access control conditions.

These privacy policy languages allow for a description and processing of privacy policies, but they do not provide the elements for specifying the protection measures used nor for detailing the evidences supporting their correct implementation. As mentioned in Section 4, p-Assert can rely on policy languages for expressing the privacy conditions, and complement them with a more granular description of the protection measures and evidences.

Another major source of related work for this paper resides in the area of software testing. Similarly to this paper, several works (e.g., [23,24,25]) focused on testing non-functional requirements of software systems. As far as web service testing is concerned, the line of research closest to the one in this paper considers the problem of testing a web service to assess its correct functioning and to automatically generate test cases used in the verification process [26,27]. Heckel and Lohmann [28] propose a solution for testing web services that uses Design by Contract and adds behavioral information to the web service specifications. More recently, Bentakouk et al. [29] propose a solution using STS-based testing and STM solver to check the conformance of the composite service implementation with respect to its specifications and/or client requirements. Endo and Simao [30] present a model-based testing process for service-oriented applications. Existing approaches have mainly focused on static or dynamic testing, while they have not focused on certification. More in detail, these approaches test services with the scope of verifying their security mechanisms (i.e., policy enforcement) ex-post. The p-Assert approach elaborates on the approach presented in [17] and concentrates on container-level certification, which implies security policy testing and certification. In [31], a security testing method for stateful Web Services is proposed. It defines specific (i.e., for each security property to be tested) security rules, eventually derived from policy, using Nomad language with the scope of generating test cases. This rule set allows to test different properties such as availability, authorization, and authentication by means of malicious requests based on random parameters, or on SQL and XML injections. The rules are applied to the operation set (obtained from service specifications like WSDL) of the service under test, to generate test requirements (modeled as STSs), which are then synchronized with the specifications to produce the test set. The goal is to perform a black-box testing of web services exploiting a rule-based approach for the generation of an ad hoc test set. The test set is aimed at discovering if the service under test violates or not the security rules. Other approaches focusing on general testing or on authentication and authorization policies, construct abstract test cases directly from models describing policies [32,33,34]. Le Traon et al. [32] proposed test generation techniques to cover security rules modeled with OrBAC. They identified rules from the policy specification and generated abstract test cases to validate some of them via mutation. In [33], the authors developed an approach for random test generation from XACML policies. The policy is analyzed to generate test cases by randomly selecting requests from the set of all possible requests. In [34], the authors proposed a model-driven approach for testing security policies in Java applications. The policy is modeled with a control language such as OrBAC and translated into XACML. In our case we describe an approach for testing privacy-specific policies for web services and how is it possible to generate a certification scheme for them.

6 Conclusions

We proposed a representation for digital privacy certificates (p-Assert), which describes the outcome of a privacy certification process for web services in a machine-readable format. These structured and machine-readable certificates enable the service consumer to: *i)* know the details about the privacy features of the service (*transparency*), *ii)* access detailed information on the evidence supporting the claim of the certificate (*assurance*) *iii)* automatically reason about privacy properties (e.g., allowing service discovery based on privacy requirements). Our proposal extends the security certification framework developed in the context of the European project Assert4Soa. Following the same approach, we described a digital privacy certificated supported by a model-based testing process, which allows to automatically produce evidence that a given privacy property holds for the service. The corresponding machine-readable certificate contains the certified property, the model of the service used for the automatic generation of the test cases, and the evidence produced by their execution. We also provided a worked-out example of the application of our scheme to the certification of privacy property *retention-based unlinkability.* In this context, we introduced the concept of *deferred testing* as a testing activity that specifies the time intervals between consecutive test cases. Our example focused on *offline* deferred testing, meaning that the test cases on retention policies are executed in an accredited Lab in the framework of a certification process and the retention periods are randomly generated to maximize the coverage of their domain. We note however that our solution can support *online* deferred testing, verifying the correctness of the retention mechanism implemented by the service on real client requests and client-defined retention periods. We plan to further analyze this post-deployment testing scenario in our future work. Our future work will also evaluate the efficiency of our approach, analyzing the overhead required for automatic test generation at the increasing of policy complexity.

Acknowledgements. This work was partly supported by the EU-funded projects Assert4Soa (contract n. FP7-257351) and Cumulus (contract n. FP7-318580). We thank Kirsten Bock and Marit Hansen for fruitful discussions.

References

1. Ardagna, C., Camenisch, J., Kohlweiss, M., Leenes, R., Neven, G., Priem, B., Samarati, P., Sommer, D., Verdicchio, M.: Exploiting cryptography for privacy-enhanced access control: A result of the prime project. Journal of Computer Security (JCS) 18(1), 123–160 (2010)
2. Ardagna, C., Cremonini, M., De Capitani di Vimercati, S., Samarati, P.: A privacy-aware access control system. Journal of Computer Security (JCS) 16(4), 369–392 (2008)
3. Ardagna, C., De Capitani di Vimercati, S., Foresti, S., Paraboschi, S., Samarati, P.: Minimizing disclosure of private information in credential-based interactions: A graph-based approach. In: Proc. of the 2nd IEEE International Conference on

Information Privacy, Security, Risk and Trust (PASSAT), Minneapolis, Minnesota, USA (August 2010)
4. Ashley, P., Hada, S., Karjoth, G., Schunter, M.: E-P3P privacy policies and privacy authorization. In: Proc. of the ACM workshop on Privacy in the Electronic Society (WPES), Washington, DC, USA (November 2002)
5. Moses, T.: eXtensible Access Control Markup Language (XACML) Version 2.0 (February 2005), `http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf`
6. W3C: Platform for privacy preferences (P3P) project (April 2002), `http://www.w3c.org/TR/P3P/`
7. Bock, K.: Europrise trust certification. Datenschutz und Datensicherheit - DuD 32(9), 610–614 (2008)
8. Trust-E: website, `http://www.truste.com`
9. Ali, M., Sabetta, A., Bezzi, M.: A marketplace for business software with certified security properties. In: Proc. of Cyber Security and Privacy EU Forum (2013)
10. Assert4Soa consortium: Assert4Soa project website, `http://www.assert4soa.eu`
11. Herrmann, D.: Using the Common Criteria for IT security evaluation. Auerbach Publications (2002)
12. Bezzi, M., Sabetta, A., Spanoudakis, G.: An architecture for certification-aware service discovery. In: Proc. of the 1st International Workshop on Securing Services on the Cloud (IWSSC), pp. 14–21. IEEE (2011)
13. Kaluvuri, S.P., Koshutanski, H., Di Cerbo, F., Maña, A.: Security assurance of services through digital security certificates. In: Proc. of the 20th IEEE International Conference on Web Services (ICWS), pp. 539–546. IEEE (2013)
14. Rost, M., Bock, K.: Privacy by Design and the Protection Goals - English translation of Privacy By Design und die Neuen Schutzziele - Grundsätze, Ziele und Anforderungen. DuD 35(1), 30–35 (2011), `https://www.european-privacy-seal.eu/results/articles/BockRost-PbD-DPG-en.pdf` (2010)
15. Hansen, M.: Top 10 mistakes in system design from a privacy perspective and privacy protection goals. In: Camenisch, J., Crispo, B., Fischer-Hübner, S., Leenes, R., Russello, G. (eds.) Privacy and Identity Management for Life. IFIP AICT, vol. 375, pp. 14–31. Springer, Heidelberg (2012)
16. Cavoukian, A.: Privacy by design. IEEE Technology and Society Magazine 31(4), 18–19 (2012)
17. Anisetti, M., Ardagna, C.A., Damiani, E., Saonara, F.: A test-based security certification scheme for web services. ACM Trans. Web 7(2), 5:1–5:41 (2013)
18. Damiani, E., Ardagna, C., Ioini, N.E.: Open source systems security certification. Springer, New York (2009)
19. Frantzen, L., Tretmans, J., Willemse, T.A.C.: A symbolic framework for model-based testing. In: Havelund, K., Núñez, M., Roşu, G., Wolff, B. (eds.) FATES 2006 and RV 2006. LNCS, vol. 4262, pp. 40–54. Springer, Heidelberg (2006)
20. Microsoft: Understanding Retention Tags and Retention Policies (December 2012), `http://technet.microsoft.com/en-us/library/dd297955%28v=exchg.141%29.aspx`
21. IBM: IBM, Enterprise Privacy Authorization Language (EPAL (1.2) (November 2003), `http://www.w3.org/Submission/2003/SUBM-EPAL-20031110`
22. Ardagna, C., Bussard, L., di Vimercati, S.D.C., Neven, G., Pedrini, E., Paraboschi, S., Preiss, F., Samarati, P., Trabelsi, S., Verdicchio, M.: Primelife policy language. In: Proc. of the W3C Workshop on Access Control Application Scenarios, W3C (2009)

23. Chandramouli, R., Blackburn, M.: Automated testing of security functions using a combined model and interface-driven approach. In: Proc. of the 37th Annual Hawaii International Conference on System Sciences (HICSS), Big Island, HI, USA (January 2004)
24. Jürjens, J.: Model-based security testing using UMLsec: A case study. Electronic Notes in Theoretical Computer Science 220(1), 93–104 (2008)
25. Zulkernine, M., Raihan, M.F., Uddin, M.G.: Towards model-based automatic testing of attack scenarios. In: Buth, B., Rabe, G., Seyfarth, T. (eds.) SAFECOMP 2009. LNCS, vol. 5775, pp. 229–242. Springer, Heidelberg (2009)
26. Bozkurt, M., Harman, M., Hassoun, Y.: Testing web services: A survey. Technical Report TR-10-01. Department of Computer Science, King's College London (January 2010)
27. Canfora, G., di Penta, M.: Service-oriented architectures testing: A survey. In: De Lucia, A., Ferrucci, F. (eds.) ISSSE 2006-2008. LNCS, vol. 5413, pp. 78–105. Springer, Heidelberg (2009)
28. Heckel, R., Lohmann, M.: Towards contract-based testing of web services. In: Proc. of the International Workshop on Test and Analysis of Component Based Systems (TACoS), Barcelona, Spain (March 2004)
29. Bentakouk, L., Poizat, P., Zaïdi, F.: Checking the behavioral conformance of web services with symbolic testing and an SMT solver. In: Gogolla, M., Wolff, B. (eds.) TAP 2011. LNCS, vol. 6706, pp. 33–50. Springer, Heidelberg (2011)
30. Endo, A., Simao, A.: Model-based testing of service-oriented applications via state models. In: Proc. of the 8th IEEE International Conference of Service Computing (SCC), Washington, DC, USA (July 2011)
31. Salva, S., Laurencot, P., Rabhi, I.: An approach dedicated for web service security testing. In: Proc. of the 2010 Fifth International Conference on Software Engineering Advances, ICSEA 2010, pp. 494–500. IEEE Computer Society, Washington, DC (2010)
32. Le Traon, Y., Mouelhi, T., Baudry, B.: Testing security policies: going beyond functional testing. In: Proc. of the International Symposium on Software Reliability Engineering, ISSRE, Sweden (2007)
33. Martin, E.: Automated test generation for access control policies. In: Companion to the 21st ACM SIGPLAN Symposium on Object-Oriented Programming Systems, Languages, and Applications, OOPSLA 2006, pp. 752–753 (2006)
34. Mouelhi, T., Fleurey, F., Baudry, B., Le Traon, Y.: A model-based framework for security policy specification, deployment and testing. In: Czarnecki, K., Ober, I., Bruel, J.-M., Uhl, A., Völter, M. (eds.) MoDELS 2008. LNCS, vol. 5301, pp. 537–552. Springer, Heidelberg (2008)

ODBASE 2013 PC Co-Chairs Message

We are delighted to present the proceedings of the 12th International Conference on Ontologies, DataBases, and Applications of Semantics (ODBASE) which was held in Graz (Austria) on September 10-12, 2013. The ODBASE Conference series provides a forum for research and practitioners on the use of ontologies and data semantics in novel applications, and continues to draw a highly diverse body of researchers and practitioners. ODBASE is part of On the Move to Meaningful Internet Systems (OnTheMove) that co-locates with two other conferences: DOA-Trusted Cloud (International Conference on Secure Virtual Infrastructures) and CoopIS (International Conference on Cooperative Information Systems).

Of particular interest in the 2013 edition of the ODBASE Conference are the research and practical experience papers that bridge across traditional boundaries between disciplines such as ontologies, databases, knowledge engineering, data mining, information extraction, and computational linguistics.

In this edition, we received 50 paper submissions and had a program committee of 92 dedicated colleagues, including researchers and practitioners from diverse research areas. Special arrangements were made during the review process to ensure that each paper was reviewed by 3-4 members of different research areas. The result of this effort is the selection of high quality papers: fourteen regular papers (28%), eleven short papers (22%), and six posters (12%). Their themes included studies and solutions to a number of modern challenges such as querying and management of linked data and RDF documents, ontology engineering, semantic matching and mapping, application of data mining techniques, semantic discovery, and probabilistic data management.

Enjoy the reading!

July 2013

Pieter De Leenheer
Deijing Dou

Efficient Parallel Processing of Analytical Queries on Linked Data

Stefan Hagedorn and Kai-Uwe Sattler

Ilmenau University of Technology, Ilmenau, Germany
first.last@tu-ilmenau.de

Abstract. Linked data has become one of the most successful movements of the Semantic Web community. RDF and SPARQL have been established as de-facto standards for representing and querying linked data and there exists quite a number of RDF stores and SPARQL engines that can be used to work with the data. However, for many types of queries on linked data these stores are not the best choice regarding query execution times. For example, users are interested in analytical tasks such as profiling or finding correlated entities in their datasets.

In this paper we argue that currently available RDF stores are not optimal for such scan-intensive tasks. In order to address this issue, we discuss query evaluation techniques for linked data exploiting the features of modern hardware architectures such as big memory and multi-core processors. Particularly, we describe parallelization techniques as part of our CameLOD system. Furthermore, we compare our system with the well-known linked data stores Virtuoso and RDF-3X by running different analytical queries on the DBpedia dataset and show that we can outperform these systems significantly.

Keywords: linked data, parallel query processing, micro benchmark.

1 Introduction

The concept of linked data refers to a set of best practices for publishing and connecting structured data on the Web. In practical use RDF and SPARQL have become de-facto standards for this based on a set of rules outlined by Berners-Lee [23]. With a fast growing popularity and a collection of almost 300 datasets containing more than 31 billion triples [36], the linked data movement can be seen as one of the most successful trends in the Semantic Web community.

Using SPARQL as the de-facto language for querying and processing linked data makes triple store-based engines or federated SPARQL engines a viable or even obvious choice. However, in this paper we argue that there are many tasks and use cases of linked data processing where plain SPARQL processors are not an ideal platform. Examples of such tasks are data profiling to determine (statistical) characteristics of a dataset or to detect links and correlations between triples or graphs. Such operations typically need to scan large portions of the datasets and, therefore benefit only marginally from traditional indexing and even prohibit query processing on distributed sources. An alternative way of handling such tasks on large datasets would be to leverage the MapReduce paradigm. However, this requires higher effort for programming compared to flexible ad-hoc query facilities of RDF databases – at least as long no automatic translation of declarative languages like Pig or Jaql is used. Furthermore, the nature of triple data requires to compute many joins which is a tedious task to implement in MapReduce.

In summary, there is a need for explicitly supporting analytical tasks on linked data in larger scale while providing the flexibility and expressiveness of a SPARQL engine. For an efficient support, query processing should exploit the features of modern hardware architectures such is big memory and multi-core processors.

In this paper, we try to address these challenges by discussing and evaluating techniques for efficient processing of analytical queries on linked data. We present these techniques as part of our CameLOD[1] system but focus on receipts and experiences derived from experimental evaluations on comparison with other approaches.

The remainder of this paper is structured as follows. In Section 2 we describe what use cases we think are important for analytical queries and what their requirements to a RDF query engine are. After a brief discussion of currently known RDF stores and SPARQL engines in Section 3, we describe how we store RDF data and process queries in CameLOD in Section 4. In Section 5 we show the results of experiments that we ran to compare CameLOD to other systems.

2 Use Cases

There are various use cases for analytical queries on linked data. In this section we introduce two of them and show on what operations their efficiency depends.

2.1 Data Profiling

A common analytical task on any dataset is to compute aggregations. Typical tasks include to compute the average value, the minimum or maximum value, or just to count the frequencies of certain values. For relational data this is done for certain columns. The traditional analytical processes require known semantics of the columns and also well integrated data. [6]

However, for linked data these preconditions are not given. The various sources of the datasets lead to different ways of how data and relations are expressed. Hence, analytical tasks on linked data focus on profiling the data in order to get a better understanding and overview of the contained information. This can be achieved in different ways. For example, the predicates (also referred to as *properties*) can help to determine the semantics of the dataset, but also outgoing links and data types are important.

[1] We pronounce it like the mystical British castle Camelot.

One example is to find statements with the same subject and object, but with different predicates. This can happen because of the many vocabularies that are available for the same topic. This makes it hard to formulate an appropriate query that does not miss any statement in its result set. A second example is to simply count the occurrences of the URIs in datasets to get an overview of the used vocabularies and the main content. In order to execute such profiling tasks, the query processing engine has to be very fast and efficient for scan and aggregation operations.

2.2 Correlations

By analyzing linked datasets, one might be interested in correlations within the data. Such correlations can be found in different aspects. First, entities might be correlated. This means, the dataset contains different entities that describe the same "thing". This might happen if different ontologies were used to create the dataset, e.g., when datasets from various sources are combined. Finding such entities is an active field of research of ontology matching and entity resolution [34] .

Additionally, correlations can be found within the data itself. This means, if the dataset contains information about events, finding correlations means to find events that are similar in any form. For example, two events can correlate spatially, by taking place in the same area, or they can correlate temporally, if they take place at the same time. In order to find such correlated evens efficiently, the data store has to provide dedicated indexes for spatial as well as for temporal data. For spatial data, a R-tree [18] has proven to be very efficient. For temporal data one could use e.g., an interval tree.

2.3 Requirements

By analyzing these use cases we can identify several requirements for efficient query processing techniques:

Highly Efficient Scans: Because scans on large portions of triple data are the core operations in the above scenarios, they have to be highly optimized. This includes to avoid expensive IO operations by exploiting in-memory processing, dictionary encoding of string values as well as further compression to reduce memory consumption and increase CPU cache utilization.

Ordered Data: Organizing data in an ordered fashion helps to increase the compression rate of data. Furthermore, using sort-based operators for aggregation/grouping and join computation avoids memory-intensive hashing strategies, allows to leverage caching effects, and to produce early results.

Special-Purpose Indexing: Efficient handling of spatial and/or temporal data requires dedicated index structures such as R-trees for spatial data or interval trees for temporal data. If a generic triple structure is used as the underlying storage scheme, these index structures should capture all relevant data, e.g., based on the predicates of the triples. Furthermore, scan and lookup operations have to leverage these index structures for appropriate query predicates.

Parallel Processing: When using the general data organization scheme of triple stores, partitioning of data and parallelization of querying are fundamental techniques to utilize the processing power of today's hardware architectures and guarantee short response times even for big datasets. The partitioning should be flexible and allow a fine-grained work distribution among the CPU cores.

In the following sections we first discuss to which extent these techniques are supported by existing approaches and then describe how we use them in CameLOD for processing analytical tasks efficiently.

3 Related Work

Since RDF and SPARQL have become widely used to represent and query linked data, more and more storage systems and query engines have been published. In the following section we briefly describe some of them and also introduce some benchmarks created to compare them.

RDF stores and SPARQL engines. While RDF expresses information in triples, the need for a storage system that is optimized for the characteristics of such data arose. This lead to different implementations of triple stores that try to allow fast query execution on the often very large datasets.

One often discussed triple store is RDF-3X by Neumann and Weikum [29]. It stores all data in a centralized table and uses a dictionary to compress long literals to IDs. It further uses B^+-trees as its index structure. RDF-3X creates an index for every possible subject – predicate – object combination. Additionally, aggregate indexes are created in order to support partial queries. Furthermore, to speed up query execution it uses two kinds of statistics. The first one are histograms which are used for selectivity estimations. But as it assumes the predicates to be independent to each other, it cannot always be used. Therefore, the second statistics measure computes frequent join paths. These information are used during query optimization to find the optimal execution plan. In [30] Neumann and Weikum present enhancements to RDF-3X that allow fast updating, versioning, and transaction support.

Another triple store is Hexastore introduced by Weiss et al. in [38]. Like RDF-3X, Hexastore creates an index for each possible combination of S, P, and O and also the partial indexes. These indexes share the same payload, i.e., for a triple consisting of *(s, p, o)*, the PO and OP index both point the the same *s* for a given pair *(p,o)* and *(o,p)*, respectively. To compress data Hexastore also applies dictionary encoding.

In [19] Harris and Gibbins present 3store. 3store maintains a central triple table that stores the hashes of the S, P, and O component. A symbol table allows reverse lookups of the hash values. 3store supports SQL and also SPARQL.

To process SPARQL queries, the table is joined with itself for each triple in a graph pattern [32]. 4store [20] is a storage system and query engine for RDF and is designed to run on a cluster of nodes in a shared nothing architecture.

In [39] Wylog et al. propose dipLODocus. It uses a hybrid storage model, that stores RDF data as a graph but also as lists of literal values. Although this hybrid model causes single inserts and simple lookup queries to be less efficient than in other systems, the authors argue that it speeds-up complex analytical queries.

Virtuoso [11] is a hybrid database server supporting in memory processing and row- and column-wise storage. To support RDF, Virtuoso imports the triples into one big Quad-Table, that stores subject, predicate, and object and a graph identifier. The commercial version also supports spatial indexing. Virtuoso also provides a SPARQL endpoint, which is used very often in production[2]. Other well-known relational database systems were extended to support RDF, too. For example, DB2 [7] and Oracle [31] both provide object-relational mapping features to support RDF. The D2R Server from the D2RQ project [4] can be used to provide access to relational data via SPARQL.

Apache Jena[3] is a Java framework to build semantic web applications by providing an API to process RDF. It can be combined with various third party projects that actually store the data. The Jena project already includes two stores. On the one hand there is the SDB project which can be used to store RDF data in relational SQL-based databases. On the other hand, the TDB project is a high performance storage system, that creates indexes for any combination of subject, predicate, and object. The Fuseki[4] server provides a SPARQL endpoint for Jena. To process incoming queries Jena uses ARQ, which supports SPARQL 1.1.

With RDF HDT [13] Fernández et al. present a binary representation model for RDF data that achieves very high compression rates. However, HDT was not designed as a query engine and does not support query operations but only a single graph pattern at a time.

Some systems also support spatial operations. The OGC has published the GeoSPARQL[5] standard for operations on geographic information. The Parliament triple store [2] already has a implementation of GeoSPARQL. In [22] the authors present Strabon, a RDF store that can handle GeoSPARQL and their own development, stSPARQL.

Parallel Query Execution. Parallelization of (relational) database queries dates back to the eighties [9]. Whereas earlier works focused on multiprocessor systems by proposing techniques for intra-operator parallelism, e.g., particularly for expensive join and sort operators [10,24,15], Graefe proposed in [14] the exchange operator as a more general model for parallelizing other operators. This operator model allows both intra-operator parallelism on partitioned datasets as well

[2] `http://www.w3.org/wiki/SparqlEndpoints`
[3] `http://jena.apache.org`
[4] `http://jena.apache.org/documentation/serving_data/`
[5] `http://www.opengeospatial.org/standards/geosparql`

as horizontal and vertical inter-operator parallelism. Our parallel_do operator described in Section 4.2 is inspired by this idea. More recent approaches aim at exploiting features of modern multicore systems. Examples are dedicated join algorithms [21,1]. The works described in [26,27] provide a detailed analysis of in-memory joins on modern hardware architectures.

In [12] the authors build a SPARQL compiler and optimizer tailored for their Bobox framework. In [16] the authors develop parallel SPARQL engine focusing on join algorithms. The LHD project [37] is a distributed SPARQL engine built for a parallel infrastructure. All these works show that parallel execution can speed up query execution significantly.

SPARQL Benchmarks. There is a number of benchmarks for SPARQL engines and linked data stores. One widely known and accepted benchmark is the Berlin SPARQL Benchmark [3] that was introduced by Bizer and Schultz. The topic of the benchmark is e-commerce where products are offered and purchased and reviewed. The authors run their benchmark among others on D2R server, sesame, and Virtuoso. Their results showed, that none of the tested systems is superior in all queries.

Another benchmark that uses real world data is DBpedia SPARQL benchmark introduced by Morsey et al. in [28]. The queries were selected from the log of queries posed to the official DBpedia SPARQL endpoint.

A third often used benchmark for linked data was proposed by Guo et al. in [17]. The Lehigh University Benchmark (LUBM) generates data in university domain and provides fourteen queries on the data. The goal of the benchmark is to compare knowledge base systems for OWL. This benchmark was extended in [25] to enhance inference and scalability testing.

In the database community the TPC (Transaction Processing Performance Council) benchmarks have become popular. They provide several benchmarks that aim to evaluate different aspects of database systems. For example, the TPC-H benchmark provides ad-hoc queries and concurrent data modifications that simulate real life usage of a database [35]. The TPC benchmarks make heavy use of aggregates and grouping. Then, there is also SP^{2}B [33] which is settled in the DBLP scenario.

To the best of our knowledge, currently there is only one benchmark that concentrates on analytical query processing on linked data. In [8] Demartini et al. present BowlognaBench. The benchmark provides queries that analyze the content of the generated dataset, e.g., to find a student with the highest grades. However, we think that an analytical benchmark should include queries that generate more workload on the tested stores than the currently provided queries do.

4 Query Processing Techniques in CameLOD

In this section, we present several techniques for query processing to address the requirements described above. We start by giving an overview on the architecture of our CameLOD system followed by a detailed discussion of the query processing techniques.

4.1 System Architecture

The idea of our CameLOD system is to build a membase-like system for LOD which fetches data from sources, materializes them in an in-memory database and evaluates queries (including analytical tasks) on this database only. Sources are explicitly registered by users – CameLOD acts only as a caching systems where primary data sources are checked regularly or on-demand for updated data sets which then are downloaded and (re-)inserted.

In CameLOD triples are stored in chunks which are memory-resident blocks in a linked list. Using a linked list of chunks instead of a single big chunk very similar to index sequential file organization simplifies the inserting of triples but represents a trade-off with scan performance. Each chunk itself is organized into columns where a column is a simple array of integer codes matching the L2 cache size. We employ a dictionary encoding scheme for mapping all string values to integer codes; numeric values are encoded as integers, too. The dictionary encoding is order-preserving – thus, we store triples sorted on a given index column and the list of chunks represents a sorted list of all triples. Furthermore, each chunk stores also the minimum and maximum value of the index column in this chunk. Finally, for each column we maintain a pointer to the dictionary used for encoding the values. This allows to have separate dictionaries for the different triple components as well as for different graphs.

CameLOD follows the index-everything approach by using indexes on the subject, predicate, and object components of the triples resulting in materializing each triple three times in different sort order. These indexes are hash-based in-memory indexes mapping dictionary codes to chunks. Note, that a chunk may contain triples for multiple index entries. In addition, spatial and temporal indexes are built for triples with spatial or temporal values. However, in these indexes only the corresponding triples are stored.

In addition to these exact indexes (i.e., each triple value can be found in the index) a further min-value index is maintained. This index is an additional list of chunks where for each triple chunk an entry is stored which consists of a pair of minimal code value of the indexed column and the chunk address. This index is used to support partitioned scans as described below. For this purpose, each triple chunk maintains also a pointer to its min-value chunk. An overview of the storage organization in CameLOD is given in Figure 1.

The query engine is a Basic Graph Pattern (BGP) evaluator consisting of a set of physical algebra operators such as scan, filter, projection, sort, aggregation, several join implementations as well as special-purpose scans such as spatial and temporal index scans. Although there exists a SPARQL frontend based on the ARQ parser, in this paper we focus on the algebra level provided by a simple algebra language. We use the following notation where $\$i$, $\$k$ denote intermediate results in the form of a tuple stream which is produced by query operators and is also used as input for other query operators:

$$\$i := \mathit{operator}(\$k, \mathit{params})$$

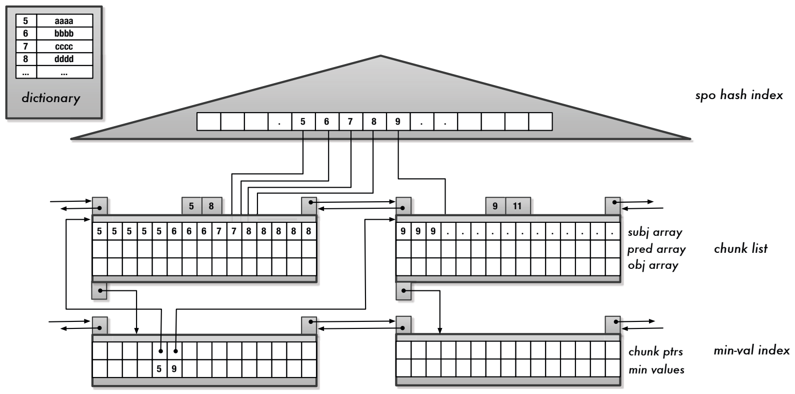


Fig. 1. Storage organization in CameLOD

Furthermore, parameters like `_0`, `_1`, ... denote the first, second, etc. component of a tuple, i.e., for a RDF triple they refer to the subject, predicate, and object component.

Query operators work always on chunks by resembling the vectorized processing model [40], i.e., instead of a one-tuple-at-a-time strategy in the open-next-close cycle, an operator consumes a complete chunk and produces also a chunk as the result of the `next` call. In order to avoid unnecessary memory allocation and copying, operators which do not create new data but restrict existing data (e.g., scan, filter) do not allocate new chunks. This is achieved by using a bitstring per result chunk indicating by the corresponding bit position whether the triple is still in the result or not. Operators such as join, sort, and aggregation create new chunks with the appropriated number of columns. However, theses chunks are only temporary and are not indexed. Note, that operators do not decode the triple values: predicates etc. are translated by the query parser into predicates using integer codes.

4.2 Partition-Aware Scans

Exploiting the capabilities of modern multi-core architectures requires parallel processing of queries. Apart from increasing the throughput of a system by using inter-query parallelism, intra-query parallelization helps to reduce response times of single queries. However, modern architectures draw several challenges to query parallelization in order to overcome limiting factors of scalability:

- low overhead for synchronization (e.g., by using lock-free data structures) and thread creation,
- dealing with load imbalance in partitioning the work caused by data skew,
- keeping the cores busy for better CPU utilization.

For intra-query parallelization, CameLOD provides a special physical operator parallel_do(*qtree*, *n*) which distributes the work of the query tree *qtree* across *n* tasks. Here, the leaf node of a query tree is always an index scan operator. An scan operator accepts the index, an optional range specification, and an optional predicate as arguments. For example

```
$1 := scan(s-index, "http://example.org/Alice", "http://example.org/Paul",
        _1 == "http://xmlns.com/foaf/0.1/mbox");
```

uses the subject index to find triples with a subject in the range from Alice to Paul and a predicate with the value foaf:mbox. This operator returns all chunks containing such triples in increasing order of subjects and initializes the corresponding bitstring accordingly.

To partition the work, i.e., the scan region, with low synchronization overhead and to be able to adjust the query execution to the number of available cores, the parallel_do operator is processed in three steps:

1. The query tree qtree is cloned n−1 times such that each query tree instance can be assigned to its own CPU thread.
2. In the preprocessing phase the min-val index is consulted to assign each query tree instance its own partition in terms of a range of consecutive chunks. This is done by looking up the chunk for the lower range value and getting the corresponding min-val index chunk. By scanning the index chunks, all chunks up to the end of range are collected in a list which is then split by the number n of tasks.
3. Finally, the query is executed by running each query tree instance in a thread. In our implementation we use the Intel TBB library which provides a task abstraction over CPU threads.

The parallel_do operator is also responsible for collecting and merging the individual results from the different query tree instances. This is achieved by maintaining an internal queue of result chunks which are forwarded to the parent operator.

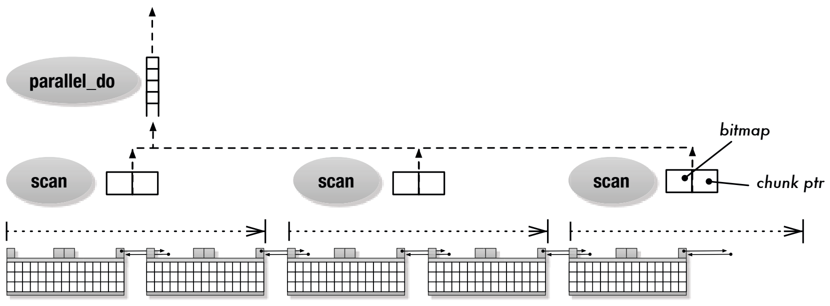


Fig. 2. Parallel scans

Figure 2 illustrates this for the following example query plan where the subject index is again scanned in the range from Alice to Paul in parallel by ten scan instances which also evaluate the given predicate:

```
$1 := scan(s-index, "http://example.org/Alice", "http://example.org/Paul",
        _1 == "http://xmlns.com/foaf/0.1/mbox");
$2 := parallel_do($1, 10);
printer($2);
```

Note, that this does not require any coordination or synchronization between the multiple query tree instances.

4.3 Query Operators

Beside index scans several standard query operators for evaluating SPARQL queries are implemented in CameLOD. This includes physical operators for the SPARQL algebra such as filter, project, sort (order by), distinct, union as well as several join implementations. CameLOD provides different implementations of these operators, e.g., there exist variants of sort, distinct, aggregation, and merge join which exploit the ordering of tuple stream stemming from the chunk list of a given index. In this way, we can avoid both building hash tables and sorting tuples.

In addition, the merge join uses further information from the chunks: because ordered chunks maintain the minimum and maximum values of the sorted column, we can quickly check whether two chunks overlap in their join columns and, therefore, we should join their tuples or simply skip one of the chunks.

A further special join implementation is the star join. This join operator is used to evaluate basic graph patterns like

$$\{\ \ ?\mathsf{s}\ \mathsf{pred}_1\ ?\mathsf{o}_1\ .\ ?\mathsf{s}\ \mathsf{pred}_2\ ?\mathsf{o}_2\ \ldots.\ ?\mathsf{s}\ \mathsf{pred}_n\ ?\mathsf{o}_n\ \}$$

The star join performs a variant of an index join: for each subject value of the first binding the subject index is probed and the corresponding chunk is retrieved. Next, this chunk is scanned for all predicates $\text{pred}_1 \ldots \text{pred}_n$ of the given subject value which may be stored at the same chunk or one of the few following chunks. This is further supported by pushing down filter predicates to the chunk scans. The following query plan implements the query shown above:

```
$1 := scan(s-index);
$2 := star-join($1, s-index, _0, _1, [pred1, pred2, ..., predn]);
```

The parameters of the star join operator denote which index is used (subject index), which column of the input stream provides the lookup values (column 0, i.e., the subject), and which column of the right hand-side input is used for checking the predicate literals pred_i (in this case, the predicate column 1).

The star join operator provides performance benefits by improving cache utilization – not only while joining the various components of a given subject but also while scanning triples in subject order. Furthermore, the star join implementation is available in different forms. Beside the full join a left join variant is provided, too.

4.4 Work Stealing vs. Work Sharing

The parallel_do operator as the core building block for intra-query parallelization in CameLOD is not restricted to partitioned scans. In fact, any partitionable query tree can be parallelized using parallel_do – even operators such as sorting and aggregation at least partially.

However, among the above mentioned limiting factors to scalability load imbalance could become a serious problem. Load imbalance occurs if some query tree instances filter out a significant portion of data earlier and as a result the threads executing these query instances are left idle and have to wait for the busy threads to finish.

Work stealing is a popular scheduling strategy for MIMD computation. According to [5] the base idea dates back to the eighties. In the work stealing model, underutilized processors take the initiative if they need computational work: they try to steal threads from other processors. This results in less frequent thread migration, simply because when all processors are busy, no threads have to be migrated by the scheduler. This model is implemented for example in Intel's Threading Building Blocks (TBB) library[6]. The TBB library abstracts the physical threading model by allowing to treat units of work as *tasks* which are mapped to the underlying threads. In addition, TBB provides several parallel control structures, e.g., parallel loops where the iteration space is divided into multiple sub-spaces which are assigned to tasks.

We rely on this model by assigning query tree instances to tasks instead of threads and let the TBB scheduler do the work. This leads to the following possible strategies for query execution:

Data Flow Graphs: Each operator of a the query plan is implemented by its own task running independently from other operators. Though supported by dedicated TBB control structures (flow graphs), it requires communication between tasks and, therefore, does not seem to be a good idea for query execution in our context.

Work Stealing: Creating enough tasks in parallel_do which can be assigned to physical threads by the scheduler. Each task evaluates its own query tree instance without sharing data with other tasks.

Work Sharing: Similar to work stealing, but as soon as one operator in a query tree instance has finished its work, it tries to get work in terms of chunks from its corresponding neighbor operator in other query tree instances.

In CameLOD, work stealing is used for parallelizing scans as described in Section 4.2, because equal-sized partitions can be easily determined using the min-val indexes. However, if a query tree contains filter and join operators the individual partitions (i.e., range of chunks) may be reduced in their size in different ways resulting in load imbalance. Although it would be possible to redistribute the partitions again, this would require a synchronization among the tasks between two operators which we want to avoid.

[6] `http://threadingbuildingblocks.org`

To address this issue, the work sharing approach can be used particularly for more expensive operators such as joins. For this purpose, we have developed a cooperative variant of the index join which maintains two additional state variables: a role and a sibling list. The role describes whether the particular join instance acts as a giver (if not finished with its own partition) or an asker (otherwise), the sibling list contains the sibling operators, i.e., the corresponding join instances of the other tasks. As soon as a join instance changes its role to asker, it contacts one of its siblings from its sibling list and asks for taking over some work. The giver passes one or more chunks from the end of its partition which are not yet processed. The process stops when all join instances changed their state to asker.

In this approach, the following factors have a strong impact on the efficiency:

– the granularity of work shared between tasks,
– the overhead for registering givers and requesting work from them.

Obviously, there exist a tradeoff between the granularity of work and the overhead: finer granularities (e.g., triples) allow a better load balancing but lead to higher overhead. Because we use chunks as unit of processing in query operators, chunks are the natural granularity level for work sharing.

Askers and givers interact in the following way as part of the `next` function of an operator: each join task maintains a list of its siblings for checking their state as well as registration vector for exchanging work. As soon as a join task has finished the processing of its partition it asks one of the non-finished siblings from the vector by registering in its registration vector. If all tasks have already finished their work the processing stops. Furthermore, after processing a chunk each join tasks checks its registration vector for request to share work. In this case, a yet non-processed chunk is passed to the asker.

4.5 Spatial Indexing and Joins

Usally, B-trees or its variants are used to index column values. However, if the column does not just contain plain values, but rather values that describe an object in a higher domain, a more dedicated index can be used to make use of characteristics of these objects and to perform operations on them. Linked datasets often contain information about the location of events or the description of the shape of an entity. In order to efficiently use these information in query processing, a dedicated index structure is needed.

Statements that describe such spatial features of entities can be identified by the corresponding predicate. There exist various vocabularies that can be used to express such spatial relations. Currently CameLOD can identify spatial predicates from the GeoRSS[7] and W3C[8] ontology. When such predicates are recognized during data import, they will be used together with the corresponding object value to create a shape object. This shape will be indexed in a R-tree

[7] http://www.georss.org/georss/
[8] http://www.w3.org/2003/01/geo/

along with a triple identifier. The triple identifier points to the chunk on which the triple is stored and also contains the row offset within this chunk. Using this R-tree we are able to efficiently process queries that rely on some spatial relationship between objects. This allows to scan for all entities that for instance intersect with a given query region or belong to the k nearest neighbors of a given point. An example query, that scans for all entities that intersect with a given region is shown in the following:

```
$1 := scan(geo-index, intersects[region 46.999 15.355 47.109 15.481]);
printer($1);
```

To process this query CameLOD simply uses the given region to query its spatial index. CameLOD also implements a spatial join operator, which allows to join entities in the dataset regarding a spatial predicate, i.e., some spatial relation. We currently support four spatial join predicates, where a shape r is added to the join result of shape s, if

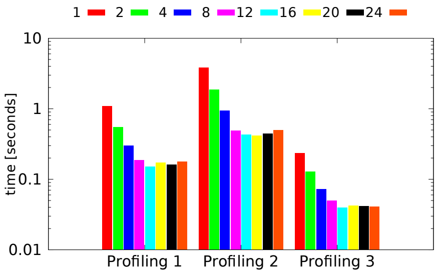
contains: s completely contains r,
location: r completely contains s,
intersects: r intersects with s, and
kNN: r belongs to the k nearest neighbors of s.

This spatial join operator is implemented as an index join. To compute the join result the operator gets the result of a previous operator (e.g., a scan) as input. For each triple in the input a lookup in the spatial index is performed. The type of this R-tree lookup depends on the join predicate. Because a R-tree only stores the minimum bounding rectangle (MBR) of shapes, the result of this index lookup may contain false positives. For example, the MBR of a circle intersects with another shape, but the circle itself does not. In order to prune these false positives, each shape of the index lookup has to be checked explicitly if it matches the join predicate. The pruned lookup result contains all join partners of the current triple which are now passed to the next operator in the plan and the next input triple can be processed.

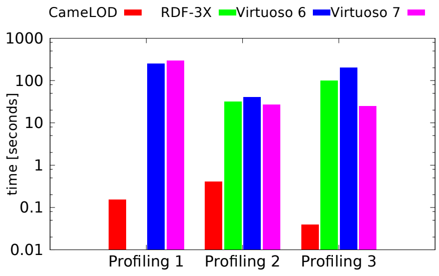
5 Comparison and Evaluation

As we showed in Section 3, currently there is only the BowlognaBench benchmark that focuses on analytical queries. This benchmark contains queries that e.g., aim to find the students with the best grades or the professor supervising the most master theses. These queries are a good starting point for evaluating and comparing RDF stores. However, in our opinion to really show the strengths and weak points of such systems, queries that produce more workload are needed.

In Section 2 we showed some use cases that we believe are typical analytical tasks on linked data. In the following, we are going to show that the none of the existing systems provides adequate support for such tasks and we will further show how the query execution time can be reduced by utilizing parallel execution and special purpose indexes.



(a) Run times for different numbers of tasks



(b) Comparison of different systems

Fig. 3. Run times for profiling queries

5.1 Setup

The tests were run on a server machine with 4 Intel Xeon E5-2630 CPUs with 6 cores each running at 2.30 GHz and 128GB DDR3 RAM. The machine has two 2TB S-ATA disks running in an RAID unit. As operating system we use Ubuntu 12.04 LTS with Kernel 3.2.0-40, 64 bit. We decided to work with the latest DBpedia 3.8[9] dataset. From this dataset we imported 215.000.000 triples into each store and executed different types of analytical queries, that are described in the following. The size of the raw dataset is 30 GB, while CameLOD only needs about 18 GB to store it.

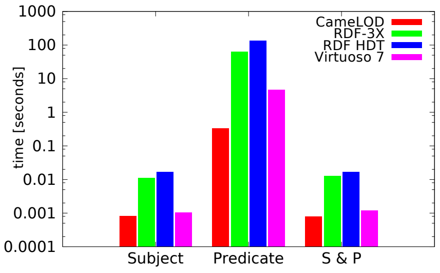
5.2 Experiments

First, we tested how parallelization of query plans impacts execution times. Since CameLOD is the only one of the considered systems that is able to execute the same query in parallel, we can only show the results for different numbers of tasks for CameLOD. We created the following three profiling queries:

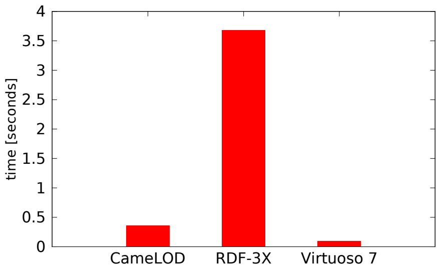
Profiling 1. Count the number of subjects, that have more that 100 predicates
Profiling 2. Count the number of subjects, that are of at least two types[10]
Profiling 3. Get the 50 most often used predicates and their counts for subjects that start with a given prefix

As Figure 3a shows, the execution time decreases significantly with the increase of the number of tasks, but only until a certain point. If the number of tasks gets too high, no more benefit can be achieved and the execution times start to increase. This is due to the increasing overhead of data partitioning and for work sharing. The results of this experiment suggest that the optimal number of tasks for a parallel query is circa half the number of available cores. Further experiments will have to evaluate this issue.

[9] `http://wiki.dbpedia.org/Downloads38`
[10] where type means `http://www.w3.org/1999/02/22-rdf-syntax-ns#type`



(a) Simple scans



(b) Spatial scans

Fig. 4. Query execution times for scans

Next, we compare how different systems can handle data profiling queries. We will compare our CameLOD with RDF-3X 0.3.7, OpenLink Virtuoso 6.1.4 Open Source Edition, and Virtuoso 7 Commercial Edition for glibc 2.12. Although RDF-3X and Virtuoso 6 are disk-based, we chose them because Virtuoso is often used as a SPARQL endpoint for large datasets and RDF-3X is famous for its very fast query execution times and known to be one of the fastest engines currently available. We took the three profiling queries from the previous experiment and adapted them to each system. It showed, that RDF-3X is not capable of GROUP BY operations and implemented COUNT differently from the common sense. Thus, we can only show results for two of our three example queries for RDF-3X. The results of the query executions are shown in Figure 3b.

One can see that CameLOD is significantly faster than the other stores. These results can be achieved because CameLOD is an in-memory system and focuses on scan efficiency. RDF-3X is able to quickly process queries that rely on index lookups, but if the query needs to scan large parts of the dataset, the execution time increases. Virtuoso 6 is much slower than the other systems. For this reason we decided to exclude Virtuoso 6 from the rest of our experiments. Although Virtuoso 7 is an in-memory system, it still is not as fast as CameLOD in this scenario.

To compare the scan efficiency, we conducted a micro benchmark with our CameLOD, RDF-3X, RDF HDT, and Virtuoso 7. The queries we posed for this experiment are simple scans. The scan finds all entities with a given subject, the second one finds all entities with a given predicate, and the last query is a combination of these both queries, i.e., it contains a scan on the subject and another scan on the predicate. We chose the subject[11] and the predicate[12] that occur most frequently in the dataset so that the system has to scan large parts of the dataset. The result of this micro benchmark is shown in Figure 4a. HDT needs the longest time to complete the queries.

[11] `http://dbpedia.org/resource/Alphabetical_list_of_comuni_of_Italy`, 8498 times

[12] `http://dbpedia.org/ontology/wikiPageWikiLink`, $\sim$ 5 Million times

Their very high compression ratio makes it difficult to execute such scan operations as fast as the other systems. RDF-3X can use its B^+-tree index to easily find the needed entries. However, CameLOD is even faster than RDF-3X, because it can also use its ordered indexes to find the first chunk containing the needed entries, but does not need to load the values from disk. Virtuoso 7 is almost as fast as CameLOD in the cases where the subject is scanned. This may be because the optimizer reduces the intermediate result size using the subject and thus speeds up the join and retrieval of the result elements.

Analytical queries also have to filter for entities within ranges of complex types like location or time. To test the support for this kind of queries, a benchmark must provide appropriate queries. BowlognaBench contains queries that test the support for temporal operations, but it misses queries that operate on spatial data. Our spatial query scans for all entities that are located within a given region. This query region is given as a rectangle with the latitude/longitude coordinates of the lower left (29, -127) and upper right (49, -68) corner, which covers the continental area of the USA. The dataset contains ca. 1.3 million points. We executed the query 100 times each on a randomly chosen subset of these points. The results of the spatial queries are shown in Figure 4b. Using a R-tree as spatial index, CameLOD can quickly decide which sub-tree might contain objects that may belong to the query result. In contrast, RDF-3X does not have built-in support for spatial data. Hence, the query had to be expressed as shown in listing 1.1

```
SELECT ?s WHERE {
 ?s <http://w3.org/2003/01/geo/wgs84_pos#lat> ?lat .
 ?s <http://w3.org/2003/01/geo/wgs84_pos#long> ?lon .
 FILTER(?lat > 29 && ?lat < 49 && ?lon > -127 && ?lon < -68) }
```

Listing 1.1. Spatial Query rewritten for RDF-3X

Because RDF-3X cannot use a spatial index for this range query, it has to check every entity that has the latitude and longitude property if their values fall within the given query region. Fetching the entries from disk and the large amount of comparisons takes much longer than the index lookup that CameLOD has to execute. On the other hand, Virtuoso 7[13] is faster than CameLOD in this particular case. The query is not representative as real spatial analytical queries would use distance functions or polygons, which cannot be easily expressed in SPARQL.

6 Conclusion

In this paper we showed that existing RDF systems are not optimal for scan-intensive tasks or for tasks on complex types like spatial or temporal data. Especially analytical queries that require to scan large portions of a dataset can benefit from a scan-optimized and cache-aware engine, because often the dataset

[13] At the time of writing Virtuoso 7 was just released and therefore, we were not yet able to run this experiment with their spatial processing enabled. This will be part of our future work.

can be partitioned easily and then the multi-core architectures of modern hardware can be utilized efficiently.

We introduced our RDF storage engine CameLOD and compared it to the well known systems RDF-3X and Virtuoso. Thanks to our work stealing and work sharing approach, our scan-efficient implementation of the required indexes as well as our special purpose indexes for spatial and temporal data, we are able to execute scan-intensive queries much faster than the other systems. However, because there is no complete benchmark for analytical queries on linked data, we could only run queries that we think are representative for analytical use cases and we leave it to the linked data community to create a benchmark that covers all important features of efficient processing of analytical queries on linked data.

References

1. Albutiu, M.-C., Kemper, A., Neumann, T.: Massively parallel sort-merge joins in main memory multi-core database systems. PVLDB 5(10), 1064–1075 (2012)
2. Battle, R., Kolas, D.: Enabling the geospatial Semantic Web with Parliament and GeoSPARQL. Semantic Web 3(4), 355–370 (2012)
3. Bizer, C., Schultz, A.: Benchmarking the performance of storage systems that expose SPARQL endpoints. In: Proceedings of the ISWC (2008)
4. Bizer, C., Seaborne, A.: D2RQ - treating non-RDF databases as virtual RDF graphs. In: Proceedings of the ISWC (2004)
5. Blumofe, R.D., Leiserson, C.E.: Scheduling multithreaded computations by work stealing. J. ACM 46(5), 720–748 (1999)
6. Böhm, C., Naumann, F., Abedjan, Z., Fenz, D., Grütze, T., Hefenbrock, D., Pohl, M., Sonnabend, D.: Profiling linked open data with ProLOD. In: ICDEW, pp. 175–178. IEEE (2010)
7. Briggs, M., Sahoo, P.R., Raghavendra, G., Appavu, R., Anwar, F.: Resource description framework application development in DB2 10. Technical report (2013)
8. Demartini, G., Enchev, I., Wylot, M., Gapany, J., Cudré-Mauroux, P.: BowlognaBench - Benchmarking RDF Analytics. In: Aberer, K., Damiani, E., Dillon, T. (eds.) SIMPDA 2011. LNBIP, vol. 116, pp. 82–102. Springer, Heidelberg (2012)
9. DeWitt, D., Gray, J.: Parallel Database Systems: The Future of High Performance Database Systems. Communications of the ACM (1992)
10. DeWitt, D.J., Gerber, R.H.: Multiprocessor Hash-Based Join Algorithms. In: VLDB, pp. 151–164 (1985)
11. Erling, O., Mikhailov, I.: Rdf support in the virtuoso dbms. CSSW 221 (2007)
12. Falt, Z., Čermák, M., Dokulil, J., Zavoral, F.: Parallel SPARQL Query Processing Using Bobox. International Journal on Advances in Intelligent Systems 5, 1–13 (2012)
13. Fernández, J.D., Martínez-Prieto, M.A., Gutiérrez, C., Polleres, A., Arias, M.: Binary RDF Representation for Publication and Exchange (HDT). Journal of Web Semantics (2013)
14. Graefe, G.: Encapsulation of Parallelism in the Volcano Query Processing System. In: SIGMOD Conference 1990, pp. 102–111 (1990)
15. Graefe, G.: Parallel Query Execution Algorithms. In: Encyclopedia of Database Systems, pp. 2030–2035 (2009)
16. Groppe, J., Groppe, S.: Parallelizing join computations of SPARQL queries for large semantic web databases. In: SAC, New York (2011)

17. Guo, Y., Pan, Z., Heflin, J.: LUBM: A benchmark for OWL knowledge base systems. Semantic Web Journal 3(2-3), 158–182 (2005)
18. Guttman, A.: R-trees: A dynamic index structure for spatial searching. In: SIGMOD Conference, pp. 47–57 (1984)
19. Harris, S., Gibbins, N.: 3store: Efficient Bulk RDF Storage. In: PSSS (2003)
20. Harris, S., Lamb, N., Shadbolt, N.: 4store: The design and implementation of a clustered rdf store. In: SSWS (2009)
21. Kim, C., Sedlar, E., Chhugani, J., Kaldewey, T., Nguyen, A.D., Blas, A.D., Lee, V.W., Satish, N., Dubey, P.: Sort vs. Hash Revisited: Fast Join Implementation on Modern Multi-Core CPUs. PVLDB 2(2), 1378–1389 (2009)
22. Kyzirakos, K., Karpathiotakis, M., Koubarakis, M.: Strabon: A Semantic Geospatial DBMS. In: Cudré-Mauroux, P., et al. (eds.) ISWC 2012, Part I. LNCS, vol. 7649, pp. 295–311. Springer, Heidelberg (2012)
23. Lee, T.B.: Linked Data - Design Issues (2009), `http://www.w3.org/DesignIssues/LinkedData.html` (last accessed: May 24, 2013)
24. Lu, H., Tan, K.-L., Shan, M.-C.: Hash-Based Join Algorithms for Multiprocessor Computers. In: VLDB, pp. 198–209 (1990)
25. Ma, L., Yang, Y., Qiu, Z., Xie, G.T., Pan, Y., Liu, S.: Towards a complete OWL ontology benchmark. In: Sure, Y., Domingue, J. (eds.) ESWC 2006. LNCS, vol. 4011, pp. 125–139. Springer, Heidelberg (2006)
26. Manegold, S., Boncz, P.A., Kersten, M.L.: What happens during a join? dissecting cpu and memory optimization effects. In: VLDB, pp. 339–350 (2000)
27. Manegold, S., Boncz, P.A., Kersten, M.L.: Optimizing main-memory join on modern hardware. IEEE Trans. Knowl. Data Eng. 14(4), 709–730 (2002)
28. Morsey, M., Lehmann, J., Auer, S., Ngonga Ngomo, A.-C.: DBpedia SPARQL benchmark – performance assessment with real queries on real data. In: Aroyo, L., Welty, C., Alani, H., Taylor, J., Bernstein, A., Kagal, L., Noy, N., Blomqvist, E. (eds.) ISWC 2011, Part I. LNCS, vol. 7031, pp. 454–469. Springer, Heidelberg (2011)
29. Neumann, T., Weikum, G.: RDF-3X: a RISC-style engine for RDF. Proceedings of the VLDB Endowment 1(1), 647–659 (2008)
30. Neumann, T., Weikum, G.: x-RDF-3X: Fast Querying, High Update Rates, and Consistency for RDF Databases 3(1), 256–263 (2010)
31. Oracle. Semantic Technologies Developer's Guide. Technical report (2012)
32. Sakr, S., Al-Naymat, G.: Relational processing of RDF queries. ACM SIGMOD Record 38(4), 23 (2010)
33. Schmidt, M., Hornung, T., Lausen, G., Pinkel, C.: SP2Bench: A SPARQL Performance Benchmark. CoRR (2008)
34. Shvaiko, P., Euzenat, J.: Ontology Matching: State of the Art and Future Challenges. TKDE 25(1), 158–176 (2013)
35. Transcation Processing Performance Council. TPC Benchmark H - Decision Support. Technical report (2013)
36. W3C. Linking Open Data (2013), `http://www.w3.org/wiki/SweoIG/TaskForces/CommunityProjects/LinkingOpenData` (last accessed: May 24, 2013)
37. Wang, X., Tiropanis, T., Davis, H.C.: LHD: optimising linked data query processing using parallelisation. In: Linked Data on the Web, LDOW 2013 (2013)
38. Weiss, C., Karras, P., Bernstein, A.: Hexastore: sextuple indexing for semantic web data management. PVLDB 1(1), 1008–1019 (2008)
39. Wylot, M., Pont, J., Wisniewski, M., Cudré-Mauroux, P.: dipLODocus$_{\text{[RDF]}}$—Short and Long-Tail RDF Analytics for Massive Webs of Data. In: Aroyo, L., Welty, C., Alani, H., Taylor, J., Bernstein, A., Kagal, L., Noy, N., Blomqvist, E. (eds.) ISWC 2011, Part I. LNCS, vol. 7031, pp. 778–793. Springer, Heidelberg (2011)
40. Zukowski, M., Boncz, P.A., Nes, N., Héman, S.: Monetdb/x100 - a dbms in the cpu cache. IEEE Data Eng. Bull. 28(2), 17–22 (2005)

Analysis of User Editing Patterns in Ontology Development Projects

Hao Wang[1], Tania Tudorache[2], Dejing Dou[1], Natalya F. Noy[2], and Mark A. Musen[2]

[1] Department of Computer and Information Science
1202 University of Oregon, Eugene, OR 97403, USA
{csehao,dou}@cs.uoregon.edu
[2] Stanford Center for Biomedical Information Research
Stanford University, 1265 Welch Road, Stanford, CA 94305, USA
{tudorache,nyulas,musen}@stanford.edu

Abstract. The development of real-world ontologies is a complex undertaking, commonly involving a group of domain experts with different expertise that work together in a collaborative setting. These ontologies are usually large scale and have a complex structure. To assist in the authoring process, ontology tools are key at making the editing process as streamlined as possible. Being able to predict confidently what the users are likely to do next as they edit an ontology will enable us to focus and structure the user interface accordingly and to enable more efficient interaction and information discovery. In this paper, we use data mining techniques to investigate whether we are able to predict the next editing operation that a user will make based on the change history. We have analyzed the change logs of two real-world biomedical ontologies, and used association rule mining to find editing patterns using different features. We evaluated the prediction accuracy on a test set of change logs for these two ontologies. Our results indicate that we can indeed predict the next editing operation a user is likely to make. We will use the discovered editing patterns to develop a recommendation module for our editing tools, and to design user interface components that are better fitted with the user editing behaviors.

1 Collaborative Ontology Development and Related Work

Distributed and collaborative development by teams of scientists is steadily becoming a norm rather than an exception for large ontology-development projects. In domains such as biomedicine the majority of large ontologies are authored by groups of domain specialists and knowledge engineers. The development of ontologies such as the Gene Ontology (GO) [7], the National Cancer Institute Thesaurus (NCI Thesaurus) [17] and the International Classification of Diseases (ICD-11) deploys varying collaborative workflows [16]. Many of these projects have several things in common: First the ontologies are very large (e.g., GO has over 39,000 classes; ICD-11 has over 45,000). Second, many users who contribute to the ontologies are not themselves ontology experts and they do not see ontology development as part of their day-to-day jobs. Indeed, the majority of contributors to ICD-11, for example, are medical professionals. At the same time, researchers have long contended that ontology development is a cognitively complex and error prone process [6,15]. The overarching goal of our research on collaborative ontology development is to develop methods that facilitate this process and make it more efficient for users.

In this paper, we explore the use of structured change logs to predict the changes that users are likely to make next. Ontology change logs provide an extremely rich source of information. We and other investigators have used change data from ontologies to measure the level of community activity in biomedical ontologies [11], to migrate data from an old version of an ontology to a new one [10], and to analyze user roles in the process of collaboration [4,5,18,21]. For example, we have demonstrated that we can use the change data to assess the level of stabilization in ontology content [21], to find implicit user roles [5], and to describe the collaboration qualitatively [18]. For example, we found that changes to ICD-11 tend to propagate along the class hierarchy: A user who alters a property value for a class is significantly more likely to make a change to a property value for a subclass of that class than to make an edit anywhere else in the ontology [14]. Similarly, Pesquita and Couto found that structural features and the locations of changes in the Gene Ontology are predictive of where future changes will occur [13]. Cosley and colleagues developed an application that provided specific suggestions to Wikipedia editors regarding new articles to which they might want to contribute [3]. The model aggregated information about the users, such as preferences and edit history. The researchers found that recommendations based on models of the user editing behaviors made the contributors four times more likely to edit any article compared with random suggestions.

We explore the following hypothesis in this paper: In large collaborative ontology development projects, there are patterns of change that persist over time, across different users, and across different projects/ontologies. We evaluate a set of features for ontology changes, such as properties being edited, and evaluate their predictive power. We use association rule mining, a popular data mining technique to extract the patterns based on these features. Association rules provide straightforward guidance to the user-interface designer by suggesting editing patterns. Indeed, we focus on the features and the types of pragmatic patterns that help us build more efficient interface for ontology development.

Specifically, this paper makes the following contributions:

- We develop a data mining method to predict change patterns in collaborative ontology development (Section 3).
- We propose a set of features for association rules that describe change patterns in collaborative ontology development (Section 3).
- We evaluate our method by analyzing a large number of changes from change logs on two large real-world ontology development projects that are run by the World Health Organization (WHO) (Section 4).

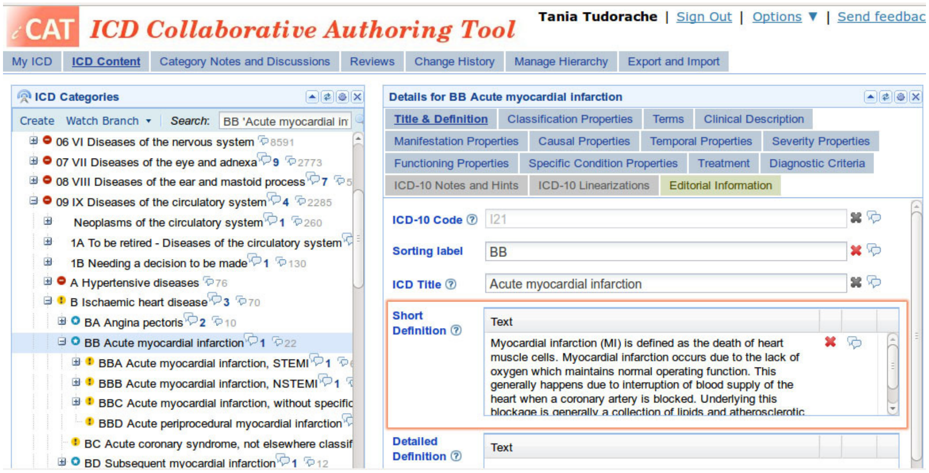
2 Preliminaries

We start by providing background on iCAT, which is a custom-tailored version of WebProtégé [20], a tool that we designed for collaborative ontology development. Today hundreds, if not thousands of users, rely on WebProtégé in their projects (Section 2.1). We then describe the two large ontologies that use WebProtégé and that we used in our evaluation. These are two ontologies in the Family of International Classifications that are developed and maintained by the WHO (Section 2.2). Finally, we provide background on association rule mining (Section 2.3), a technique that we use to find patterns of changes.

2.1 WebProtégé

In this paper, we analyze the data from two ontologies that are developed using a custom-tailored version of WebProtégé [20], which is itself a web-based version of Protégé, the most widely used open-source ontology-editing environment. WebProtégé enables users to edit ontologies in their web browser in a distributed fashion. Users can contribute to the ontology simultaneously, comment on each other's edits, maintain discussions, monitor changes and so on. One of the key features of WebProtégé is the ability of project administrators to custom tailor the user interface to suit the needs of a particular project. Specifically, in this paper we focus on the two ontologies that are developed in iCAT, a version of WebProtégé that is custom tailored to the data model that the WHO uses. Figure 1 shows a screenshot of a panel for editing classes in iCAT. Because each class (e.g., disease description) has as many as 56 properties defined in the data model, iCAT groups these properties visually into "tabs" in the user interface. Each tab is used for editing values for properties in the same *property category.* For example, the `Title & Definition` tab in Figure 1 shows the properties in the category with the same name: `ICD-10 Code`, `Sorting label`, `ICD Title`, `Short Definition` and `Detailed Definition`. The `Clinical Description` tab and property category contains the properties: `Body system`, `Body part` and `Morphology`. iCAT has 15 such tabs and corresponding property categories.

Protégé (and, hence, iCAT) keeps a detailed structured log of changes and their metadata [12] shown in Figure 2. This log contains information about the content of the change and its provenance. We focus on changes to property values in the editing of ICD-11 and ICTM, by far the most frequent operation performed by the users. For example, in ICD-11 from 182,835 total changes, 180,896 are property changes. An example of a property value change tracked by iCAT is shown in the first row of Figure 2: *Replaced Sorting label of DB Acute myocardial infarction. Old value: DB. New value: BB.* For each property-value change, Protégé records the following structure information: *property name*, *class identifier* where the change occurred, the *old* and *new value*, the *author*, and *timestamp* of the change. Based on the user interface configuration (which follows the underlying data model), there is a unique association between a property



BB 'Acute myocardial in

06 VI Diseases of the nervous system 8591
07 VII Diseases of the eye and adnexa 9 2773
08 VIII Diseases of the ear and mastoid process 7 5
09 IX Diseases of the circulatory system 4 2285
Neoplasms of the circulatory system 1 260
1A To be retired - Diseases of the circulatory system
1B Needing a decision to be made 1 130
A Hypertensive diseases 76
B Ischaemic heart disease 3 70
BA Angina pectoris 2 10
BB Acute myocardial infarction 1 22
BBA Acute myocardial infarction, STEMI 1
BBB Acute myocardial infarction, NSTEMI 1
BBC Acute myocardial infarction, without specific
BBD Acute periprocedural myocardial infarction
BC Acute coronary syndrome, not elsewhere classif
BD Subsequent myocardial infarction 1 12

ICD-10 Code	I21
Sorting label	BB
ICD Title	Acute myocardial infarction
Short Definition	Myocardial infarction (MI) is defined as the death of heart muscle cells. Myocardial infarction occurs due to the lack of oxygen which maintains normal operating function. This generally happens due to interruption of blood supply of the heart when a coronary artery is blocked. Underlying this blockage is generally a collection of lipids and atherosclerotic
Detailed Definition	

Fig. 1. The iCAT user interface used for editing the ICD-11 and ICTM ontologies. The left panel shows the disease class hierarchy and the right panel shows the properties of the selected disease class. The properties are organized in different tabs. Each tab (e.g., `Title & Definition`) corresponds to a property category with the same name. A property category contains several properties that are displayed in the respective tab. For example, the `Title & Definition` category contains the properties `ICD-10 Code`, `Sorting label`, `ICD Title`, `Short Definition` and `Detailed Definition`.

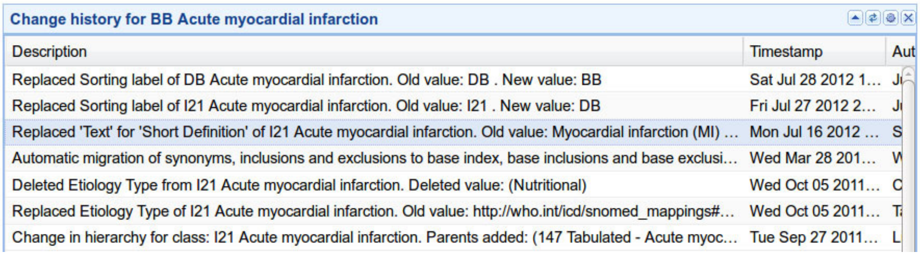
and a property category, so we can easily associate to each change the property category in which it occurred.

However, Protégé is not a requirement for the method that we describe in this paper; it is the presence of a detailed log of changes that is a requirement for the type of data mining that we present. As long as an ontology has a detailed structured log of changes available—regardless of the development environment that its authors use—it is amenable to association rule mining that we describe.

2.2 Ontologies: ICD-11 and ICTM

The 11th Revision of the International Classification of Disease (ICD-11) [1], developed by the World Health Organization, is the international standard for diagnostic classification that health officials in all United Nations member countries use to encode information relevant to epidemiology, health management, and clinical use. Health officials use ICD to compile basic health statistics, to monitor health-related spending, and to inform policy makers. As a result, ICD is an essential resource for health care all over the world. ICD traces its origins to the 19th century and has since been revised at regular intervals. The current in-use version, ICD-10, the 10th revision of the ICD, contains more than 20,000

[1] `http://www.who.int/classifications/icd/ICDRevision/`



Change history for BB Acute myocardial infarction

Description	Timestamp	Aut
Replaced Sorting label of DB Acute myocardial infarction. Old value: DB . New value: BB	Sat Jul 28 2012 1...	J
Replaced Sorting label of I21 Acute myocardial infarction. Old value: I21 . New value: DB	Fri Jul 27 2012 2...	J
Replaced 'Text' for 'Short Definition' of I21 Acute myocardial infarction. Old value: Myocardial infarction (MI) ...	Mon Jul 16 2012 ...	S
Automatic migration of synonyms, inclusions and exclusions to base index, base inclusions and base exclusi...	Wed Mar 28 201...	W
Deleted Etiology Type from I21 Acute myocardial infarction. Deleted value: (Nutritional)	Wed Oct 05 2011...	C
Replaced Etiology Type of I21 Acute myocardial infarction. Old value: http://who.int/icd/snomed_mappings#...	Wed Oct 05 2011...	T
Change in hierarchy for class: I21 Acute myocardial infarction. Parents added: (147 Tabulated - Acute myoc...	Tue Sep 27 2011...	L

Fig. 2. Protégé (and iCAT) track every change in the system in a structured format. A change record has a textual description, a timestamp and an author, as well as other metadata not shown in this screenshot.

terms. The development of ICD-11 represents a major change in the revision process. Previous versions were developed by relatively small groups of experts in face-to-face meetings. ICD-11 is being developed via a web-based process with many experts contributing to, improve, and reviewing the content online [19]. It is also the first version to use OWL as its representation format.

The International Classification of Traditional Medicine (ICTM) is another terminology in the WHO Family of International Classifications. Its structure and development process is very similar to that of ICD-11. However, it is a smaller project, which was started later than the ICD-11 project. Thus, it has benefitted from the experiences of ICD-11 development and it used the tools that were already built for ICD-11. ICTM will provide an international standard terminology as well as a classification system for Traditional Medicine that can be used for encoding information in health records and as a standard for scientific comparability and communication, similar to ICD-11. Teams of domain experts from China, Japan and Korea are collaborating on a web platform with the goal of unifying the knowledge of their own traditional medicines into a coherent international classification. Even though ICTM shares some of the structures with ICD-11, there are many characteristics that are specific only for traditional medicine. ICTM is also developed concurrently in four different languages (English, Chinese, Japanese and Korean).

Data sources. We used the change logs generated by iCAT for both ICD-11 and ICTM. Table 1 shows some statistics about the ontologies themselves and their change logs. As the statistics show, ICTM is a smaller project compared to ICD-11. While ICTM has around 1,500 classes, ICD-11 has over 45,000. ICD-11 has also a deeper class hierarchy with 11 levels, compared to ICTM which has 7 levels. ICTM had a small number of users (12) who were making actively changes for the period of our data, while ICD-11 had 90 such users. The number of property changes which we use for our analysis also differ a lot: ICTM has 21,466 changes, while ICD-11 has 180,896.

Table 1. Ontology and change history statistics for ICTM and ICD-11

Data source characteristic	ICTM	ICD-11
Number of classes	1,511	45,028
Depth of ontology (number of levels)	7	11
Number of users	12	90
Time period	2/7/2011 - 8/21/2011	11/19/2009 - 5/24/2012
Total number of changes	26,607	182,835
Total number of property edit changes	21,466	180,896

2.3 Association Rule Mining

The change logs generated by iCAT provide a wealth of information that we can use to extract change patterns. These patterns of change can enable us to predict what operation the user is likely to perform next, based on her current operation and other features. We used *data mining* for the pattern-discovery task. Data mining is “the process of discovering interesting patterns from a large database” [9].

Association Rule Mining is a data mining technique that explores frequent patterns in large transactional data. The frequent patterns are usually expressed in terms of the combinations of features with certain values that appear together more frequently than the others. Agrawal and his colleagues introduced association rule mining in 1993 [1] and developed the Apriori Algorithm, a fast association rule mining algorithm, in 1994 [2]. The rules were presented in the form of inference rules with quantitative values to indicate the measure of interestingness. In the past decades, researchers have shown that association rules can discover and predict patterns with high efficiency and accuracy [9].

Let D be the set of n data tuples $D = t_1, t_2, ..., t_n$, where $t_i \subseteq \mathcal{I}$. $\mathcal{I} = i_1, i_2, ..., i_m$ is the set of features we want to discover the associations on. Let X and Y be two disjoint events such that $X \subset \mathcal{I}, Y \subset \mathcal{I}$ and $X \cap Y = \emptyset$. An association rule is an implication, $X \Rightarrow Y$, where X is called the antecedent and Y is called the consequent. The antecedent and consequent are conjunction of conditions on disjoint events. The rule provides the information on how likely Y is, given that we observed X. For example, if a user edits the *title* for a class (X), she may be likely to edit its *definition* next (Y). Therefore, association rule mining is a promising approach to predict the next editing operation that a user will make given the previous change logs.

It is common to use qualitative measures of interestingness in order to rank and filter association rules. Two of the most popular measures of interestingness are support and confidence. The **support** of an association rule $supp(X \Rightarrow Y)$ is a measure of how frequently the set of involved items appears in the data. Given event set X, support $s(X)$ is defined as the fraction of tuples $Ti \in D$ such that $X \subset Ti$. For rule $X \Rightarrow Y$,

$$support(X \Rightarrow Y) = P(X \cap Y)$$

is defined as a percentage of data tuples $X \cap Y$; in other words, it is the probability that both X and Y happens. Support is used to filter out association rules with too few occurrences because these rules do not provide enough information about the data and they are usually rare patterns.

Confidence is a measure of how precise these rules are. For rule $X \Rightarrow Y$,

$$Confidence(X \Rightarrow Y) = P(Y|X) = P(X \cap Y)/P(X)$$

In other words, confidence is the probability of Y given that X happens.

3 Method Description

The main goal of our analysis is to predict what the user is likely to do next given her current action. Therefore, our data tuples are *transitions* from one action to the next. Each transition in our set captures two operations from the structured change log: the features describing the current operation that the user performed and the features describing the next operation. We look for co-occurrences of features of the current operation and the next operation. For example, if the user edited the *title* of a class and then edited the *definition*, then the first edit is the current operation and the edit of the definition is the next one.

3.1 Data Preprocessing

We start our data processing by performing the following two preprocessing steps: (1) *feature extraction* and (2) *data aggregation.* The first step extracts the prediction related features from change log entries. The second step aggregates goal-irrelevant data into one data entry which will result in a cleaner and more goal-concentrated result.

Feature Extraction. A typical entry for a property change in the ICD-11 ontology (Figure 2) contains: (1) the information on the user who performed the change, (2) the timestamp, (3) the class identifier on which the change occurred, and (4) a textual description of the change. The latter item, the key source of features for our analysis, looks as follows:

Replaced ***'Text'*** *for* ***'Short Definition'*** *of* ***I21 Acute myocardial infarction. Old value****: Myocardial infarction (MI) is defined as of heart muscle cells. Myocardial infarction occurs ...* ***New value****: Myocardial infarction (MI) is defined as the death of heart muscle cells. Myocardial infarction occurs...*

To use this log entry in our data mining analysis, we need the structured information that the change log provides and the additional features that we extract from the change description text. For example, for the change entry from the example, we extract the property on which the change occurred (`Short Definition`) and we associate to it the property category (`Title & Definition`). We then analyze the next change performed by the same user—represented by a similar string—to extract the same feature about the next operation, as well as the feature reflecting whether the next change occurred in the same class or a different one.

As a result, we generate five features (see Table 2). Two features describe the current change—the *antecedent features*—and three features that describe the next change and the transition information—the *consequent features*.

Table 2. The 5 extracted features for each record in the change log that are used for association mining

Feature	Description of feature
`NAME_OF_PROPERTY` (antecedent)	The name of the edited property (Example: `Short Definition`)
`CATEGORY_OF_PROPERTY` (antecedent)	The category of the edited property (Example: `Title & Definition`)
`NEXT_NAME_OF_PROPERTY` (consequent)	The name of the next edited property (Example: `Body System`)
`NEXT_CATEGORY_OF_PROPERTY` (consequent)	The category of the next edited property (Example: `Clinical Description`)
`NEXT_ENTITY` (consequent)	A boolean flag that describes if the next edit operation is on the same entity as the previous change, or not. (Possible values: `Same` or `Not the same`)

Data Aggregation. The data change log provides abundant information that captures all aspects of user editing behaviors. For example, the user might edit a few characters of a property value, click elsewhere, and then come back and continue editing the same property. This behavior will result in two log entries describing consecutive edits to the same property. In reality though, it is usually just one editing operation from the user's point of view. We define a *consecutive operation* as two editing operations by one user on the same entity and the same property or category of property within a certain time interval (e.g., one hour). We aggregate such consecutive operations into a single operation.

Datasets for Rule Mining. The aggregated data with selected five features are ready for association rule mining. In our work, the data processing step generates four independent data sets. For each ontology (i.e., ICTM or ICD-11), we generated two datasets: one dataset with the operations aggregated based on property category and another one aggregated on property name.

3.2 Association Rule Mining: Apriori Algorithm

We generate the association rules by using the Apriori algorithm [2]. We use WEKA [8], the open source data mining software to generate association rules.

The Apriori algorithm contains two steps: *find all frequent itemsets* and *generate strong association rules from frequent itemsets*. An item is defined as a feature with assignment values, such as `CATEGORY_OF_PROPERTY`= `Temporal Properties`. An itemset is the conjunction of items. The Apriori Algorithm take two threshold as input: *t_support* and *t_confidence*. The *find all frequent itemsets* step will generated all the possible itemset I that satisfy support(I) > *t_support*. It uses the downward closure property of frequent itemsets: itemsets with more features are generated from frequent itemsets with fewer features. This property greatly reduces the search space and lowers the algorithm complexity. After finding all the frequent itemsets, the *find strong association rules* step divides the features in the frequent itemset I into two disjoint sets: antecedent X and consequent Y. We test the condition $confidence(X \Longrightarrow Y) > t\_confidence$ to generate the association rules.

The following is an example of an association rule generated by WEKA based on ICTM data:

$$\texttt{CATEGORY\_OF\_PROPERTY} = \texttt{Temporal Properties}\ 101 \Longrightarrow$$

$$\texttt{NEXT\_CATEGORY\_OF\_PROPERTY} = \texttt{Diagnostic Method}\ \texttt{NEXT\_ENTITY} = \texttt{same}\ 70$$

$$conf : (0.69)$$

This rule indicates an association between the feature `CATEGORY_OF_PROPERTY`, `NEXT_CATEGORY_OF_PROPERTY` and `NEXT_ENTITY`. It shows that the users performed 101 edits in `Temporal Properties`, and 70 of these edits were followed by the edits in `Diagnostic Method` property on the `same` class. Therefore the confidence of this rule is 69% (i.e., 70 divided by 101).

3.3 Prediction Using Association Rules

Recall that our goal is to predict the next editing operation that a user will make given the current change. The association rules show the relationships between users' next editing operations and the previous edits. To simulate the prediction process, we split our data into two sets: a training set and a test set. We generate the association rules based on the training set and assess the confidence values of these rules in the test set. The difference in the confidence values between these two sets will indicate how much the editing patterns drift. Specifically we evaluate the drift along three dimensions: (1) different stages of ontology development over time; (2) different user groups; and (3) different ontologies.

To split the data based on different group of users, we introduce a method that keeps splitting the data randomly by users into training and testing sets until the two data sets satisfy: 1) They are of roughly the same size. 2) The number of users in the two data sets are roughly the same. The advantage of splitting the data in this way is obvious. First, with two sets with roughly equal

Table 3. Number of data entries in the change log before and after aggregation

No. of change entries	ICTM	ICD-11
Original Data	21,466	180,896
Aggregated on *property category*	6,962	53,908
Aggregated on *property name*	10,208	63,654

size we will have enough data for both training and testing data sets. Secondly, users with different numbers of data entries are randomized into both training and testing datasets so that no bias is introduced due to the splitting process. To split the data based on the time, we divide the data roughly in the middle of the dataset to have equal size of data for both training and testing.

We present the results of this evaluation in the following section.

4 Experimental Results

We have applied the data aggregation process to all ICTM and ICD-11 datasets with five selected features. To show the effect of data aggregation, we list the statistics before and after the data aggregation in Table 3. In both the ICTM and ICD-11 datasets, more than half of the data have been aggregated.

4.1 Rule Analysis for Training Data

All the meaningful rules that we generated from the training data fall into the following three types:

Type One

(a) `CATEGORY_OF_PROPERTY` $=$ `A` $\Longrightarrow$
`NEXT_CATEGORY_OF_PROPERTY` $=$ `B` `NEXT_ENTITY` $=$ `Same`

(b) `NAME_OF_PROPERTY` $=$ `A` $\Longrightarrow$
`NEXT_NAME_OF_PROPERTY` $=$ `B` `NEXT_ENTITY` $=$ `Same`

Type Two

(a) `CATEGORY_OF_PROPERTY` $=$ `A` $\Longrightarrow$
`NEXT_CATEGORY_OF_PROPERTY` $=$ `A` `NEXT_ENTITY` $=$ `Not the same`

(b) `NAME_OF_PROPERTY` $=$ `A` $\Longrightarrow$
`NEXT_NAME_OF_PROPERTY` $=$ `A` `NEXT_ENTITY` $=$ `Not the same`

Type Three

(a) `CATEGORY_OF_PROPERTY` $=$ `A` $\Longrightarrow$
`NEXT_CATEGORY_OF_PROPERTY` $=$ `B` `NEXT_ENTITY` $=$ `Not the same`

(b) `NAME_OF_PROPERTY` = `A` $\Longrightarrow$
`NEXT_NAME_OF_PROPERTY` = `B` `NEXT_ENTITY` = `Not the same`

For each rule type, we show the rules generated when aggregating on the *property category* (rules a), and when aggregating on the *property name* (rules b). *Type One* rules capture the case where the user continues to edit the same class, but changes the property category (1a) that she edits or the property name (1b). The transition means that the following edit occurs in a different tab (1a), or in a different field on the form (1b), respectively (Figure 1). *Type Two* rules describe the situation where the user is focused on editing a single property category (2a) or a single property (2b), e.g., `Short Definition`, for different classes: she edits the property for one class and then edits the same property for another class. *Type Three* rules describe the user who edits both in a different entity and a different property category (3a) or property (3b) in the next operation.

In the rest of this section, we show the top five association rules for ICD-11 and ICTM datasets, aggregated on property category.

Example Association Rules from the ICTM Data. Specifically, Table 4 lists the top five association rules generated from the ICTM data aggregated on category of property. We rank the rules by the confidence measures. For example, rule 1 states that after editing property category `Title & Definition`, users will, with probability of 77% (i.e., 2632 divided by 3409), edit the same property category (`Title & Definition`) but on a different entity. The rest rules are interpreted in a similar way. Rule 2 and rule 3 (*Type One*) show that after editing property category `Temporal Property` or `Causal Property`, users likely continue editing the same class, transitioning to property category `Diagnostic Method`. Rule 1 and rule 4 (*Type Two*) indicate that users will keep editing on the same category of property in the next operation even they transit into another entity. These rules show that the editing tasks of these categories of properties usually come as a batch work. During certain period of time users will edit a set of `Title & Definitions` or `Classification Properties` on different entities.

Example Association Rules from the ICD-11 Data. Table 5 lists the top five generated association rules from the ICD-11 data aggregated on category of property. Again, we rank them by the confidence measures. We found that the editing patterns in ICD-11 are different from the ones in ICTM. The first four rules are *Type Two* rules and the fifth rule is a *Type Three* rule. There are no *Type One* rules in this set. The first four rules show that the users of ICD-11 are very likely to keep editing on the same category of property in the next operation even they transit into another entity. Only rule 5 shows that users may change from editing `Diagnostic Method` to `Title & Definition` while they transit into another entity.

Table 4. Top 5 Association Rules from the ICTM Data

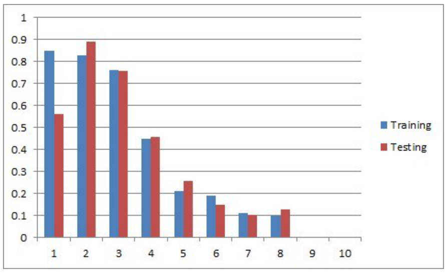
1. CATEGORY_OF_PROPERTY=Title & Definition 3409 $\Longrightarrow$
NEXT_CATEGORY_OF_PROPERTY=Title & Definition NEXT_ENTITY=Not the same 2632 conf:(0.77)
2. CATEGORY_OF_PROPERTY=Temporal Properties 101 $\Longrightarrow$
NEXT_CATEGORY_OF_PROPERTY=Diagnostic Method NEXT_ENTITY=Same 70 conf:(0.69)
3. CATEGORY_OF_PROPERTY=Causal Properties 369 $\Longrightarrow$
NEXT_CATEGORY_OF_PROPERTY=Diagnostic Method NEXT_ENTITY=Same 205 conf:(0.56)
4. CATEGORY_OF_PROPERTY=Classification Properties 1413 $\Longrightarrow$
NEXT_CATEGORY_OF_PROPERTY=Classification Properties NEXT_ENTITY=Not the same 673 conf:(0.48)
5. CATEGORY_OF_PROPERTY=Diagnostic Method 447 $\Longrightarrow$
NEXT_CATEGORY_OF_PROPERTY=Classification Properties NEXT_ENTITY=Not the same 170 conf:(0.38)

Table 5. Top 5 Association Rules from the ICD-11 Data

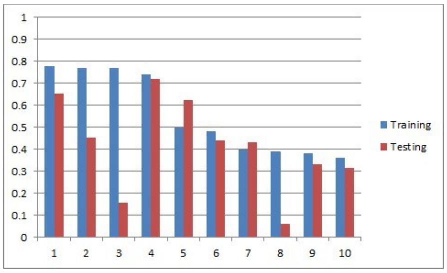
1. CATEGORY_OF_PROPERTY=Classification Properties 16794 $\Longrightarrow$
NEXT_CATEGORY_OF_PROPERTY=Classification Properties NEXT_ENTITY=Not the same 14772 conf:(0.88)
2. CATEGORY_OF_PROPERTY=Clinical Description 1951 $\Longrightarrow$
NEXT_CATEGORY_OF_PROPERTY=Clinical Description NEXT_ENTITY=Not the same 1639 conf:(0.84)
3. CATEGORY_OF_PROPERTY=Editorial Information 4214 $\Longrightarrow$
NEXT_CATEGORY_OF_PROPERTY=Editorial Information NEXT_ENTITY=Not the same 3430 conf:(0.81)
4. CATEGORY_OF_PROPERTY=Title & Definition 23658 $\Longrightarrow$
NEXT_CATEGORY_OF_PROPERTY=Title & Definition NEXT_ENTITY=Not the same 17993 conf:(0.76)
5. CATEGORY_OF_PROPERTY=Diagnostic Method 447 $\Longrightarrow$
NEXT_CATEGORY_OF_PROPERTY=Title & Definition NEXT_ENTITY=Not the same 157 conf:(0.35)

4.2 Prediction Results on the Testing Data

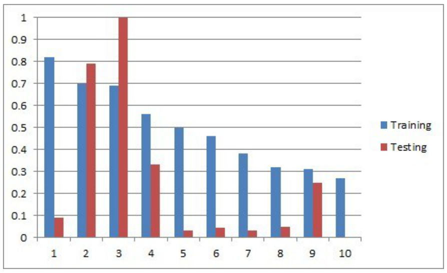
We apply the association rules generated from to training data to the testing data to simulate the prediction process. If more than 10 meaningful rules are generated from the training data, we report top 10 rules based on the measure of confidence. We calculate the confidence values of these rules in the testing data compared with the original confidence values in the training data. The difference in the confidence values between these two sets will indicate how much the editing patterns drift.



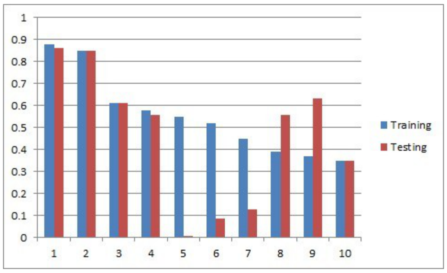
(a) ICD-11 on Category of Property



(b) ICD-11 on Property Name



(c) ICTM on Category of Property



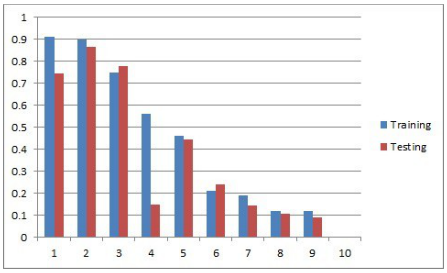
(d) ICTM on Property Name

Fig. 3. Prediction Across User Groups

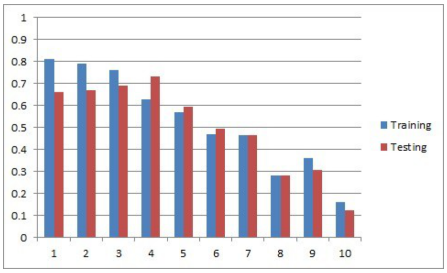
Prediction across User Groups. Figure 3 shows a set of prediction results measured by the confidence values from the training and testing data of ICTM and ICD-11. We can see that the results from the ICD-11 data have good prediction accuracy (i.e., the similarity of confidence values from the training and testing data) and are better than the results from the ICTM data. The prediction results from the ICTM data aggregated on the property name are better than results from the data aggregated on the category of property. It shows the users in ICD-11 have similar editing patterns.

Prediction across Time. In Figure 4, we show the prediction results when we split the training and testing data based on time. We can see that the results from the ICD-11 data aggregated on category of property or property name, and the results from the ICTM data aggregated on the property name have good prediction accuracy. The results from the ICTM data aggregated on the category of property only have 6 rules and the prediction results are not good.

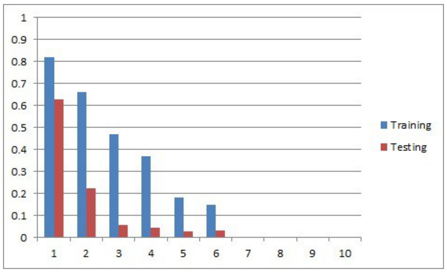
Prediction across Ontologies. We also report the prediction results across ontologies (Figure 5). There are two scenarios in our study: 1) We use the ICD-11 data as the training data and the ICTM data as the testing data. 2) We use the ICTM data as the training data and the ICD-11 data as the testing data. We can see in Figure 5 that prediction results from the data aggregated



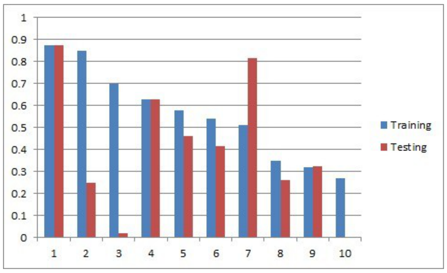
(a) ICD-11 on Category of Property



(b) ICD-11 on Property Name



(c) ICTM on Category of Property



(d) ICTM on Property Name

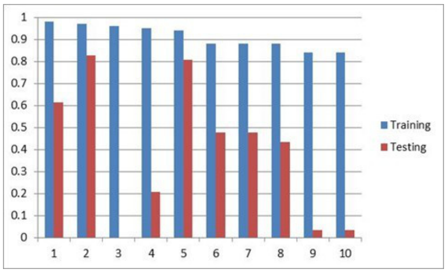
Fig. 4. Prediction Across Time

on property name are still better than the results from the data aggregated on category of property. On the other hand, using ICTM to predict ICD-11 based on the data aggregated on property name is a little better than using ICD-11 to predict ICTM. It may be explained that ICTM use the property names which have been used in ICD-11. Prediction across ontologies might not be as accurate as the ones from across time and across user groups, however they still share plenty of similarities especially on top frequent patterns.

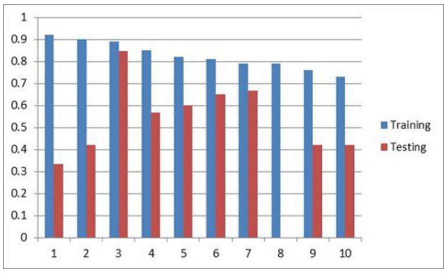
5 Discussion

Our data shows mixed predictive power of association rules. However, the difference in the patterns is in itself useful in analyzing how ontology editing changes from one ontology to another and through different stages of the life cycle. Recall that our key motivation for this work was to find editing patterns so that we can custom-tailor the user interface in order to direct the users' attention to the areas of class definitions that they are likely to edit next.

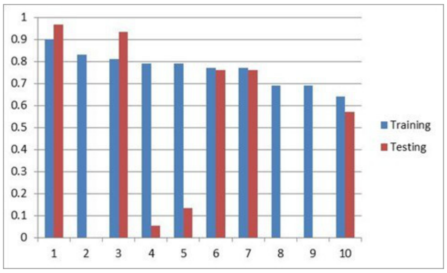
In general, rules that are based on property name rather than property category appear to be more predictive, regardless of whether we look across different users (Figure 3) or different time spans (Figure 4). In other words, the users' transitions between categories of properties are less consistent across the training and test data than their transitions across specific properties. Indeed, a closer look at the data reveals that in the case of ICTM, patterns were particularly



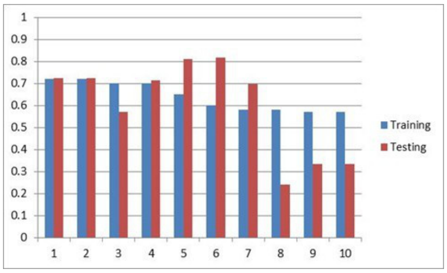
(a) Training on ICD-11 and Testing on ICTM (Category of Property)



(b) Training on ICD-11 and Testing on ICTM (Property Name)



(c) Training on ICTM and Testing on ICD-11 (Category Of Property)



(d) Training on ICTM and Testing on ICD-11 (Property Name)

Fig. 5. Prediction Across Ontologies

different. For example, only two property category (`Title & Definition` and `Classification Properties`) account for almost 90% of the training data in both cases. Thus, half of the users edited only these two property categories, and did so in the beginning of the observation period.

Similarly, while prediction on property names was noticeably better, it is informative to look at the outliers. Consider Figure 3(b), rule 3 in ICD-11: this rule had very high confidence in the training data and was not useful in the testing data. This rule involves a very specialized property, `Primary TAG`, a property describing which group of editors is responsible for the definition of the class. It is likely that the editors would fill out this property only at a particular stage in the lifecycle, and not return to it later. We also selected the association rules based on the measure of support. Comparing the results from rules selected by confidence, the prediction results are similar.

Another reason for better prediction results on property names compared to property categories could be because we had more data in the latter case: consecutive edits on different properties in the same category (a frequent editing pattern) were aggregated in the data preprocessing step (Table 3). In general, the more training data we have, the more reliable the data mining results are.

For the same reason, ICD-11 results show better predictive value than ICTM results: we had considerably more data for ICD-11. It will be interesting to see, as we get more change logs from the users, whether or not the prediction accuracy for ICTM improves as well.

We have also observed that in cross-ontology prediction, ICTM rules were better predictors for ICD-11 rules than the other way around. Indeed, because our data capture the earlier stages of the ICTM development lifecycle, the ICTM editors focused on the more basic properties, and only occasionally ventured into the more advanced properties. Thus, the rules capturing the basic properties carry over well to ICD-11, but there is not enough data—and the patterns are not yet established—for the other properties.

The rules that we identified have given us important feedback on how we can improve the user interface to support the users' editing patterns better. For instance, we have seen that both in ICTM (Table 4, rules 1 and 4), and especially in ICD-11 (Table 5, rules 1-4), users are editing the same property category over and over again, but in different classes. This rule means that the we can improve the editing experience, if our user interface will preserve the same tab when the user switches to a different class. Furthermore, we have identified that the users are editing the same property for different classes very often. The predominance of this type of rule indicates that we should support a tabular type of user interface that makes it easier for users to edit the same property for different classes. For example, a spreadsheet-like tabular interface could contain in each row a column for the class, another for its title and a third one for its definition. This type of interface would very likely speed the data entry and support the editing patterns we have identified in a data-driven way.

The key lesson from the previous observations is the need for including in the analysis not only the change data but also the data on the lifecycle of the ontology and the roles of the user. In our earlier work, we demonstrated that it is possible to distinguish different user roles by analyzing the change data [5]. Integrating these two analyses will likely produce better predictions. For example, we can analyze the change data for each user individually, or for a set of users with the same role, and use data mining on this subset to predict what that particular user is likely to do. Similarly, accounting for the distribution of the features themselves in the data—and the changes in this distribution—will enable us to capture yet another key aspect of changing logs.

6 Conclusions

In this paper, we analyzed the user editing pattern in ontology development projects with the help of data mining algorithms, specially association rule mining. The experiment result shows that the patterns we generated from the ontology editing history data provide useful and straightforward patterns that could help design a better ontology-editing software by focusing the user attention on the components of the complex class definition that the user is likely to edit next. We can use the discovered editing patterns to develop a recommendation module for our editing tools, and to design user interface components that are better fitted with the user editing behaviors.

In order to achieve better predictive power in the data mining, future analyses must also account for different stages in the ontology life cycle, changing user roles, and, potentially, other components. However, our initial results reported in this paper point the way to the data-driven development of user interfaces that alleviates the cognitive load of complex tasks such as ontology editing for domain experts.

Acknowledgments. This work was supported in part by grants GM086587, EB007684, and GM103309 from the US National Institutes of Health. We thank Margaret Qian for her inputs to this project and this paper.

References

1. Agrawal, R., Imielinski, T., Swami, A.N.: Mining association rules between sets of items in large databases. In: ACM SIGMOD International Conference on Management of Data, pp. 207–216 (1993)
2. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules. In: International Conference on Very Large Data Bases, pp. 487–499 (1994)
3. Cosley, D., Frankowski, D., Terveen, L., Riedl, J.: Suggestbot: Using intelligent task routing to help people find work in wikipedia. In: International Conference on Intelligent User Interfaces, pp. 32–41 (2007)
4. De Leenheer, P., Debruyne, C., Peeters, J.: Towards social performance indicators for community-based ontology evolution. In: Workshop on Collaborative Construction, Management and Linking of Structured Knowledge at the International Semantic Web Conference (2009)
5. Falconer, S.M., Tudorache, T., Noy, N.F.: An analysis of collaborative patterns in large-scale ontology development projects. In: International Conference on Knowledge Capture, pp. 25–32 (2011)
6. Gibson, A., Wolstencroft, K., Stevens, R.: Promotion of ontological comprehension: Exposing terms and metadata with web 2.0. In: Workshop on Social and Collaborative Construction of Structured Knowledge (2007)
7. GO Consortium: Creating the Gene Ontology resource: design and implementation. Genome Res. 11(8), 1425–1433 (2001)
8. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The WEKA data mining software: an update. SIGKDD Explorations 11(1), 10–18 (2009)
9. Han, J., Kamber, M.: Data Mining: Concepts and Techniques. Morgan Kaufmann Publishers (2001)
10. Hartung, M., Kirsten, T., Gross, A., Rahm, E.: Onex: Exploring changes in life science ontologies. BMC Bioinformatics 10, 250 (2009)
11. Malone, J., Stevens, R.: Measuring the level of activity in community built bio-ontologies. Journal of Biomedical Informatics 46(1), 5–14 (2013)
12. Noy, N.F., Chugh, A., Liu, W., Musen, M.A.: A framework for ontology evolution in collaborative environments. In: Cruz, I., Decker, S., Allemang, D., Preist, C., Schwabe, D., Mika, P., Uschold, M., Aroyo, L.M. (eds.) ISWC 2006. LNCS, vol. 4273, pp. 544–558. Springer, Heidelberg (2006)
13. Pesquita, C., Couto, F.M.: Predicting the extension of biomedical ontologies. PLoS Computational Biology 8(9) (2012)

14. Pöschko, J., Strohmaier, M., Tudorache, T., Noy, N.F., Musen, M.A.: Pragmatic analysis of crowd-based knowledge production systems with icat analytics: Visualizing changes to the icd-11 ontology. In: AAAI Spring Symposium on Wisdom of the Crowds, pp. 59–64 (2012)
15. Rector, A.L., Drummond, N., Horridge, M., Rogers, J., Knublauch, H., Stevens, R., Wang, H., Wroe, C.: OWL pizzas: Practical experience of teaching OWL-DL: Common errors & common patterns. In: International Conference on Knowledge Engineering and Knowledge Management, pp. 63–81 (2004)
16. Sebastian, A., Noy, N.F., Tudorache, T., Musen, M.A.: A generic ontology for collaborative ontology-development workflows. In: International Conference on Knowledge Engineering and Knowledge Management, pp. 318–328 (2008)
17. Sioutos, N., de Coronado, S., Haber, M., Hartel, F., Shaiu, W., Wright, L.: NCI Thesaurus: A semantic model integrating cancer-related clinical and molecular information. Journal of Biomedical Informatics 40(1), 30–43 (2007)
18. Strohmaier, M., Walk, S., Pöschko, J., Lamprecht, D., Tudorache, T., Nyulas, C., Musen, M.A., Noy, N.F.: How ontologies are made: Studying the hidden social dynamics behind collaborative ontology engineering projects. Journal of Web Semantics 20, 18–34 (2013)
19. Tudorache, T., Falconer, S., Nyulas, C., Noy, N.F., Musen, M.A.: Will semantic web technologies work for the development of ICD-11? In: Patel-Schneider, P.F., Pan, Y., Hitzler, P., Mika, P., Zhang, L., Pan, J.Z., Horrocks, I., Glimm, B. (eds.) ISWC 2010, Part II. LNCS, vol. 6497, pp. 257–272. Springer, Heidelberg (2010)
20. Tudorache, T., Nyulas, C., Noy, N.F., Musen, M.A.: WebProtégé: A collaborative ontology editor and knowledge acquisition tool for the web. Semantic Web Journal 4(1), 89–99 (2013)
21. Walk, S., Pöschko, J., Strohmaier, M., Andrews, K., Tudorache, T., Noy, N.F., Nyulas, C., Musen, M.A.: Pragmatix: An interactive tool for visualizing the creation process behind collaboratively engineered ontologies. International Journal on Semantic Web and Information Systems to appear (Special issue on Visualisation of and Interaction with Semantic Web Data) (2013)

Complexity of Inconsistency-Tolerant Query Answering in Datalog+/–

Thomas Lukasiewicz, Maria Vanina Martinez, and Gerardo I. Simari

Department of Computer Science, University of Oxford, UK
{thomas.lukasiewicz,vanina.martinez,gerardo.simari}@cs.ox.ac.uk

Abstract. The study of inconsistency-tolerant semantics for query answering in ontological languages has recently gained much attention. In this work, we consider three inconsistency-tolerant semantics recently proposed in the literature, namely: consistent query answering, the intersection (also called IAR) semantics, and the intersection of closed repairs (ICR) semantics. We study the data complexity of conjunctive query answering under these semantics for a wide set of tractable fragments of Datalog+/–. The Datalog+/– family of ontology languages covers several important description logics (DLs), bridging the gap in expressive power between database query languages and DLs as ontology languages, and extending the well-known Datalog language in order to embed DLs. Its properties of decidability of query answering and of tractability of query answering in the (data) complexity make Datalog+/– very useful in modeling real-world applications in which inconsistency-tolerant reasoning is essential.

1 Introduction

The issue of inconsistency handling in AI in general and in knowledge representation (KR) in particular has been widely studied in the last three decades. In the database community, the field of *database repairing* and *consistent query answering* (CQA) has gained much attention since the work of [1], which provided a model-theoretic construct of a database *repair*. The most widely accepted semantics for querying a possibly inconsistent database is that of *consistent answers*, which yields the set of tuples (atoms) that appear in the answer to the query over *every* possible repair. CQA enforces consistency at query time as an alternative to enforcing it at the instance level, as conventional data cleaning techniques do. This allows to focus on a smaller portion of the database for which repairs can be computed more easily. Furthermore, techniques have been developed so that it is not necessary to materialize every possible repair. The work of [14] addresses the basic concepts and results of the area of CQA.

More recently, the advent of the Semantic Web, a vision for the future Web, which is strongly based on ontology languages, has led to a resurgence of interest in this area, specially focusing on the development of efficient inconsistency-tolerant reasoning and query answering in DLs and ontology languages. Lately, several works have focused on inconsistency handling for different classes of DLs, adapting and specializing general techniques previously considered for traditional logics [25,18,26,24]. In [20], the adaptation of CQA for *DL-Lite* ontologies is studied. The *intersection semantics* (*ICons*) is presented as a sound approximation of consistent answers, which for the *DL-Lite* family

Table 1. Summary of this paper's data complexity results for BCQ answering in 11 fragments of Datalog+/– under the three different inconsistency-tolerant semantics *Cons*, *ICons*, and *ICR*

Fragment of Datalog+/–		*Cons*	*ICons*	*ICR*
guarded	(Theorem 9)	co-NP-complete [22]	co-NP-complete [22]	co-NP-complete
linear	(Theorem 10)	co-NP-complete [22]	FO-rewritable [23]	co-NP-complete
multi-linear	(Theorem 10)	co-NP-complete	FO-rewritable	co-NP-complete
frontier-one	(Theorem 11)	co-NP-complete	co-NP-complete	co-NP-complete
frontier-guarded	(Theorem 12)	co-NP-complete	co-NP-complete	co-NP-complete
joint-acyclic	(Theorem 14)	co-NP-complete	co-NP-complete	co-NP-complete
weakly-acyclic	(Theorem 13)	co-NP-complete	co-NP-complete	co-NP-complete
sticky	(Theorem 15)	co-NP-complete	FO-rewritable	co-NP-complete
sticky-join	(Theorem 16)	co-NP-complete	FO-rewritable	co-NP-complete
weakly-sticky	(Theorem 17)	co-NP-complete	co-NP-complete	co-NP-complete
weakly-sticky-join	(Theorem 17)	co-NP-complete	co-NP-complete	co-NP-complete

is easier to compute, as well as the *closed ABox* version, which considers the closure of the set of assertional axioms (ABox, or extensional database) by the terminological axioms (TBox, or intensional database). Later, in [21], the authors explore first-order (FO-) rewritability of *DL-Lite* ontologies under *ICons*. The data and combined complexity of the semantics were studied in [27] for a wide spectrum of DLs. Despite the fact that computing consistent answers is an inherently hard problem ([20] shows co-NP completeness even for ground atomic queries in *DL-Lite*), some works identify cases for simple ontologies (within the *DL-Lite* family) for which tractable results can be obtained [6,8]. In [27], the complexity for query answering under inconsistency-tolerant semantics is provided for a wide spectrum of DLs, ranging from tractable ones ($\mathcal{EL}$) to very expressive ones ($\mathcal{SHIQ}$). In [22], consistent query answering and query answering under the *ICons* semantics for linear and guarded Datalog+/– ontologies are studied. An alternative inconsistency-tolerant semantics, called the k-lazy semantics, is also proposed. In [23], FO-rewritability is shown for query answering under the *ICons* semantics for linear Datalog+/–.

In this work, we analyze the (data) complexity of query answering of Boolean conjunctive queries for a set of fragments of Datalog+/– under the three different recent semantics *consistent query answering* (CQA) [20], the *intersection semantics* (*ICons*) [20], and the *intersection of closed repairs* (*ICR*) [8]. We focus on the Datalog+/– family of ontology languages [10], particularly on several fragments of Datalog+/– that guarantee termination of query answering procedures in polynomial time in the data complexity or FO-rewritability. Datalog+/– enables a modular rule-based style of knowledge representation, and it represents syntactical fragments of first-order logic so that answering a BCQ Q under a set Σ_T of Datalog+/– rules for an input database D is equivalent to the classical entailment check $D \cup \Sigma_T \models Q$. The following example illustrates how description logic (DL) knowledge bases are expressed in Datalog+/–.

Example 1. A DL knowledge base consists of a *TBox* and an *ABox*. For example, the knowledge that every conference paper is an article and that every scientist is the author of at least one paper can be expressed by two *TBox* axioms: $\mathit{ConfPaper} \sqsubseteq \mathit{Article}$ and $\mathit{Scientist} \sqsubseteq \exists \mathit{isAuthor}$. The fact that Mark is a scientist can be expressed by the *ABox* axiom $\mathit{Scientist}(\mathit{Mark})$. The *TBox* can be encoded in Datalog+/– rules as follows: $\mathit{ConfPaper}(X) \rightarrow \mathit{Article}(X)$ and $\mathit{Scientist}(X) \rightarrow \exists Y\, \mathit{isAuthor}(X, Y)$. The *ABox* is encoded by an identical set of facts in the database. The *TBox* axiom $\mathit{ConfPaper} \sqsubseteq \neg \mathit{JournalPaper}$ saying that conference papers are not journal papers, can be expressed by the following constraint $\mathit{ConfPaper}(X) \wedge \mathit{JournalPaper}(X) \rightarrow \bot$. Finally, a simple Boolean conjunctive query (BCQ) asking if Mark authors a paper is $\exists X\, \mathit{isAuthor}(\mathit{Mark}, X)$.

Datalog+/–'s properties of decidability and data tractability of query answering, along with its expressive power, make it very useful in many real-world application areas, such as ontology querying, Web data extraction, data exchange, ontology-based data access, and data integration. Studying the complexity of consistent query answering for several semantics and languages can provide insights for finding tractable special cases, average cases, and approximation algorithms for each of these combinations, or even inspirations for inconsistency handling semantics with better computational properties. Table 1 summarizes this paper's data complexity results for BCQ answering in 11 fragments of Datalog+/– under the three different inconsistency-tolerant semantics.

This paper is organized as follows. In Section 2, we recall the basic notions of Datalog+/– ontologies. Section 3 provides the definitions for the three inconsistency-tolerant semantics and some results about their relationships. In Section 4, we provide a complete data complexity analysis for 11 fragments of Datalog+/– and the three inconsistency-tolerant semantics (proofs of all results are given in the extended version of this paper). Section 5 summarizes the main results and gives an outlook on future work.

2 Preliminaries on Datalog+/–

In this Section, we briefly recall the basics on Datalog+/– [10], namely, on relational databases, (Boolean) conjunctive queries ((B)CQs), tuple- and equality-generating dependencies (TGDs and EGDs, respectively), negative constraints, and Datalog+/– ontologies.

Databases and Queries. We assume (i) an infinite universe of *(data) constants* Δ (which constitute the "normal" domain of a database), (ii) an infinite set of *(labeled) nulls* Δ_N (used as "fresh" Skolem terms, which are placeholders for unknown values, and can thus be seen as variables), and (iii) an infinite set of variables $\mathcal{V}$ (used in queries, dependencies, and constraints). Different constants represent different values (*unique name assumption*), while different nulls may represent the same value. We denote by $\mathbf{X}$ sequences of variables $X_1, \ldots, X_k$ with $k \geqslant 0$. We assume a *relational schema* $\mathcal{R}$, which is a finite set of *predicate symbols* (or simply *predicates*). A *term* t is a constant, null, or variable. An *atomic formula* (or *atom*) $\mathbf{a}$ has the form $P(t_1, ..., t_n)$, where P is an n-ary predicate, and $t_1, ..., t_n$ are terms.

A *database (instance)* D for a relational schema $\mathcal{R}$ is a (possibly infinite) set of atoms with predicates from $\mathcal{R}$ and arguments from Δ. A *conjunctive query (CQ)* over $\mathcal{R}$ has the form $Q(\mathbf{X}) = \exists \mathbf{Y}\, \Phi(\mathbf{X}, \mathbf{Y})$, where $\Phi(\mathbf{X}, \mathbf{Y})$ is a conjunction of atoms (possibly equalities, but not inequalities) with the variables $\mathbf{X}$ and $\mathbf{Y}$, and possibly constants, but without nulls. A *Boolean CQ (BCQ)* over $\mathcal{R}$ is a CQ of the form $Q()$, often written as the set of all its atoms, without quantifiers. Answers to CQs and BCQs are defined via *homomorphisms*, which are mappings $\mu\colon \Delta \cup \Delta_N \cup \mathcal{V} \rightarrow \Delta \cup \Delta_N \cup \mathcal{V}$ such that (i) $c \in \Delta$ implies $\mu(c) = c$, (ii) $c \in \Delta_N$ implies $\mu(c) \in \Delta \cup \Delta_N$, and (iii) μ is naturally extended to atoms, sets of atoms, and conjunctions of atoms. The set of all *answers* to a CQ $Q(\mathbf{X}) = \exists \mathbf{Y}\, \Phi(\mathbf{X}, \mathbf{Y})$ over a database D, denoted $Q(D)$, is the set of all tuples $\mathbf{t}$ over Δ for which there exists a homomorphism $\mu\colon \mathbf{X} \cup \mathbf{Y} \rightarrow \Delta \cup \Delta_N$ such that $\mu(\Phi(\mathbf{X}, \mathbf{Y})) \subseteq D$ and $\mu(\mathbf{X}) = \mathbf{t}$. The *answer* to a BCQ $Q()$ over a database D is *Yes*, denoted $D \models Q$, iff $Q(D) \neq \emptyset$.

Given a relational schema $\mathcal{R}$, a *tuple-generating dependency (TGD)* σ is a first-order formula of the form $\forall \mathbf{X} \forall \mathbf{Y}\, \Phi(\mathbf{X}, \mathbf{Y}) \rightarrow \exists \mathbf{Z}\, \Psi(\mathbf{X}, \mathbf{Z})$, where $\Phi(\mathbf{X}, \mathbf{Y})$ and $\Psi(\mathbf{X}, \mathbf{Z})$ are conjunctions of atoms over $\mathcal{R}$ (without nulls), called the *body* and the *head* of σ, denoted $\mathit{body}(\sigma)$ and $\mathit{head}(\sigma)$, respectively. Such σ is satisfied in a database D for $\mathcal{R}$ iff, whenever there exists a homomorphism h that maps the atoms of $\Phi(\mathbf{X}, \mathbf{Y})$ to atoms of D, there exists an extension h' of h that maps the atoms of $\Psi(\mathbf{X}, \mathbf{Z})$ to atoms of D. All sets of TGDs are finite here. Since TGDs can be reduced to TGDs with only single atoms in their heads, in the sequel, every TGD has w.l.o.g. a single atom in its head.

Query answering under TGDs, i.e., the evaluation of CQs and BCQs on databases under a set of TGDs is defined as follows. For a database D for $\mathcal{R}$, and a set of TGDs Σ on $\mathcal{R}$, the set of *models* of D and Σ, denoted $\mathit{mods}(D, \Sigma)$, is the set of all (possibly infinite) databases B such that (i) $D \subseteq B$ and (ii) every $\sigma \in \Sigma$ is satisfied in B. The set of *answers* for a CQ Q to D and Σ, denoted $\mathit{ans}(Q, D, \Sigma)$, is the set of all tuples $\mathbf{a}$ such that $\mathbf{a} \in Q(B)$ for all $B \in \mathit{mods}(D, \Sigma)$. The *answer* for a BCQ Q to D and Σ is *Yes*, denoted $D \cup \Sigma \models Q$, iff $\mathit{ans}(Q, D, \Sigma) \neq \emptyset$. Note that query answering under general TGDs is undecidable [5], even when the schema and TGDs are fixed [9]. Both problems of CQ and BCQ evaluation under TGDs are LOGSPACE-equivalent [17,16]. Moreover, the query output tuple (QOT) problem (as a decision version of CQ evaluation) and BCQ evaluation are AC_0-reducible to each other. Henceforth, we focus only on BCQ evaluation, and any complexity results carry over to the other problems.

Negative constraints (or constraints) υ are first-order formulas $\forall \mathbf{X} \Phi(\mathbf{X}) \rightarrow \bot$, where $\Phi(\mathbf{X})$ (called the *body* of υ) is a conjunction of atoms (without nulls). Under the standard semantics of query answering of BCQs in Datalog+/– with TGDs for each constraint $\forall \mathbf{X} \Phi(\mathbf{X}) \rightarrow \bot$, we only have to check that the BCQ $\Phi(\mathbf{X})$ evaluates to false in D under Σ; if one of these checks fails, then the answer to the original BCQ Q is true, otherwise the constraints can simply be ignored when answering the BCQ Q.

Equality-generating dependencies (or *EGDs*) σ, are first-order formulas $\forall \mathbf{X}\, \Phi(\mathbf{X}) \rightarrow X_i = X_j$, where $\Phi(\mathbf{X})$, called the *body* of σ, denoted $\mathit{body}(\sigma)$, is a (without nulls) conjunction of atoms, and X_i and X_j are variables from $\mathbf{X}$. Such σ is satisfied in a database D for $\mathcal{R}$ iff, whenever there exists a homomorphism h such that $h(\Phi(\mathbf{X}, \mathbf{Y})) \subseteq D$, it holds that $h(X_i) = h(X_j)$. In this work, we assume *non-conflicting* [10] EGDs; this guarantees that EGDs have no impact on the chase with respect to query answering. We usually omit the universal quantifiers in TGDs, negative constraints, and EGDs, and assume that all sets of dependencies and/or constraints are finite.

Datalog+/– Ontologies. A *Datalog+/– ontology* $KB = (D, \Sigma)$, where $\Sigma = \Sigma_T \cup \Sigma_E \cup \Sigma_{NC}$, consists of a finite database D, a set of TGDs Σ_T, a set of non-conflicting EGDs Σ_E, and a set of negative constraints Σ_{NC}.

3 Inconsistency-Tolerant Query Answering

Different works on inconsistency handling in description logics (DLs) allow for inconsistency to occur for different reasons. Depending on the expressive power of the underlying formalism, some works allow for both terminological axioms (TBox) and assertional axioms (ABox) to be inconsistent. In this work, following the idea from database theory in which formulas in Σ are interpreted as integrity constraints expressing the semantics of the data contained in D, we assume that Σ is itself consistent, and inconsistencies can only arise when D and Σ are considered together.

Definition 2 (Consistency). A Datalog+/– ontology $KB = (D, \Sigma)$ is *consistent* iff $\mathit{mods}(D, \Sigma) \neq \emptyset$.

We recall now the notion of *data repairs* of a Datalog+/– ontology from [22]; intuitively these are minimal (in the set-inclusion sense) consistent subsets of D.

Definition 3 (Data Repair). A *data repair* of a Datalog+/– ontology $KB = (D, \Sigma)$ is a set D' such that (i) $D' \subseteq D$, (ii) $\mathit{mods}(D', \Sigma) \neq \emptyset$, and (iii) there is no other set $D'' \subseteq D$ such that $D' \subset D''$ and $\mathit{mods}(D'', \Sigma) \neq \emptyset$. We denote by $\mathit{DRep}(KB)$ the set of all data repairs for KB.

Data repairs play a central role in *consistent answers* for a query to an ontology, which are intuitively the answers relative to each ontology built from a data repair.

Definition 4 (Consistent Answers). Let $KB = (D, \Sigma)$ be a Datalog+/– ontology, and Q be a BCQ. Then, *Yes* is a *consistent answer* for Q to KB, denoted $KB \models_{\mathit{Cons}} Q$, iff it is an answer for Q to each $KB' = (D', \Sigma)$ with $D' \in \mathit{DRep}(KB)$.

The problem of determining if an answer is a consistent answer has been shown to be hard even for simple ontological languages such as *DL-Lite*$_{\mathit{Core}}$ [20] even for more restrictive sublanguages [6]. To avoid intractability, approximations to consistent answers have been developed. The first we consider only takes into account the atoms that are in the *intersection* of all data repairs; it was presented as the *IAR* semantics for *DL-Lite* in [20]. The *intersection semantics* (or *ICons* semantics), as called in a generalization for Datalog+/– ontologies in [22], yields a unique way of repairing inconsistency, and the consistent answers are intuitively the answers that can be obtained from that unique set. This semantics is also called *cautious query answering* in [6].

Definition 5 (Intersection Semantics). Let $KB = (D, \Sigma)$ be a Datalog+/– ontology, and Q be a BCQ. Then, *Yes* is a *consistent answer* for Q to KB *under the intersection semantics*, denoted $KB \models_{\mathit{ICons}} Q$, iff it is an answer for Q to $KB_I = (D_I, \Sigma)$, where $D_I = \bigcap \{D' \mid D' \in \mathit{DRep}(KB)\}$.

A finer approximation to consistent answers is proposed by [7] which, adapted to Datalog+/–, corresponds to closing repairs with respect to Σ_T before intersecting them. In the next definition, $Cn(D', \Sigma_T)$ denotes the set of all ground atoms entailed by a set of atoms D' and a set of TGDs Σ_T. Note that even though the set of constants may be infinite, we restrict the closure of D with respect to Σ_T to the active domain of D (i.e., the constants that appear in the database instance).

Definition 6 (Intersection of Closed Repairs). Let $KB = (D, \Sigma)$ be a Datalog+/– ontology, and Q be a BCQ. Then, *Yes* is a *consistent answer* for Q to KB *under the ICR semantics*, denoted $KB \models_{ICR} Q$, iff it is an answer for Q to $KB_I = (D_I, \Sigma)$, where $D_I = \bigcap \{Cn(D', \Sigma_T) \mid D' \in DRep(KB)\}$.

Soundness for this semantics is shown in [7]; that is, for a given BCQ Q and *DL-Lite*$_{core}$ ontology KB, if $KB \models_{ICR} Q$ then $KB \models_{Cons} Q$. The following proposition shows the equivalence between the two semantics whenever Q is an atomic ground BCQ. Note that the result holds for arbitrary Datalog+/– ontologies.

Proposition 7. *Let KB be a Datalog+/– ontology, and Q be a ground atomic BCQ. We have that* $KB \models_{Cons} Q$ *iff* $KB \models_{ICR} Q$.

Proof. We want to show that $KB \models_{ICR} Q$ iff $KB \models_{Cons} Q$, that is the same as showing that $(\bigcap_{r \in DRep(KB)} Cn(r, \Sigma_T)) \models Q$ iff for every $r \in DRep(KB)$, $(r, \Sigma_T) \models Q$.

($\Rightarrow$) If $(\bigcap_{r \in DRep(KB)} Cn(r, \Sigma_T)) \models Q$ then for every data repair r, either $Q \in \bigcap_{r \in DRep(KB)} Cn(r, \Sigma_T)$ or there exists a set $B \subseteq \bigcap_{r \in DRep(KB)} Cn(r, \Sigma_T)$ such that $(B, \Sigma_T) \models Q$. This means that $(r, \Sigma_T) \models Q$ for every $r \in DRep(KB)$.

($\Leftarrow$) If for every $r \in DRep(KB)$, we have that $(r, \Sigma_T) \models Q$ then, for every r, $Cn(r, \Sigma_T) \models Q$ and therefore $Q \in Cn(r, \Sigma_T)$, then $Q \in \bigcap_{r \in DRep(KB)} Cn(r, \Sigma_T)$ and thus $KB \models_{ICR} Q$. □

In the next section we analyze the data complexity of query answering under the three different inconsistency-tolerant semantics for a variety of fragments of Datalog+/– for which classical query answering of BCQs has already been shown to be tractable. The following proposition helps us prove some of the complexity results; in showing the data complexity in some of the cases; it states that the property of FO-rewritability for classical query answering of BCQs still holds under the intersection semantics.

Proposition 8. *Let KB be a F-Datalog+/– ontology. If query answering is FO-rewritable for the fragment F, then so is query answering under the ICons semantics.*

Proof (sketch). The rewriting algorithm for linear ontologies from [23] can be used to show FO-rewritability of any fragment of Datalog+/– for which query answering is FO-rewritable, since this is the only assumption made in the proofs of the results presented in that paper. □

4 Landscape of Datalog+/– Classes

The problem of entailment is known to be undecidable in the presence of general TGDs [5,13]. However, in the recent years several decidable and even tractable classes have been defined based on different syntactic properties of the TGDs and abstract properties related to the behavior of reasoning mechanisms [3,2,12,4,10]. In this Section we recall several Datalog+/– sublanguages and analyze the data complexity of query answering under the different inconsistency-tolerant semantics described in Section 3. Table 1 shows the summary of the results obtained in this section.

4.1 Guarded Sets of TGDs

We first recall the notion of *guardedness* from [10]. A TGD σ is said to be *guarded* iff it contains an atom in its body that contains all universally quantified variables of σ. The leftmost such atom is the *guard atom* (or *guard*) of σ.

Consistent query answering and query answering under the $ICons$ semantics for guarded Datalog+/– were shown to be co-NP-complete in [22]. To establish membership in co-NP for consistent query answering, we can show a witness that consists of a $D' \in \mathit{DRep}(\mathit{KB})$ for which $(D', \Sigma_T) \not\models Q$. Note that D' can be verified to be a repair in polynomial time, since atoms from D can be added in turn and verified to give rise to an inconsistency, as can $(D', \Sigma_T) \not\models Q$ be checked in polynomial time in the guarded case.

Theorem 9. *Let KB be a guarded Datalog+/– ontology, and Q be a BCQ. Deciding if* $\mathit{KB} \models_{ICR} Q$ *is co-NP-complete.*

Proof (sketch). Membership in co-NP for query answering under the ICR semantics can be shown by guessing and verifying some $D^\star \subseteq Cn(D, \Sigma_T)$ with $(D^\star, \Sigma_T) \not\models Q$, and, for every $\alpha \in Cn(D, \Sigma_T) - D^\star$, some $D'_\alpha \subseteq \mathit{DRep}(\mathit{KB})$ such that $\alpha \notin Cn(D'_\alpha, \Sigma_T)$. Note that the latter implies that $\bigcap_{D' \in \mathit{DRep}(\mathit{KB})} Cn(D', \Sigma_T) \subseteq D^\star$. Hence, $(D^\star, \Sigma) \not\models Q$ implies $\mathit{KB} \not\models_{ICR} Q$. The above guessing and verifying can be done in polynomial time in the data complexity in the guarded case. Membership in co-NP for the $ICons$ semantics can be shown analogously (without computing the closure of the repairs).

Finally, co-NP hardness for consistent query answering is shown in [20] even for ground atomic queries; by Proposition 7, this means that it is also hard for *DL-Lite*$_{core}$ under the ICR semantics, and since this language is a subset of guarded Datalog+/– we can conclude that query answering under the ICR semantics is also co-NP hard in the guarded case. □

A TGD *linear* iff it contains only a single atom in its body. Furthermore, sets of multi-linear TGDs correspond to TGDs whose bodies contain only guards. For linear Datalog+/– co-NP completeness for consistent query answering was shown in [22] and FO-rewritability for linear Datalog+/– ontologies under the $ICons$ semantics was shown in [23]. For multi-linear sets of TGDs we have the following results.

Theorem 10. *Let KB be a multi-linear Datalog+/– ontology, and Q be a BCQ. Deciding if* (a) *KB* $\models_{Cons}$ *Q and if* (b) *KB* $\models_{ICR}$ *Q are both co-NP-complete. Also,* (c) *query answering for BCQs under ICons is FO-rewritable.*

Proof (sketch). Since linear is a sublanguage of multi-linear Datalog+/– [10], we can easily show that consistent query answering is co-NP-hard for the multi-linear case. Membership in co-NP can be shown analogously to the linear/guarded case. Furthermore, by Proposition 8 we now that query answering under the $ICons$ semantics is also FO-rewritable for the multi-linear case. Finally, following the same argument made for guarded Datalog+/–, by Proposition 7 we conclude that query answering under the ICR semantics is also co-NP hard in both the linear and multi-linear case. □

4.2 Frontier-Restricted Sets of TGDs

The frontier of a TGD σ is the set of variables that occur both in the head and body of σ. A TGD σ is *frontier-one* if its frontier is of size one.

Theorem 11. *Let KB be a frontier-one Datalog+/– ontology, and Q be a BCQ. Deciding if* (a) *KB* $\models_{Cons}$ *Q,* (b) *KB* $\models_{ICons}$ *Q, and* (c) *KB* $\models_{ICR}$ *Q are co-NP-complete.*

Proof (sketch). Since classical query answering can be performed in polynomial time in the data complexity for frontier-one ontologies [4], membership in co-NP for the three problems can be shown similar to the way it is done for the guarded case. Hardness can be shown analogously to the hardness proof for the guarded case in [22], with a slight modification in the definition of the set of TGDs to ensure using frontier-one TGDs. Query answering under the $ICons$ semantics can be proved by reduction from the k-clique problem of undirected graphs: the set of TGDs contains the characterization of a clique of size k. Hardness in co-NP under the ICR semantics follows from the fact that consistent query answering for frontier-one Datalog+/– is shown to be co-NP-hard even for ground atomic queries and Proposition 7. □

By noticing that the “shape” of derived facts depends only on how the frontier of the TGDs is mapped (and not on how the whole body is mapped, since only the images of the frontier are used to apply a TGD), one obtains a generalization of both fr1- and guarded-rules: a TGD σ is frontier-guarded (fg) if there is an atom a in its body that contains all variables in its frontier, i.e., $vars(fr(\sigma)) \subseteq vars(a)$.

Theorem 12. *Let KB be a frontier-guarded Datalog+/– ontology, and Q be a BCQ. Deciding if* (a) *KB* $\models_{Cons}$ *Q,* (b) *KB* $\models_{ICons}$ *Q, and* (c) *KB* $\models_{ICR}$ *Q are co-NP-complete.*

Proof (sketch). Given that classical query answering can be performed in polynomial time in the data complexity for frontier-guarded ontologies [4], membership in co-NP for the three problems can easily be shown similarly to the guarded case. Furthermore, hardness for the consistent query answering problem and query answering under the $ICons$ semantics can be proved by reduction from the corresponding problems for frontier-one ontologies, whose data complexity was shown above. Hardness for query answering under the ICR semantics follows from the fact that consistent query answering for frontier-guarded is shown to be co-NP-hard even for ground atomic queries and Proposition 7. □

4.3 Weakly- and Joint-Acyclic Sets of TGDs

Other tractable cases are obtained by restricting possible interactions between TGDs. These interactions have been encoded in different directed graphs that represent encodings of variable sharing between positions in predicates or the encoding of dependencies between TGDs. Here we briefly recall the notion of *(position) dependency graph* [17]. The nodes in a dependency graph represent positions in predicates, i.e., the node (p, i) represents a position i in predicate p. For each TGD σ and each variable X in $\mathit{body}(\sigma)$ occurring in position i, edges with origin (p, i) are built as follows: if X is in the frontier of σ, there is an edge from (p, i) to each position of X in $\mathit{head}(\sigma)$; furthermore, for each existential variable Y in $\mathit{head}(\sigma)$ occurring in position (q, j), there is a *special* edge from (p, i) to (q, j). A set of rules is said to be *w*eakly-acyclic if its dependency graph has no cycle passing through a special edge. Weakly-acyclic sets of TGDs strictly include both sets of full TGDs and acyclic sets of inclusion dependencies [17].

Theorem 13. *Let KB be a weakly-acyclic Datalog+/– ontology, and Q be a BCQ. Deciding if* (a) *KB* $\models_{\mathit{Cons}}$ *Q,* (b) *KB* $\models_{\mathit{ICons}}$ *Q, and* (c) *KB* $\models_{\mathit{ICR}}$ *Q are co-NP-complete.*

Proof (sketch). Membership in co-NP for the three problems can be obtained in the same way that is done for the guarded case since query answering for weakly-acyclic sets of TGDs is in PTIME [15,17]. As for co-NP hardness for consistent query answering, the reduction provided in [22] to show co-NP hardness can be used in this case since the set of TGDs generated in the reduction is trivially a set of weakly-acyclic set of TGDs. Co-NP hardness can be shown for the ICons by slightly modifying the reduction in Theorem 11. Finally, co-NP hardness for the ICR semantics follows directly from the fact that consistent query answering for weakly-acyclic is shown to be co-NP-hard even for ground atomic queries and Proposition 7. □

Joint-acyclicity [19]. This class is obtained by simply shifting the focus from positions to existential variables; that is, replacing the position dependency graph by the existential dependency graph, where the nodes are the existential variables occurring in TGDs. This yields a finer analysis of potentially infinite creations of existential variables.

Theorem 14. *Let KB be a joint-acyclic Datalog+/– ontology, and Q be a BCQ. Deciding if KB* $\models_{\mathit{Cons}}$ *Q, KB* $\models_{\mathit{ICons}}$ *Q, and KB* $\models_{\mathit{ICR}}$ *Q are co-NP-complete.*

Proof (sketch). Membership in co-NP for all three problems can be obtained in the same way that is done for the guarded Datalog+/– since classical query answering for joint-acyclic sets of TGDs is in PTIME [19]. co-NP hardness for all three problems follows directly from the fact that joint-acyclic sets of TGDs strictly generalize weakly-acyclic sets of TGDs. □

4.4 Sticky Sets of TGDs

The stickiness property restricts multiple occurrences of variables (in the same atom or in distinct atoms, i.e., in joins) in the TGD bodies. The class of *sticky* sets of TGDs is defined in [11] by a syntactic criterion that is easily testable, which is as follows. For every database D, assume that during the chase (or forward chaining processing) of D regarding a set Σ of TGDs, we apply a TGD $\sigma \in \Sigma$ which has a variable V appearing more than once in its body. Assume also that V maps (via a homomorphism) on the symbol z, and that by virtue of this application the atom a is generated. In this case, for each atom $b \in \mathit{body}(\sigma)$ we say that a is derived from b. Then, z appears in a, and in all atoms resulting from some derivation sequence starting from a, "sticking" to them (hence the name "sticky" sets of TGDs). We mark the variables that occur in the body of the TGDs of Σ_T according to the following marking procedure. First, for each TGD $\sigma \in \Sigma_T$ and for each variable V in $\mathit{body}(\sigma)$, if there exists an atom a in $\mathit{head}(\sigma)$ such that V does not appear in a, then we mark each occurrence of V in $\mathit{body}(\sigma)$. Now, we apply exhaustively (i.e., until a fixpoint is reached) the following step: for each TGD $\sigma \in \Sigma_T$, if a marked variable in $\mathit{body}(\sigma)$ appears at position π, then for every TGD $\sigma' \in \Sigma_T$ (including the case $\sigma = \sigma'$), we mark each occurrence of the variables in $\mathit{body}(\sigma')$ that appear in $\mathit{head}(\sigma')$ at the same position π. We say that Σ_T is sticky if there is no TGD $\sigma \in \Sigma_T$ such that a marked variable occurs in $\mathit{body}(\sigma)$ more than once.

Theorem 15. *Let KB be a sticky Datalog+/– ontology, and Q be a BCQ. Deciding if* (a) *KB* $\models_{\mathit{Cons}}$ *Q and if* (b) *KB* $\models_{\mathit{ICR}}$ *Q are both co-NP-complete. Furthermore,* (c) *query answering for BCQs under ICons is FO-rewritable.*

Proof (sketch). Hardness in co-NP for consistent query answering follows from the fact that sticky Datalog+/– generalizes *DL-Lite*$_{core}$ (the translation of a *DL-Lite*$_{core}$ *TBox* yields sticky sets of TGDs), a DL for which the problem is co-NP-complete [20]. Membership can be shown analogously to the guarded case since query answering for sticky sets of TGDs is in PTIME [12]. FO-rewritability for the *ICons* semantics follows from Proposition 8. Finally, consistent query answering is co-NP hard even for ground atomic queries [20]; by Proposition 7, this means that it is also hard for *DL-Lite*$_{core}$ under the *ICR* semantics. Therefore, query answering under the *ICR* semantics is also co-NP hard in the sticky case. □

Finally, sticky sets of TGDs are not expressive enough to model simple cases such as the following linear TGD $p(X, Y, X) \rightarrow \exists Z q(Y, Z)$; variable X is marked and therefore the stickiness condition violated. The class of *sticky-join* TGDs extends both linear and sticky TGDs preserving tractability (FO-rewritability) for query answering [12]. As in the case of sticky, sticky-join sets of TGDs are defined by a testable condition based on variable-marking, though a more sophisticated one. For lack of space we omit the formal definition of such marking (cf. [12] for more details on this class).

Theorem 16. *Let KB be a sticky-join Datalog+/– ontology, and Q be a BCQ. Deciding if* (a) *KB* $\models_{\mathit{Cons}}$ *Q and if* (b) *KB* $\models_{\mathit{ICR}}$ *Q are both co-NP-complete. Also,* (c) *query answering for BCQs under ICons is FO-rewritable.*

Proof (sketch). Membership in co-NP for consistent query answering and query answering under the ICR semantics can be shown analogously to the guarded case, since query answering for weakly-sticky sets of TGDs is in PTIME [12]. Hardness for co-NP in both cases follows directly from the fact that sticky-join Datalog+/– generalizes linear Datalog+/–, a fragment for which it is already shown that the problems are co-NP-complete (cf. Theorem 10). FO-rewritability for the $ICons$ semantics follows from Proposition 8. $\square$

The class of *weakly-sticky* [12] TGDs generalizes both weakly-acyclic and sticky sets of TGDs. Intuitively, in a weakly-sticky set of TGDs, the variables that appear more than once in the body of a TGD are either non-marked (as defined above for sticky TGDs), or occur at positions where a finite number of distinct values can appear during the chase. Lastly, we look at weakly-sticky-join [12] sets of TGDs, which generalize both weakly-sticky sets and sticky-join sets.

Theorem 17. *Let KB be a weakly-sticky(-join) Datalog+/– ontology, and Q be a BCQ. Deciding if* (a) *KB* $\models_{Cons}$ *Q,* (b) *KB* $\models_{ICons}$ *Q, and* (c) *KB* $\models_{ICR}$ *Q are co-NP-complete.*

5 Summary and Outlook

In this work, we have studied the problem of inconsistency-tolerant query answering of BCQs for a wide range of tractable fragments of Datalog+/–. For each of the 11 languages chosen, we have determined the data complexity of consistent query answering and of query answering under two of its approximations proposed recently in the literature: $ICons$ (intersection of repairs) and ICR (intersection of closed repairs) semantics. As has been shown in previous work, the problem of computing consistent answers is a hard problem even for very simple languages; therefore, the results obtained for this semantics are not surprising, though the tractability of query answering in these languages at least yields non-deterministic polynomial time upper bounds. On the other hand, we see that for the $ICons$ semantics, there is a large gap in the data complexity, when going from FO-rewritable fragments of Datalog+/– to those for which classical query answering is *PTIME*-complete. Finally, for the chosen languages, the ICR semantics proved to be as hard as the consistent answers to compute. These results are suggestive, showing that it is still necessary to look for alternative inconsistency-tolerant semantics that provide a better tradeoff between quality of answers and efficiency.

The present work can be continued along several lines. For instance, there are several more inconsistency-tolerant semantics that would be interesting to analyze for this set of languages. Furthermore, it would be very important for practical purposes to search for other fragments of Datalog+/– that allow for tractable query answering under any of the current semantics existing in the literature. The complexity results of this paper provide us with a direction for this search.

Acknowledgments. This work was supported by the Engineering and Physical Sciences Research Council (EPSRC) grant EP/J008346/1 "PrOQAW: Probabilistic Ontological Query Answering on the Web", the European Research Council (FP7/2007-2013)/ERC grant 246858 ("DIADEM"), and by a Yahoo! Research Fellowship.

References

1. Arenas, M., Bertossi, L.E., Chomicki, J.: Consistent query answers in inconsistent databases. In: Proc. PODS 1999, pp. 68–79. ACM Press (1999)
2. Baget, J.F., Leclère, M., Mugnier, M.L.: Walking the decidability line for rules with existential variables. In: Proc. KR 2010, pp. 466–476. AAAI Press (2010)
3. Baget, J.F., Leclère, M., Mugnier, M.L., Salvat, E.: Extending decidable cases for rules with existential variables. In: Proc. IJCAI 2009, pp. 677–682 (2009)
4. Baget, J.F., Mugnier, M.L., Rudolph, S., Thomazo, M.: Walking the complexity lines for generalized guarded existential rules. In: Proc. IJCAI 2011, pp. 712–717. IJCAI/AAAI Press (2011)
5. Beeri, C., Vardi, M.Y.: The implication problem for data dependencies. In: Even, S., Kariv, O. (eds.) ICALP 1981. LNCS, vol. 115, pp. 73–85. Springer, Heidelberg (1981)
6. Bienvenu, M.: First-order expressibility results for queries over inconsistent *DL-Lite* knowledge bases. In: Proc. DL 2011. CEUR Workshop Proceedings, vol. 745. CEUR-WS.org (2011)
7. Bienvenu, M.: Inconsistency-tolerant conjunctive query answering for simple ontologies. In: Proc. DL 2012. CEUR Workshop Proceedings, vol. 846. CEUR-WS.org (2012)
8. Bienvenu, M.: On the complexity of consistent query answering in the presence of simple ontologies. In: Proc. AAAI 2012, pp. 705–711. AAAI Press (2012)
9. Calì, A., Gottlob, G., Kifer, M.: Taming the infinite chase: Query answering under expressive relational constraints. In: Proc. KR 2008, pp. 70–80. AAAI Press (2008)
10. Calì, A., Gottlob, G., Lukasiewicz, T.: A general Datalog-based framework for tractable query answering over ontologies. J. Web Sem. 14, 57–83 (2012)
11. Calì, A., Gottlob, G., Pieris, A.: Advanced processing for ontological queries. Proceedings of the VLDB Endowment 3(1/2), 554–565 (2010)
12. Calì, A., Gottlob, G., Pieris, A.: Query answering under non-guarded rules in Datalog+/-. In: Hitzler, P., Lukasiewicz, T. (eds.) RR 2010. LNCS, vol. 6333, pp. 1–17. Springer, Heidelberg (2010)
13. Chandra, A.K., Lewis, H.R., Makowsky, J.A.: Embedded implicational dependencies and their inference problem. In: Proc. STOC 1981, pp. 342–354. ACM Press (1981)
14. Chomicki, J.: Consistent query answering: Five easy pieces. In: Schwentick, T., Suciu, D. (eds.) ICDT 2007. LNCS, vol. 4353, pp. 1–17. Springer, Heidelberg (2006)
15. Dantsin, E., Eiter, T., Gottlob, G., Voronkov, A.: Complexity and expressive power of logic programming. ACM Comput. Surv. 33(3), 374–425 (2001)
16. Deutsch, A., Nash, A., Remmel, J.B.: The chase revisited. In: Proc. PODS 2008, pp. 149–158. ACM Press (2008)
17. Fagin, R., Kolaitis, P.G., Miller, R.J., Popa, L.: Data exchange: Semantics and query answering. Theor. Comp. Sci. 336(1), 89–124 (2005)
18. Huang, Z., van Harmelen, F., ten Teije, A.: Reasoning with inconsistent ontologies. In: Proc. IJCAI 2005, pp. 354–359. Professional Book Center (2005)
19. Krötzsch, M., Rudolph, S.: Extending decidable existential rules by joining acyclicity and guardedness. In: Proc. IJCAI 2011, pp. 963–968. IJCAI/AAAI Press (2011)
20. Lembo, D., Lenzerini, M., Rosati, R., Ruzzi, M., Savo, D.F.: Inconsistency-tolerant semantics for description logics. In: Hitzler, P., Lukasiewicz, T. (eds.) RR 2010. LNCS, vol. 6333, pp. 103–117. Springer, Heidelberg (2010)
21. Lembo, D., Lenzerini, M., Rosati, R., Ruzzi, M., Savo, D.F.: Query rewriting for inconsistent DL-lite ontologies. In: Rudolph, S., Gutierrez, C. (eds.) RR 2011. LNCS, vol. 6902, pp. 155–169. Springer, Heidelberg (2011)

22. Lukasiewicz, T., Martinez, M.V., Simari, G.I.: Inconsistency handling in Datalog+/– ontologies. In: Proc. ECAI 2012, pp. 558–563. IOS Press (2012)
23. Lukasiewicz, T., Martinez, M.V., Simari, G.I.: Inconsistency-tolerant query rewriting for linear Datalog+/–. In: Barceló, P., Pichler, R. (eds.) Datalog 2.0 2012. LNCS, vol. 7494, pp. 123–134. Springer, Heidelberg (2012)
24. Ma, Y., Hitzler, P.: Paraconsistent reasoning for OWL 2. In: Polleres, A., Swift, T. (eds.) RR 2009. LNCS, vol. 5837, pp. 197–211. Springer, Heidelberg (2009)
25. Parsia, B., Sirin, E., Kalyanpur, A.: Debugging OWL ontologies. In: Proc. WWW 2005, pp. 633–640. ACM Press (2005)
26. Qi, G., Du, J.: Model-based revision operators for terminologies in description logics. In: Proc. IJCAI 2009, pp. 891–897 (2009)
27. Rosati, R.: On the complexity of dealing with inconsistency in description logic ontologies. In: Proc. IJCAI 2011, pp. 1057–1062. IJCAI/AAAI Press (2011)

Preference-Based Query Answering in Probabilistic Datalog+/– Ontologies

Thomas Lukasiewicz, Maria Vanina Martinez, and Gerardo I. Simari

Department of Computer Science, University of Oxford, UK
{thomas.lukasiewicz,vanina.martinez,gerardo.simari}@cs.ox.ac.uk

Abstract. The incorporation of preferences into information systems, such as databases, has recently seen a surge in interest, mainly fueled by the revolution in Web data availability. Modeling the preferences of a user on the Web has also increasingly become appealing to many companies since the explosion of popularity of social media. The other surge in interest is in modeling uncertainty in these domains, since uncertainty can arise due to many uncontrollable factors. In this paper, we propose an extension of the Datalog+/– family of ontology languages with two models: one representing user preferences and one representing the (probabilistic) uncertainty with which inferences are made. Assuming that more probable answers are in general more preferable, this raises the question of how to rank answers to a user's queries, since the preference model may be in conflict with the preferences induced by the probabilistic model—the need thus arises for preference combination operators. We propose two specific operators and study their properties in terms of behavior and running time. Finally, we provide an algorithm for ranking answers based on the iteration of the well-known skyline answers to a query and show that, under certain conditions, it runs in polynomial time in the data complexity.

1 Introduction

There has recently been a marked push in the research and development of technology surrounding the Web and, perhaps most centrally, its vast repositories of data. Ontology and query languages are examples of such technology, used to share, integrate, and query large-scale and less structured data and knowledge bases, such as those occurring on the Web. One of the central issues in this domain is that Web search is still centered around the paradigm of receiving keywords from a user and returning a list of links to Web documents that are considered to be pertinent. *Semantic search*, on the other hand, has been proposed as an evolution of this paradigm that identifies objects, rather than whole documents, as candidates to answering the users' queries. At the same time, we have recently been witnessing another revolution in the rapid growth of what is generally referred to as the *Social Web* (which is often also implicitly connected to the Semantic Web [1]); the Social Web is centered around a (mostly) loosely coupled set of platforms that people make use of with the objective of sharing, viewing, and searching for information (in the form of pictures, text documents of varying lengths, videos, etc.) in a social and sometimes collaborative environment. The use of semantic search in the Social Web is of central importance, due to the missing link structure between Web pages, which is well-known from ranking (such as PageRank) in standard Web search. In addition, the fundamentally human component of these systems makes each user's *personal preferences* have a much more prevalent role than what was observed before this paradigm shift. Finally, the presence of uncertainty in the Web in general is undeniable [2–5]: information integration (as in travel sites that query multiple sources to find hotels and flights), automatic processing of Web data (analyzing an HTML document often involves uncertainty), as well as inherently uncertain data (such as user comments) are all examples of uncertainty that must be dealt with in answering queries in the Social Web. The current challenge for Web search is therefore inherently linked to: (1) leveraging the social components of Web content towards the development of some form of semantic search and query answering on the Web as a whole, and (2) dealing with the presence of uncertainty in a principled way throughout the process.

In this paper, we develop a novel integration of ontology languages with both preference and uncertainty management mechanisms by developing an extension of the Datalog+/– family of ontology languages [6] with a preference model over the consequences of the ontology, as well as a probabilistic model that assigns probabilities to them—note that all previous work on extending Datalog+/– with uncertainty (cf. [7] and references therein) does not deal with preferences. The former is assumed to model a user's (or group of users') preferences, while the latter is assumed to model the uncertainty in the domain. The specific mechanisms by which these models are given are left unspecified, since our focus is on the study of how to rank answers to a query when the two models may be in disagreement regarding the ranking: assuming that higher-probability consequences are more preferable than lower-probability ones, it is clear that such situations can arise. The main contributions of this paper are:
(i) the introduction of the PP-Datalog+/– framework, to our knowledge the first extension of ontology languages with both preferences as in relational databases and probabilistic uncertainty;
(ii) the formalization of *preference combination* operators, which take a preference relation in the form of a strict partial order and a score-based preference relation (a weak order) and produce a new preference relation satisfying certain basic properties;
(iii) the development of two specific preference combination algorithms, called CombinePrefs and CombinePrefsRank, which are shown to satisfy the requirements of a preference combination operator as well as several other desirable properties, among which are several postulates proposed in the literature for a less general case; and
(iv) an algorithm for answering k-rank queries, a generalization of top-k queries based on the iterative computation of classical skyline answers, for disjunctions of atomic queries (DAQs), along with proofs of correctness and running time showing that answering DAQs in PP-Datalog+/– can be done in polynomial time in the data complexity modulo the cost of computing probabilities.

The rest of this paper is organized as follows. In Section 2, we present preliminary concepts on classical Datalog+/–. Section 3 introduces the general preference and probabilistic models that are used in this work, motivates the need to combine the user preferences with those induced by probability values by means of preference combination operators, and then goes on to present the syntax and semantics of PP-Datalog+/–. In Section 4, we present algorithms for the implementation of two preference combination operators, and study their semantic and computational properties. Section 5 presents skyline and k-rank queries and a basic algorithm to answer them, and proves its correctness and running time, showing that under certain conditions k-rank queries can be answered in polynomial time in the data complexity. Finally, Sections 6 and 7 discuss related work and conclusions, respectively.

2 Preliminaries on Datalog+/–

First, we briefly recall some basics on Datalog+/– [6], namely, on relational databases, (Boolean) conjunctive queries ((B)CQs), tuple- and equality-generating dependencies (TGDs and EGDs, respectively), negative constraints, the chase procedure, and ontologies in Datalog+/–.

Databases and Queries. We assume (i) an infinite universe of *(data) constants* Δ (which constitute the "normal" domain of a database), (ii) an infinite set of *(labeled) nulls* Δ_N (used as "fresh" Skolem terms, which are placeholders for unknown values, and can thus be seen as variables), and (iii) an infinite set of variables $\mathcal{V}$ (used in queries, dependencies, and constraints). Different constants represent different values (*unique name assumption*), while different nulls may represent the same value. We assume a lexicographic order on $\Delta \cup \Delta_N$, with every symbol in Δ_N following all symbols in Δ. We denote by $\mathbf{X}$ sequences of variables $X_1, \ldots, X_k$ with $k \geqslant 0$. We assume a *relational schema* $\mathcal{R}$, which is a finite set of *predicate symbols* (or simply *predicates*). A *term* t is a constant, null, or variable. An *atomic formula* (or *atom*) $\mathbf{a}$ has the form $P(t_1, ..., t_n)$, where P is an n-ary predicate, and $t_1, ..., t_n$ are terms.

A *database (instance)* D for a relational schema $\mathcal{R}$ is a (possibly infinite) set of atoms with predicates from $\mathcal{R}$ and arguments from Δ. A *conjunctive query (CQ)* over $\mathcal{R}$ has the form $Q(\mathbf{X}) = \exists \mathbf{Y}\, \Phi(\mathbf{X}, \mathbf{Y})$, where $\Phi(\mathbf{X}, \mathbf{Y})$ is a conjunction of atoms (possibly equalities, but not inequalities) with the variables $\mathbf{X}$ and $\mathbf{Y}$, and possibly constants, but no nulls. A *Boolean CQ (BCQ)* over $\mathcal{R}$ is a CQ of the form $Q()$, often written as the set of all its atoms, without quantifiers. Answers to CQs and BCQs are defined via *homomorphisms*, which are mappings $\mu\colon \Delta \cup \Delta_N \cup \mathcal{V} \to \Delta \cup \Delta_N \cup \mathcal{V}$ such that (i) $c \in \Delta$ implies $\mu(c) = c$, (ii) $c \in \Delta_N$ implies $\mu(c) \in \Delta \cup \Delta_N$, and (iii) μ is naturally extended to atoms, sets of atoms, and conjunctions of atoms. The set of all *answers* to a CQ $Q(\mathbf{X}) = \exists \mathbf{Y}\, \Phi(\mathbf{X}, \mathbf{Y})$ over D, denoted $Q(D)$, is the set of all tuples $\mathbf{t}$ over Δ for which there exists a homomorphism $\mu\colon \mathbf{X} \cup \mathbf{Y} \to \Delta \cup \Delta_N$ such that $\mu(\Phi(\mathbf{X}, \mathbf{Y})) \subseteq D$ and $\mu(\mathbf{X}) = \mathbf{t}$. The *answer* to a BCQ $Q()$ over a database D is *Yes*, denoted $D \models Q$, iff $Q(D) \neq \emptyset$.

Given a relational schema $\mathcal{R}$, a *tuple-generating dependency (TGD)* σ is a first-order formula $\forall \mathbf{X} \forall \mathbf{Y}\, \Phi(\mathbf{X}, \mathbf{Y}) \to \exists \mathbf{Z}\, \Psi(\mathbf{X}, \mathbf{Z})$, where $\Phi(\mathbf{X}, \mathbf{Y})$ and $\Psi(\mathbf{X}, \mathbf{Z})$ are conjunctions of atoms over $\mathcal{R}$ (without nulls), called the *body* and the *head* of σ, denoted $\mathit{body}(\sigma)$ and $\mathit{head}(\sigma)$, respectively. Such σ is satisfied in a database D for $\mathcal{R}$ iff, whenever there exists a homomorphism h that maps the atoms of $\Phi(\mathbf{X}, \mathbf{Y})$ to atoms of D, there exists an extension h' of h that maps the atoms of $\Psi(\mathbf{X}, \mathbf{Z})$ to atoms of D. All sets of TGDs are finite here. Since TGDs can be reduced to TGDs with only single atoms in their heads, in the sequel, every TGD has w.l.o.g. a single atom in its head. A TGD σ is *guarded* iff it contains an atom in its body that contains all universally quantified variables of σ. The leftmost such atom is the *guard atom* (or *guard*) of σ.

Query answering under TGDs, i.e., the evaluation of CQs and BCQs on databases under a set of TGDs is defined as follows. For a database D for $\mathcal{R}$, and a set of TGDs Σ on $\mathcal{R}$, the set of *models* of D and Σ, denoted $mods(D, \Sigma)$, is the set of all (possibly infinite) databases B such that (i) $D \subseteq B$ and (ii) every $\sigma \in \Sigma$ is satisfied in B. The set of *answers* for a CQ Q to D and Σ, denoted $ans(Q, D, \Sigma)$, is the set of all tuples $\mathbf{a}$ such that $\mathbf{a} \in Q(B)$ for all $B \in mods(D, \Sigma)$. The *answer* for a BCQ Q to D and Σ is *Yes*, denoted $D \cup \Sigma \models Q$, iff $ans(Q, D, \Sigma) \neq \emptyset$. Note that query answering under general TGDs is undecidable [8], even when the schema and TGDs are fixed [9]. Decidability and tractability in the data complexity of query answering for the guarded case follows from a bounded tree-width property.

Negative constraints (or simply *constraints*) γ are first-order formulas of the form $\forall \mathbf{X} \Phi(\mathbf{X}) \rightarrow \bot$, where $\Phi(\mathbf{X})$ (called the *body* of γ) is a conjunction of atoms (without nulls). Under the standard semantics of query answering of BCQs in Datalog+/– with TGDs, adding negative constraints is computationally easy, as for each constraint $\forall \mathbf{X} \Phi(\mathbf{X}) \rightarrow \bot$, we only have to check that the BCQ $\Phi(\mathbf{X})$ evaluates to false in D under Σ; if one of these checks fails, then the answer to the original BCQ Q is true, otherwise the constraints can simply be ignored when answering the BCQ Q.

Equality-generating dependencies (or *EGDs*) σ, are first-order formulas $\forall \mathbf{X}\, \Phi(\mathbf{X}) \rightarrow X_i = X_j$, where $\Phi(\mathbf{X})$, called the *body* of σ, denoted $body(\sigma)$, is a (without nulls) conjunction of atoms, and X_i and X_j are variables from $\mathbf{X}$. Such σ is satisfied in a database D for $\mathcal{R}$ iff, whenever there is a homomorphism h such that $h(\Phi(\mathbf{X}, \mathbf{Y})) \subseteq D$, it holds that $h(X_i) = h(X_j)$. Adding EGDs over databases with TGDs along with negative constraints does not increase the complexity of BCQ query answering as long as they are *non-conflicting* [6]. Intuitively, this ensures that, if the chase (see below) fails (due to strong violations of EGDs), then it already fails on the database D, and if it does not fail, then whenever "new" atoms (from the logical point of view) are created in the chase by the application of the EGD chase rule, atoms that are logically equivalent to the new ones are guaranteed to be generated also in the absence of the EGDs, guaranteeing that EGDs do not influence the chase with respect to query answering.

We usually omit the universal quantifiers in TGDs, negative constraints, and EGDs, and we implicitly assume that all sets of dependencies and/or constraints are finite.

The Chase. The *chase* was first introduced to enable checking implication of dependencies, and later also for checking query containment. By "chase", we refer both to the chase procedure and to its output. The TGD chase works on a database via so-called TGD *chase rules* (see [6] for an extended chase with also EGD chase rules).

TGD Chase Rule. Let D be a database, and σ a TGD of the form $\Phi(\mathbf{X}, \mathbf{Y}) \rightarrow \exists \mathbf{Z}\, \Psi(\mathbf{X}, \mathbf{Z})$. Then, σ is *applicable* to D if there exists a homomorphism h that maps the atoms of $\Phi(\mathbf{X}, \mathbf{Y})$ to atoms of D. Let σ be applicable to D, and h_1 be a homomorphism that extends h as follows: for each $X_i \in \mathbf{X}$, $h_1(X_i) = h(X_i)$; for each $Z_j \in \mathbf{Z}$, $h_1(Z_j) = z_j$, where z_j is a "fresh" null, i.e., $z_j \in \Delta_N$, z_j does not occur in D, and z_j lexicographically follows all other nulls already introduced. The *application of* σ on D adds to D the atom $h_1(\Psi(\mathbf{X}, \mathbf{Z}))$ if not already in D.

The chase algorithm for a database D and a set of TGDs Σ consists of an exhaustive application of the TGD chase rule in a breadth-first (level-saturating) fashion, which outputs a (possibly infinite) chase for D and Σ. Formally, the *chase of level up to* 0 of D relative to Σ, denoted $\mathit{chase}^0(D, \Sigma)$, is defined as D, assigning to every atom in D the *(derivation) level* 0. For every $k \geqslant 1$, the *chase of level up to* k of D relative to Σ, denoted $\mathit{chase}^k(D, \Sigma)$, is constructed as follows: let $I_1, \ldots, I_n$ be all possible images of bodies of TGDs in Σ relative to some homomorphism such that (i) $I_1, \ldots, I_n \subseteq \mathit{chase}^{k-1}(D, \Sigma)$ and (ii) the highest level of an atom in every I_i is $k-1$; then, perform every corresponding TGD application on $\mathit{chase}^{k-1}(D, \Sigma)$, choosing the applied TGDs and homomorphisms in a (fixed) linear and lexicographic order, respectively, and assigning to every new atom the *(derivation) level* k. The *chase* of D relative to Σ, denoted $\mathit{chase}(D, \Sigma)$, is defined as the limit of $\mathit{chase}^k(D, \Sigma)$ for $k \to \infty$.

The (possibly infinite) chase relative to TGDs is a *universal model*, i.e., there exists a homomorphism from $\mathit{chase}(D, \Sigma)$ onto every $B \in \mathit{mods}(D, \Sigma)$ [6]. This implies that BCQs Q over D and Σ can be evaluated on the chase for D and Σ, i.e., $D \cup \Sigma \models Q$ is equivalent to $\mathit{chase}(D, \Sigma) \models Q$. For guarded TGDs Σ, such BCQs Q can be evaluated on an initial fragment of $\mathit{chase}(D, \Sigma)$ of constant depth $k \cdot |Q|$, which is possible in polynomial time in the data complexity.

Datalog+/– Ontologies. A *Datalog+/– ontology* $KB = (D, \Sigma)$, where $\Sigma = \Sigma_T \cup \Sigma_E \cup \Sigma_{NC}$, consists of a database D, a set of TGDs Σ_T, a set of non-conflicting EGDs Σ_E, and a set of negative constraints Σ_{NC}. We say KB is *guarded* (resp., *linear*) iff Σ_T is guarded (resp., linear). Example 1 illustrates a simple Datalog+/– ontology, which is used in the sequel as a running example.

Example 1. Consider the following simple ontology $O = (D, \Sigma)$, where:

$$\begin{aligned}\Sigma = \{&\mathit{room}(R, H) \rightarrow \mathit{accom}(R), \quad \mathit{bed}(B, H) \rightarrow \mathit{accom}(B),\\ &\mathit{apartment}(A) \rightarrow \mathit{accom}(A), \ \mathit{hotel}(H) \rightarrow \exists R\, \mathit{room}(R, H),\\ &\mathit{hostel}(H) \rightarrow \exists B\, \mathit{bed}(B, H), \mathit{bb}(B) \rightarrow \exists R\, \mathit{room}(R, B)\}\end{aligned}$$

and $D = \{\mathit{hotel}(h_1), \mathit{hotel}(h_2), \mathit{hostel}(hs_1), \mathit{bb}(bb_1), \mathit{apartment}(a), \mathit{bed}(b_1, hs_1), \mathit{bed}(b_2, hs_1), \mathit{room}(r_1, h_1), \mathit{room}(r_2, h_1), \mathit{room}(r_3, h_2), \mathit{room}(r_4, bb_1)\}$.

This ontology models a very simple accommodation booking domain, which could be used as the underlying model in an online system. Accommodations can be either rooms in hotels or bed and breakfasts, beds in hostels, or apartments. The database D provides some instances for each kind of accommodation. ■

3 PP-Datalog+/–: Syntax and Semantics

In this section, we introduce the PP-Datalog+/– language, an extension of Datalog+/– with both a preference model and a probabilistic model; this formalism is based on the one first presented in [10], which extends Datalog+/– with preferences (but not probabilities). To this end, we assume the following sets giving rise to the logical languages for ontologies, preferences, and probability models: Δ_{Ont}, Δ_{Pref}, and Δ_M are finite sets of constants, $\mathcal{R}_{\mathit{Ont}}$, $\mathcal{R}_{\mathit{Pref}}$, and $\mathcal{R}_M$ are finite sets of predicate names such that

$\mathcal{R}_M \cap \mathcal{R}_{Ont} = \emptyset$, and $\mathcal{V}_{Ont}$, $\mathcal{V}_{Pref}$, and $\mathcal{V}_M$ are infinite sets of variables. In the following, we assume w.l.o.g. that $\mathcal{R}_{Pref} \subseteq \mathcal{R}_{Ont}$, $\Delta_{Pref} \subseteq \Delta_{Ont}$, $\mathcal{V}_{Pref} \subseteq \mathcal{V}_{Ont}$. These sets give rise to corresponding *Herbrand bases* consisting of all possible ground atoms that can be formed, which we denote with $\mathcal{H}_{Ont}$, $\mathcal{H}_{Pref}$, and $\mathcal{H}_M$, respectively. Clearly, we have $\mathcal{H}_{Pref} \subseteq \mathcal{H}_{Ont}$, meaning that preference relations are defined over a subset of the possible ground atoms.

Preference Models. A *preference relation* is any binary relation $\succ \subseteq \mathcal{H}_{Pref} \times \mathcal{H}_{Pref}$. In this paper, we are interested in strict partial orders (SPOs), which are irreflexive and transitive relations—we consider these to be the minimal requirements for a preference relation to be useful in the applications that we envision. One way of specifying such a relation that is especially compatible with our approach is the preference formula framework of [11]. In this work, we assume the existence of a general *preference model* P specifying a preference relation over a subset of $\mathcal{H}_{Ont}$, denoted $\succ_P$; in general, we treat $\succ_P$ as a set of ordered pairs. When a preference relation $\succ$ is induced by an assignment of a numeric score to each element in such a way that $a_1 \succ a_2$ iff $\mathit{score}(a_1) > \mathit{score}(a_2)$, we say that $\succ$ is *score-based.*

Example 2. Following the topic of Example 1, a preference relation might be specified by a user over the *accom* atoms; an example of such a relation is shown in Fig. 3 (top left) (we assume the transitive closure of the graph depicted). For instance, these preferences reflect that the user prefers the room in the bed and breakfast above all else; next, the apartment and rooms r_1 and r_3 are all incomparable (perhaps the user cannot decide between the cost and locations of the apartment and hotels h_1 and h_2). Towards the least preferred options are the two beds in the hostel, and room r_2 in hotel h_1. ■

Probabilistic Models. For modeling uncertainty, we assume the existence of a probabilistic model M that represents a probability distribution $\Pr_M$ over some set $X = \{X_1, \ldots, X_n\}$ of Boolean variables such that there is a 1-to-1 mapping from X to the set of all ground atoms over $\mathcal{R}_M$ and Δ_M. Examples of the type of probabilistic models that we assume in this work are Markov logic and Bayesian networks. The probabilistic extension of Datalog+/– adopted here was first introduced in [12]; probabilistic query answering (without preferences) in a version of this model using Markov logic was also studied in [7].

Substitutions. A substitution is a mapping from variables to variables or constants. Two sets S and T *unify* via a substitution θ iff $\theta S = \theta T$, where θA denotes the application of θ to all variables in all elements of A (here, θ is a unifier). A *most general unifier* (mgu) is a unifier θ such that for all other unifiers ω, there is a substitution σ such that $\omega = \sigma \circ \theta$.

Definition 1. Let M be a probabilistic model. Then, a *(probabilistic) annotation* λ relative to M is a (finite) set of expressions of the form $\langle A_i = x_i \rangle$, where: (i) A_i is an atom over $\mathcal{R}_M$, $\mathcal{V}_M$, and Δ_M; and (ii) $x_i \in \{0, 1\}$. A probabilistic annotation is *valid* iff for any two different expressions $\langle A = x \rangle, \langle B = y \rangle \in \lambda$, there does not exist a substitution that unifies A and B.

Intuitively, a probabilistic annotation is used to describe the class of events in which the random variables in a probabilistic model M are compatible with the settings of the

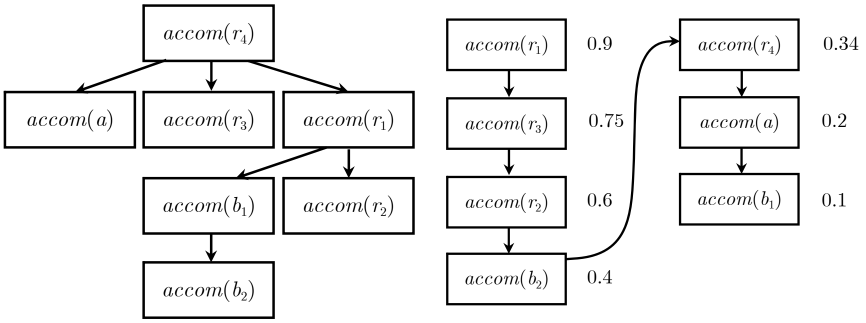


Fig. 1. User preferences for Example 1 (left), and those induced by probability values (right)

random variables described by λ, i.e., each X_i has the value x_i. A *probabilistic scenario* is a valid probabilistic annotation λ for which $|\lambda| = |X|$ and all $\langle A = x_i \rangle \in \lambda$ are such that A is ground. We use $scn(M)$ to denote the set of scenarios in M.

Example 3. Continuing with the running example, suppose the online booking system consults a probabilistic model that assigns probabilities specifying how likely it is that certain events happen. In this case, suppose that the system in question is aggregating information from multiple sources, which may contain conflicting information, as well as uncertainty due to other factors such as language. Thus, it would be useful to inform the user of the probability of accommodations being available for the dates being searched; the following is an example of how the ontology from Example 1 may be extended by replacing the atoms in the database with formulas of the form: $room(R, H) : \{available(R, H, D) = 1\}$, where $available(R, H, D)$ denotes the probabilistic event that the room R in the hotel (or hostel) H is available on the date D (which is instantiated through the query or by adding the information to the probabilistic model with the date specified by the user). Fig. 1 (right) gives an example of such a probability assignment (derived as explained in the semantics section below), along with the preference relation in graph form that is induced by these values assuming that higher probabilities are more preferable. ∎

Preference Combination Operators. As seen in Fig. 1, the particular challenge encountered in PP-Datalog+/– ontologies is that the preference model yields a certain precedence relation that might be in disagreement with the one induced by the probabilistic model. To address this situation, we make use of *preference combination operators*, which take two preference relations and produce a third one satisfying two basic properties.

Definition 2. Let $\succ_P$ be an SPO and $\succ_M$ be a score-based preference relation. A *preference combination operator* $\otimes(\succ_M, \succ_P)$ yields a relation $\succ^*$ such that: (i) $\succ^*$ is an SPO, and (ii) if $a_1 \succ_P a_2$ and $a_1 \succ_M a_2$, then $a_1 \succ^* a_2$.

The two properties required by Definition 2 are the minimal required to produce a "reasonable" combination of the two relations; as we show in Section 4 below, particular implementations may satisfy further desirable properties.

We are now ready to define PP-Datalog+/– ontologies.

Definition 3. A *PP-Datalog+/– ontology* (or *PP-KB*) is of the form $KB = (O, P, M, \otimes)$, where O is a Datalog+/– ontology, P is a preference model, M is a probabilistic model with Herbrand bases $\mathcal{H}_{Ont}$, $\mathcal{H}_{Pref}$, and $\mathcal{H}_M$, respectively, such that $\mathcal{H}_{Pref} \subseteq \mathcal{H}_{Ont}$, and $\otimes$ is a preference combination operator. If O is a guarded Datalog+/– ontology, then KB is a *guarded* PP-Datalog+/– ontology.

In the following, we discuss the semantics of PP-Datalog+/– ontologies, and formally define the kinds of queries that we address in this paper.

Semantics. Let $KB = (O, P, M, \otimes)$ be PP-Datalog+/– ontology; the semantics of PP-Datalog+/– arises as a direct combination of the semantics of Datalog+/– and that of the preference and probabilistic models. Relative to the probabilistic model, we have that $\Pr_{KB}(a) = p$ iff $p = \sum_{\lambda \in scn(M),\, O_\lambda \models a} \Pr_M(\lambda)$. We refer to the score-based preference relation induced by $\Pr_M$ as the *probabilistic preference relation* associated with KB, and denote it with $\succ_M$. Let $\succ^* = \otimes(\succ_M, \succ_P)$; we say that $KB \models a_1 \succ^* a_2$ iff (i) $O \models a_1$ and $O \models a_2$; and (ii) $a_1 \succ^* a_2$. Intuitively, the consequences of a knowledge base $KB = (O, P, M, \otimes)$ are computed in terms of the classical consequences of the Datalog+/– ontology O, and the preference combination operator yields a preference relation over pairs of atoms in $\mathcal{H}_{Ont}$.

Skyline and k-Rank Queries. In this paper, we are interested in skyline queries [13], a well-known class of queries that can be issued over preference-based formalisms, and the iterated computation of skyline answers that allows us to assign a *rank* to every atom. In the following, we focus on a specific kind of classical queries, called disjunctive atomic queries (DAQs), which are disjunctions of atoms.

Definition 4. Let $KB = (O, P, M, \otimes)$ be a PP-Datalog+/– ontology where $\mathcal{H}_{Ont}$ is the Herbrand base of O, $\succ^* = \otimes\,(\succ_M, \succ_P)$, $k \geqslant 0$, and $Q(\mathbf{X}) = q_1(\mathbf{X}_1) \vee \cdots \vee q_n(\mathbf{X}_n)$ be a DAQ. Then, a *skyline answer* to Q relative to $\succ^*$ is any θq_i entailed by O such that no θ' exists with $O \models \theta' q_j$ and $\theta' q_j \succ^* \theta q_i$, where θ and θ' are ground substitutions for the variables in $Q(\mathbf{X})$.

To obtain an ordered list of answers to queries under transitive relations, we adopt the *iterated skyline* approach proposed in [11], which simply assigns a *rank* to each answer by iterating the following procedure: (1) assign the rank k to the skyline answers; and (2) remove from consideration all answers of rank k, increment k by one, and go back to (1). As a generalization of classical top-k queries, we adopt here *k-rank queries*; answers to such queries are simply k-tuples of answers sorted by rank. In the following, we use $rank(a, \succ_M)$ to denote the rank assigned to atom a by relation $\succ_M$.

Intuitively, for DAQs, both kinds of answers can be seen as atomic consequences of O that satisfy the query: the skyline answers can be seen as sets of atoms that are not dominated by any other such atom, while k-rank answers are k-tuples sorted according to the preference relation. We refer to these as answers in *atom form*.

In the next section, we present algorithms for implementing two particular preference combination operators that, as we show through a series of properties, produce a new preference relation $\succ^*$ that is useful in answering k-rank queries that adequately

```
Algorithm 1: CombinePrefs($\succ_M, \succ_P, t$)
Input: SPO $\succ_P$, score-based preference relation $\succ_M$ over $\mathcal{H}_{Ont}$, and $t \in [0, 1]$.
Output: Preference relation $\succ_t^* \subseteq \mathcal{H}_{Ont} \times \mathcal{H}_{Ont}$.

 1. Initialize $\succ_t^*$ as a copy of $\succ_P$;
 2. For every pair $(a_i, a_j) \in \succ_P$ do begin
 3.     if $score(a_j) - score(a_i) > t$ and changing $(a_i, a_j)$ to $(a_j, a_i)$ in $\succ_t^*$
              does not introduce a cycle in its associated graph then
 4.         change $(a_i, a_j)$ to $(a_j, a_i)$ in $\succ_t^*$;
 5. End;
 6. Return $transitiveClosure(\succ_t^*)$.
```

Fig. 2. An algorithm for combining an arbitrary SPO with a weak order in the form of a score-based preference relation.

reflect both the initial preferences expressed by the user as well as the fact that higher probability answers are more desirable.

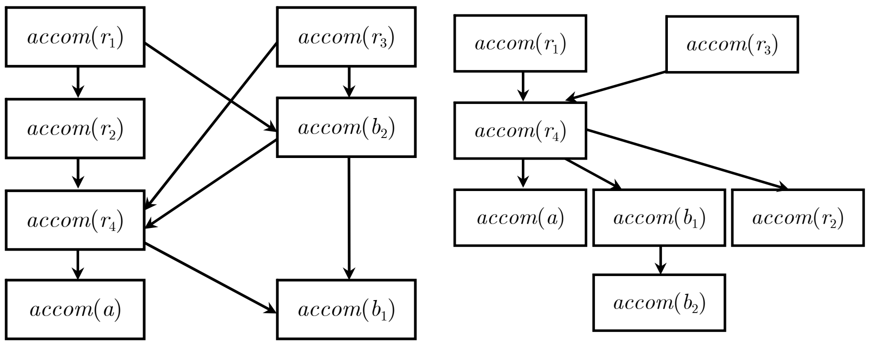
4 Two Preference Combination Operators

In this section, we study two particular instantiations of a preference combination operator, and study their semantic properties. Later, in Section 5, we show an algorithm that uses these operators for answering k-rank queries to PP-Datalog+/– ontologies in polynomial time in the data complexity (modulo the cost of computing probabilities with respect to the model M).

4.1 A General Preference Combination Operator

Algorithm CombinePrefs in Fig. 2 implements a preference combination operator using a value $t \in [0, 1]$ that allows the user to choose how much influence the probabilistic model has on the output preference relation; we use $\succ_t^*$ to denote the output relation. The algorithm iterates through all pairs of elements in $\succ_P$ and, if (i) $\succ_M$ disagrees with $\succ_P$, (ii) the difference in score is greater than t, and (iii) inserting the pair according to $\succ_M$ doesn't produce a cycle, then the pair is inserted in reverse order into the output; otherwise, the output contains the same pair as $\succ_P$. Finally, the algorithm outputs the transitive closure of this relation. Note that if $t = 0$, then the probability-induced relation has precedence; at the other end of the spectrum, if $t = 1$, we have that CombinePrefs($\succ_M, \succ_P, t$) $= \succ_P$ (these properties are presented formally in Theorem 4). The following is an example of how the algorithm works.

Example 4. Let us return to the running example; Fig. 3 (top left) shows the result of running the CombinePrefs algorithm over the two preference relations and $t = 0$, while Fig. 3 (top right) corresponds to $t = 0.3$—note that, for clarity, the transitive closures are not shown in these figures. Fig. 3 (bottom) shows the result of computing the rank via the iterative computation of the skyline answers for the three preference relations. As we can see, the user's original preferences ($\succ_P$) are preserved to a greater



Rank	$\succ_P$	$\succ_0^*$	$\succ_{0.3}^*$
1	$[X/r_4]$	$[X/r_1]$, $[X/r_3]$	$[X/r_1]$, $[X/r_3]$
2	$[X/a]$, $[X/r_3]$, $[X/r_2]$	$[X/r_2]$, $[X/b_2]$	$[X/r_4]$
3	$[X/b_1]$,$[X/r_2]$	$[X/r_4]$	$[X/a]$, $[X/b_1]$, $[X/r_2]$
4	$[X/b_2]$	$[X/a]$, $[X/b_1]$	$[X/b_2]$

Fig. 3. Combinations of user preferences and probability induced preferences with the CombinePrefs algorithm (transitive closure not shown)—top left: $t = 0$; top right: $t = 0.3$. Bottom: answers to query *accom*(X) according to their rank relative to the original preference relation (Fig. 1, left) and the two combinations.

extent in $\succ_{0.3}^*$ than in $\succ_0^*$: the answers in the former have a difference in rank of at most 1, whereas those in the latter have a difference of 2 (out of a total of 4) in multiple cases. This reflects the greater influence that $\succ_P$ has in the result for $t = 0.3$ as compared to $t = 0$. ■

Theorem 1. *Let $\succ_P$ be an SPO and $\succ_M$ be a score-based preference relation, and let S be the time required to compute the score of any atom with respect to $\succ_M$. Algorithm* CombinePrefs *runs in time $O(poly(|\succ_P|) \cdot S)$.*

Proof sketch. The `for` loop in line 2 of CombinePrefs, which involves the creation of the output relation except for the transitive closure, is carried out $|\succ_P|$ times; in each iteration, the score of two atoms is computed. Each iteration of the loop can be done in time $O(poly(|\succ_P|) \cdot S)$ and, since $\prec^*$ right before executing line 7 has the same size as $\succ_P$, the transitive closure can also be computed in time $O(poly(|\succ_P|)$. □

For PP-Datalog+/– ontologies, the cost S in Theorem 1 refers to the cost of computing probabilities relative to model M; therefore, this cost could range from polynomial time (such as in approximation algorithms, or tractable models like polytrees [14] or tractable Markov logic [15]) to #P-complete (such as in general Markov logic [16] or Bayesian networks). In particular, since $\succ_P$ describes a preference relation over the consequences of an ontology, we have the following corollary.

Corollary 1. *Let $KB = (O, P, M,$ CombinePrefs$)$ be a guarded PP-Datalog+/– ontology such that $\mathcal{H}_{Ont}$ is the Herbrand base for O, $\succ_P$ be an SPO, and $\succ_M$ be a score-based preference relation defined over $\mathcal{H}_{Ont}$, and Q be a conjunctive query. Then, if*

$\mathrm{Pr}_M(a)$ *can be computed or approximated in polynomial time in the data complexity for any a such that KB* $\models a$*, Algorithm* CombinePrefs *runs in polynomial time in the data complexity.*

Proof sketch. For guarded Datalog+/– ontologies, the necessary finite part of the chase required to answer a query is of polynomial size in the data complexity [6], and it is precisely this structure that gives rise to $\succ_P$. By the hypothesis that probabilities are computed in polynomial time and the result in Theorem 1, the statement follows. □

In the following theorem, we show that CombinePrefs satisfies the definition of a preference combination operator (cf. Definition 2).

Theorem 2. *Let* $\succ_P$ *be an SPO,* $\succ_M$ *be a score-based preference relation,* $t \in [0,1]$*, and* $\succ_t^* =$ CombinePrefs$(\succ_M, \succ_P, t)$*. Then, for any value of t, we have:*

(i) $\succ_t^*$ *is an SPO; and*

(ii) If $a_1 \succ_P a_2$ *and* $a_1 \succ_M a_2$*, then* $a_1 \succ_t^* a_2$*.*

Proof sketch. (i) We must show that $\succ^*$ is antisymmetric, irreflexive, and transitive. Let $\succ'$ be the intermediate result of the CombinePrefs operator just before computing the transitive closure. By hypothesis, $\succ_U$ is irreflexive and, since $\succ'$ cannot contain any new self-edges, this property is preserved in $\succ'$. Similarly, since $\succ_U$ is antisymmetric and CombinePrefs checks for cycles before changing edges, there will never be two edges (a,b) and (b,a)—antisymmetry therefore vacuously holds. Finally, by construction, $\succ^*$ is the transitive closure of $\succ'$; since this operation cannot add cycles, we conclude that $\succ^*$ is an SPO.

(ii) A necessary condition for CombinePrefs to change the order of a pair in $\succ_U$ is that the pair in $\succ_M$ be reversed. Since by hypothesis this is not the case, the statement follows. □

We now study the properties enjoyed by the output of the CombinePrefs algorithm. The following theorem states that the output for $t = 0$ is invariant to changes in the scores assigned as long as the order is the same.

Theorem 3. *Let* $\succ_P$ *be an SPO, and* $\succ_M$ *and* $\succ_{M'}$ *be score-based preference relations. If* $\succ_M = \succ_{M'}$*, then* CombinePrefs$(\succ_M, \succ_P, 0) =$ CombinePrefs$(\succ_{M'}, \succ_P, 0)$.

Proof sketch. If $t = 0$, the algorithm checks (in line 3) if pairs (a_i, a_j) in $\succ_P$ are such that $\mathrm{Pr}(a_j) > \mathrm{Pr}(a_i)$. Since by hypothesis, we know that $\succ_M = \succ_{M'}$, it must be the case that $\mathrm{Pr}(a_j) > \mathrm{Pr}(a_i)$ relative to $\succ_M$ iff this is the case relative to $\succ_{M'}$, and thus the outputs in both cases must be identical. □

This is an important property, since it implies that approximation algorithms can be used to compute the probability values that induce the $\succ_M$ relation—as long as the relative order of the atoms is guaranteed to be correct, there is no need to compute the exact values when $t = 0$.

The following theorem shows the behavior of the operator for the extreme values of parameter t.

Theorem 4. *Let $\succ_P$ be an SPO, and $\succ_M$ be a score-based preference relation:*

(i) rank$(a, \mathsf{CombinePrefs}(\succ_M, \succ_P, 0)) =$ *rank*$(a, \succ_M)$*, for any atom a.*

(ii) $\mathsf{CombinePrefs}(\succ_M, \succ_P, 1) = \succ_P$.

Proof sketch. (i) When $t = 0$, the output relation is the result of comparing all pairs in $\succ_P$ and replacing them with the order imposed on their elements by $\succ_M$. Thus, given that $\succ_P$ is transitive, the sequence of atoms giving rise to the rank of each atom is reordered in the resulting relation according to the rank of each element in $\succ_M$, and the statement follows.

(ii) Direct consequence of the condition in line 3 of the algorithm: since the difference in probabilities can never be greater than 1, the output relation is a copy of $\succ_P$. □

Finally, the following theorem studies a property that is analogous to the "Sensitivity postulates" presented in [17], [18, pp. 12–14,63–64]. Note, however, that these postulates were designed to combine two score-based preference relations, whereas our setting is more general—the "Faithfulness" postulate in [18] is already guaranteed by our definition of preference combination operator.

Theorem 5. *Let $KB = (O, P, M, \mathsf{CombinePrefs})$ be a PP-Datalog+/– ontology where $\mathcal{H}_{Ont}$ is the Herbrand base of O, Q be a DAQ, $k \geqslant 0$, and $\succ^*_t = \mathsf{CombinePrefs}(\succ_M, \succ_P, t)$, with $t \in [0,1]$. For every atom a that does not belong to any k-rank answer to Q over KB relative to $\succ^*_t$, there exists a probabilistic model M' and $t' \in [0,1]$ such that a belongs to some k-rank answer to Q over $KB' = (O, P, M', \mathsf{CombinePrefs})$ relative to $\succ'_{t'} = \mathsf{CombinePrefs}(\succ_{M'}, \succ_P, t')$.*

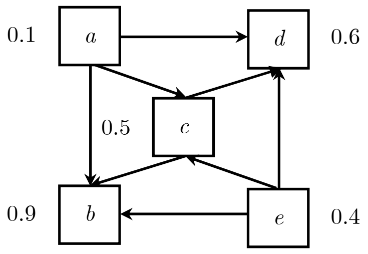
Proof sketch. By hypothesis, there exist atoms $\{a_1, ..., a_n\}$ such that $a_i \succ^*_t a$. Now, let $t' = 0$ and set the probabilities in M' so that $\Pr_{M'}(a) > \Pr_{M'}(a_i)$, for $1 \leqslant i \leqslant n$. By construction, we now have that $a \succ'_{t'} a_i$, and the statement follows. □

In the next section, we study a different kind of combination operator based on a tradeoff between generality and the properties that can be proved about the result.

4.2 An Operator Based on Rank

The following example shows a shortcoming of the CombinePrefs operator in that it fails to exhibit a kind of monotonicity that might be expected when the two input relations disagree on a given pair (a_1, a_2). In particular, if the operator chooses the order imposed by $\succ_M$, then it may not continue to do so for smaller values of t. On the other hand, if the operator chooses the order imposed by $\succ_P$, then it may also fail to do so for greater values of t.

Example 5. Consider the preference relation in Fig. 4 (left), where the directed edges between nodes describe a relation $\succ_P$ and the numbers beside each node define score-based relation $\succ_M$. The table on the right specifies what happens when the two are input to the CombinePrefs algorithm and the edges are inspected in the specified order, for two values of t. Note row 4, where edge (a, c) becomes (c, a) under $t = 0.1$ but stays unaltered under $t = 0.05$, and row 6, where edge (a, b) stays unaltered under $t = 0.05$ but becomes (b, a) under $t = 0.1$. ■



Order	Edge	$t = 0.1$	$t = 0.05$
1	(c, d)	No (t)	Yes
2	(d, e)	No (cycle)	Yes
3	(c, e)	No (t)	Yes
4	(a, c)	Yes	No (cycle)
5	(a, d)	Yes	Yes
6	(a, b)	Yes	No (cycle)
7	(c, b)	Yes	No (cycle)
8	(e, b)	Yes	No (cycle)

Fig. 4. User preferences for Example 5 (left), and the results of applying Algorithm CombinePrefs for two different values of t (right). In the table, "yes" and "no" means whether or not the edge is reversed in the result—in the negative cases, the reason is given in parenthesis.

```
Algorithm 2: CombinePrefsRank($\succ_M, \succ_P, f$)
Input: SPO $\succ_P$, score-based preference relation $\succ_M$ over $\mathcal{H}_{Ont}$,
       and integer binary function $f$ such that $f(x, y) \in [x, y]$.
Output: Score-based preference relation $\succ_f^{rank} \subseteq \mathcal{H}_{Ont} \times \mathcal{H}_{Ont}$.

1. Initialize $\succ_f^{rank}$ as an empty preference relation over $\subseteq \mathcal{H}_{Ont} \times \mathcal{H}_{Ont}$;
2. Initialize ranks as an empty mapping from preference relations to integers;
3. Compute the ranks of each atom $a$ relative to $\succ_P$ and $\succ_M$, and store them in ranks;
4. For every atom $a$ in $dom(ranks)$ do
5.     add $a$ to $\succ_f^{rank}$ with score $f(ranks(a, \succ_P), ranks(a, \succ_M))$;
6. Return $\succ_f^{rank}$.
```

Fig. 5. Algorithm for combining an SPO with a score-based preference relation, based on rank.

As shown in Example 5, the cause of this kind of unpredictability is the potential presence of cycles when merging preference relations. Algorithm 2 (Fig. 5) avoids this issue by combining the two relations into a score-based one based on the rank of each atom relative to each input relation. The algorithm takes as input an integer binary function that combines the two ranks and assigns it to the atom in the output relation—this function can be as simple as `min`, `max`, or `avg`, or a more complex function that takes into account how much the user would like to base the new rank on one relation or the other. A detailed treatment of such functions is outside the scope of this paper.

The following theorem states that CombinePrefsRank satisfies the conditions for being a preference combination operator for certain functions.

Theorem 6. *Let $\succ_P$ be an SPO, $\succ_M$ be a score-based preference relation, f be an integer binary function such that $f(x, y) \in [x, y]$, and $\succ_f^{rank}=$ CombinePrefsRank($\succ_M$, $\succ_P, f$). Then, we have:*

(i) $\succ_f^{rank}$ is an SPO; and

(ii) If $a_1 \succ_P a_2$, $a_1 \succ_M a_2$, and $f \in \{min, max, avg\}$, then $a_1 \succ_f^{rank} a_2$.

Proof sketch. (i) Direct consequence, since $\succ_f^{rank}$ is a score-based preference relation.

(ii) Let $r_1^P = \mathit{rank}(a_1, \succ_P)$, $r_2^P = \mathit{rank}(a_1, \succ_P)$, $r_1^M = \mathit{rank}(a_1, \succ_M)$, and $r_2^M = \mathit{rank}(a_1, \succ_M)$. The hypotheses imply that $r_1^P < r_2^P$ and $r_1^M < r_2^M$. Clearly, $f(r_1^P, r_1^M) < f(r_2^P, r_2^M)$ for $f \in \{\mathit{min}, \mathit{max}, \mathit{avg}\}$, and thus $a_1 \succ_f^{\mathit{rank}} a_2$. □

Analyzing the proof of Theorem 6, we can see that the result also holds for variations of the functions considered—as long as the condition $f(r_1^P, r_1^M) < f(r_2^P, r_2^M)$ is guaranteed, the result holds.

Another property of Algorithm CombinePrefsRank that can be shown is the analogous of Theorem 3; since the hypothesis implies that ranks relative to $\succ_M$ and $\succ_{M'}$ are equal, the result clearly holds. Finally, the following theorem discusses properties related to the postulates from [18] for this algorithm:

Theorem 7. *Let $KB = (O, P, M, \mathsf{CombinePrefs})$ be a PP-Datalog+/– ontology where $\mathcal{H}_{\mathit{Ont}}$ is the Herbrand base of O, Q be a DAQ, $k \geqslant 0$, and $\succ_t^* = \mathsf{CombinePrefs}(\succ_M, \succ_P, t)$, with $t \in [0,1]$. Then:*

(i) Let $f \in \{\mathit{min}, \mathit{max}, \mathit{avg}\}$, M' be a probabilistic model such that for some ground atom a we have $\Pr_{KB'}(a) \leqslant \Pr_{KB}(a)$ and $\Pr_{KB'}(a') = \Pr_{KB}(a')$ for every ground atom $a' \neq a$; let $KB' = (O, P, M', \mathsf{CombinePrefsRank})$. If a does not belong to any k-rank answer to Q over KB relative to $\succ_t^*$, then a does not belong to any k-rank answer to Q over KB' relative to $\succ_t' = \mathsf{CombinePrefsRank}(\succ_{M'}, \succ_P, t)$.

(ii) Given the setup in part (i), if a belongs to some k-rank answer to Q over KB relative to $\succ_t^*$ and $\Pr_{KB'}(a) \geqslant \Pr_{KB}(a)$, then a also belongs to some k-rank answer to Q over KB' relative to $\succ_t' = \mathsf{CombinePrefsRank}(\succ_{M'}, \succ_P, t)$.

(iii) For every atom a that does not belong to any k-rank answer to Q over KB relative to $\succ_t^*$, there exists a probabilistic model M' and $t' \in [0,1]$ such that a belongs to some k-rank answer to Q over $KB' = (O, P, M', \mathsf{CombinePrefs})$ relative to $\succ_{t'}' = \mathsf{CombinePrefsRank}(\succ_{M'}, \succ_P, t')$.

Proof sketch. (i) $\Pr_{KB'}(a) \leqslant \Pr_{KB}(a)$ implies that $\mathit{rank}(a, \succ_{M'}) \leqslant \mathit{rank}(a, \succ_M)$, and the result from Theorem 6 can be used to show the result.

(ii) Analogous to (i): $\Pr_{KB'}(a) \geqslant \Pr_{KB}(a)$ implies that $\mathit{rank}(a, \succ_{M'}) \geqslant \mathit{rank}(a, \succ_M)$.

(iii) Using $f = \mathit{min}$, the rest of the proof is analogous to that of Theorem 5. □

We conclude this section with an analysis of the running time of Algorithm CombinePrefsRank.

Theorem 8. *Let $\succ_P$ be an SPO and $\succ_M$ be a score-based preference relation, and let S be the time required to compute the score of any atom with respect to $\succ_M$. Algorithm CombinePrefsRank runs in time $O(\mathit{poly}(|\succ_P|) \cdot S)$.*

Proof sketch. Ranks for each atom relative to $\succ_P$ can be computed in polynomial time in the data complexity by a linear scan of each relation; for $\succ_M$, the cost is $O(\mathit{poly}(|\succ_P|) \cdot S)$. Using these values, the output relation can also be built within this time bound. □

5 Answering k-Rank Queries

As discussed above, in this paper, we are interested in computing the rank of the answers to queries by means of the iterated computation of its skyline answers [13].

```
Algorithm 3: $k$-Rank$(KB = (O, P, M, \otimes), Q, k)$
Input: Guarded PP-Datalog+/– ontology $KB$, DAQ $Q(\mathbf{X})$, and $k \geqslant 0$.
Output: $k$-rank answer $\langle a_1, ..., a_{k'} \rangle$ to $Q$, with $k' \leqslant k$.
 1. Initialize $Res$ as an empty vector of ground atoms;
 2. Set $\succ^* := \otimes(\succ_M, \succ_P)$;
 3. $C := computeChase(O, Q)$;
 4. $i := k$;
 5. While $i > 0$ do begin
 6.     $S := computeSkyline(C, Q, \succ^*)$;
 7.     Append $S$ to $Res$ (in arbitrary order);
 8.     Remove $S$ from $C$;
 9.     $i := i - |S|$;
10. End;
11. Return $truncate(Res, k)$.
```

Fig. 6. An algorithm for computing a k-rank answer to DAQ Q relative to the composition of $\succ_P$ and $\succ_M$

We now present a general algorithm to do so, and then analyze its correctness as well as its running time when used in conjunction with either the CombinePrefs or CombinePrefsRank algorithms.

The k-Rank Algorithm. The algorithm in Fig. 6 begins by computing the combination of the two preference relations in the PP-Datalog+/– ontology and the necessary finite part of the chase relative to Q. The main `while` loop iterates through the process of computing the skyline answers to Q relative to this new relation by using a *computeSkyline* subroutine (which can be implemented by means of a linear-time scan of C), updating the result by appending these answers in arbitrary order, and removing the atoms in the result from C. Once the loop is finished, the algorithm returns the first k results, since the last iteration might add superfluous elements.

Example 6. Consider the running example, with $Q = accom(X)$, $k = 5$, and $t = 0.3$. Fig. 3 (top right) depicts the result of the CombinePrefs algorithm, and Fig. 3 (bottom) (last column) shows the ranks associated with the different answers to Q. One possible k-rank answer to Q (in atom form) as computed by the algorithm is thus: $\langle accom(r_1)$, $accom(r_3)$, $accom(r_4)$, $accom(a)$, $accom(b_1)\rangle$. ■

The following theorem proves the correctness of the k-Rank algorithm, and shows that it runs in polynomial time under certain conditions.

Theorem 9. *Let $KB = (O, P, M, \otimes)$, with $O = (D, \Sigma)$, be a PP-Datalog+/– ontology, Q be a DAQ, and $k \geqslant 0$. Then,*

(i) Algorithm k-Rank correctly computes a k-rank answer to Q over KB; and

(ii) if O is a guarded Datalog+/– ontology and $\otimes$ can be computed in $O(poly(D) \cdot S)$, the running time of k-Rank is $O(poly(D) \cdot S)$, where S is the cost of computing $\Pr_M(a)$ for any atom a such that $O \models a$.

Proof sketch. (i) Correctness is a consequence of the direct application of the definition of k-rank: the `while` loop in line 4 iteratively computes the skyline answers to Q by means of a subroutine, adds these results to the output in arbitrary order, and removes them from consideration. Line 10 ensures that at most k results are returned.

(ii) Since O is guarded, answers to Q can be computed in polynomial time in the data complexity [6]. The $\otimes$ operator is computed in time $O(|\succ_P| \cdot S)$, and *computeSkyline* can be computed in polynomial time by a linear-time scan of the chase structure for Q (of polynomial size by hypothesis), and the results can be removed by another scan. $\Box$

As a consequence of Theorem 9 and Corollary 1, we can conclude that if probabilities in M can be computed or approximated in polynomial time (in the data complexity), then so can k-rank answers.

6 Related Work

The study of preferences has received much attention in many areas of study such as philosophy, choice theory, and certain areas of the social sciences (such as social choice). In computer science, the most relevant to our work is their incorporation into query answering mechanisms. To our knowledge, the current state of the art in this respect is centered around relational databases, and no other work to date combines general preferences with those induced by probability assignments.

The seminal work in preference-based query answering is that of [19], in which the SQL language is extended to incorporate user preferences, showing that the resulting formalism could be translated into the domain relational calculus. In [11], preference formulas are introduced as a logical formalism that allows an embedding of preference specifications into SQL through a *winnow* operator parameterized by a preference formula; the winnow operator is a generalization of the *skyline* operator, first introduced in [13]. Perhaps closest to our approach (except for the probabilistic model) is that of preference Datalog programs [20], which are a restriction of preference logic programs [21] that contain no uninterpreted function symbols and extend classical Datalog with constructs for determining which predicates must be optimized along with the optimization criteria (i.e., the set of preferences). For a recent survey on preference-based query answering, see [22].

With respect to probabilistic preferences, there are several works that have been developed in the last few years. Probabilistic skylines were introduced in [23] (and later also studied in [24]; the related *stochastic* skyline is introduced in [25]), where the authors tackle the problem of computing the probability of an element belonging to the skyline, rather than using the probability values for the purpose of ranking, as proposed in our work. In [17, 26], the authors focus on a more specific version of our problem, since they assume that tuples receive both a score and a probability value, and thus the two preference relations in question are score-based; furthermore, the approach adopted in their work is based on possible worlds, whereas, here, we focus on the use of probabilities solely as vehicles for ranking.

Preference revision is also closely related to our approach. The work of [27] tackles the problem of modeling preference change, studying axioms and postulates for the revision and contraction of a set of sentences specifying preferences by an input preference, in the style of belief revision theory; addition and subtraction of elements is also studied. The main difference with our work is that the author does not study algorithms nor complexity of deriving a revised relation, but rather focuses on a set of postulates based on those from belief revision and centered on obtaining a relation that incorporates the new preference with minimal change. The work of [28] is also related in that it addresses the problem of *query modification* in preference-based query answering in relational DBs. The focus is, however, on three specific combination mechanisms: union, prioritized, and Pareto composition, and on the study of the preservation of properties of different kinds of relations under these combinations.

Finally, social choice theory [29] is also relevant, since it seeks to combine preferences to produce a new preference relation; methods range from those using score-based relations (e.g., approval voting) to ones using more general ones (e.g., ranked pairs). In particular, the work of [30] studies possibility/impossibility results generalizing properties such as Arrow's theorem to the case in which incomparable elements exist. The study of potential applications of voting methods to query answering, especially in relation to the Social Web, is the topic of ongoing work.

7 Summary and Outlook

In this work, we have presented an extension of the Datalog+/– family of ontology languages for preference-based query answering under uncertainty. This task has recently attained central importance due to its relation with the Social (Semantic) Web. The main focus of this work has been on defining preference combination operators, which produce a preference relation, given a general SPO and a score-based preference relation. We have proposed two specific algorithms for such an operator, and analyzed their semantic and computational properties. Finally, we have studied a basic algorithm for answering k-rank queries, and showed that under certain conditions, k-rank queries can be answered in polynomial time in the data complexity, which is the same complexity as answering traditional preference-based queries in relational DBs. Current and future work involves implementing and testing the PP-Datalog+/– framework, developing other combination operators, and studying specific preference specification mechanisms and probabilistic models.

Acknowledgments. This work was supported by the UK EPSRC grant EP/J008346/1 ("PrOQAW"), ERC grant 246858 ("DIADEM"), and a Yahoo! Research Fellowship.

References

1. Berners-Lee, T., Hendler, J., Lassila, O.: The Semantic Web. Sci. Am. 284(5), 34–43 (2002)
2. Jung, J.C., Lutz, C.: Ontology-based access to probabilistic data with OWL QL. In: Cudré-Mauroux, P., et al. (eds.) ISWC 2012, Part I. LNCS, vol. 7649, pp. 182–197. Springer, Heidelberg (2012)
3. Lukasiewicz, T., Martinez, M.V., Orsi, G., Simari, G.I.: Heuristic ranking in tightly coupled probabilistic description logics. In: Proc. of UAI, AUAI, pp. 554–563 (2012)

4. Noessner, J., Niepert, M.: ELOG: A probabilistic reasoner for OWL EL. In: Rudolph, S., Gutierrez, C. (eds.) RR 2011. LNCS, vol. 6902, pp. 281–286. Springer, Heidelberg (2011)
5. Finger, M., Wassermann, R., Cozman, F.G.: Satisfiability in EL with sets of probabilistic ABoxes. In: Proc. of DL (2011)
6. Calì, A., Gottlob, G., Lukasiewicz, T.: A general Datalog-based framework for tractable query answering over ontologies. J. Web Sem. 14, 57–83 (2012)
7. Gottlob, G., Lukasiewicz, T., Martinez, M.V., Simari, G.I.: Query answering under probabilistic uncertainty in Datalog+/– ontologies. AMAI, 1–36 (2013)
8. Beeri, C., Vardi, M.Y.: The implication problem for data dependencies. In: Proc. of ICALP, pp. 73–85 (1981)
9. Calì, A., Gottlob, G., Kifer, M.: Taming the infinite chase: Query answering under expressive relational constraints. In: Proc. of KR, pp. 70–80. AAAI Press (2008)
10. Lukasiewicz, T., Martinez, M.V., Simari, G.I.: Preference-based query answering in Datalog+/– ontologies. In: Proc. of IJCAI (to appear, 2013)
11. Chomicki, J.: Preference formulas in relational queries. TODS 28(4), 427–466 (2003)
12. Lukasiewicz, T., Martinez, M.V., Simari, G.I.: Consistent answers in probabilistic Datalog+/– ontologies. In: Krötzsch, M., Straccia, U. (eds.) RR 2012. LNCS, vol. 7497, pp. 156–171. Springer, Heidelberg (2012)
13. Börzsönyi, S., Kossmann, D., Stocker, K.: The skyline operator. In: Proc. of ICDE, pp. 421–430. IEEE Computer Society (2001)
14. Kim, J., Pearl, J.: A computational model for causal and diagnostic reasoning in inference systems. In: Proc. of IJCAI, pp. 190–193 (1983)
15. Domingos, P., Webb, W.: A tractable first-order probabilistic logic. In: Proc. of AAAI (2012)
16. Richardson, M., Domingos, P.: Markov logic networks. Mach. Learn. 62(1/2), 107–136 (2006)
17. Zhang, X., Chomicki, J.: Semantics and evaluation of top-k queries in probabilistic databases. Distributed and Parallel Databases 26, 67–126 (2009)
18. Zhang, X.: Probabilities and Sets in Preference Querying. PhD thesis, University at Buffalo, State University of New York (2010)
19. Lacroix, M., Lavency, P.: Preferences: Putting more knowledge into queries. In: Proc. of VLDB, vol. 87, pp. 1–4 (1987)
20. Govindarajan, K., Jayaraman, B., Mantha, S.: Preference queries in deductive databases. New Generation Computing 19(1), 57–86 (2001)
21. Govindarajan, K., Jayaraman, B., Mantha, S., et al.: Preference logic programming. In: Proc. of ICLP, pp. 731–745 (1995)
22. Stefanidis, K., Koutrika, G., Pitoura, E.: A survey on representation, composition and application of preferences in database systems. TODS 36(3), 19:1–19:45 (2011)
23. Pei, J., Jiang, B., Lin, X., Yuan, Y.: Probabilistic skylines on uncertain data. In: Proc. of VLDB, pp. 15–26 (2007)
24. Atallah, M.J., Qi, Y.: Computing all skyline probabilities for uncertain data. In: Proc. of PODS, pp. 279–287. ACM (2009)
25. Lin, X., Zhang, Y., Zhang, W., Cheema, M.: Stochastic skyline operator. In: Proc. of ICDE, pp. 721–732 (2011)
26. Soliman, M., Ilyas, I., Chen-Chuan Chang, K.: Top-k query processing in uncertain databases. In: Proc. of ICDE, pp. 896–905 (2007)
27. Hansson, S.O.: Changes in preference. Theory and Decision 38, 1–28 (1995)
28. Chomicki, J.: Database querying under changing preferences. AMAI 50, 79–109 (2007)
29. Gaertner, W.: A primer in social choice theory: Revised edition. Oxford Univ. Press (2009)
30. Pini, M.S., Rossi, F., Venable, K.B., Walsh, T.: Aggregating partially ordered preferences. J. Log. Comput. 19(3), 475–502 (2009)

Towards a Cooperative Query Language for Semantic Web Database Queries

Stéphane Jean, Allel Hadjali, and Ammar Mars

LIAS/ISAE-ENSMA - University of Poitiers
1, Avenue Clement Ader, 86960 Futuroscope Cedex, France
{stephane.jean,allel.hadjali,ammar.mars}@ensma.fr

Abstract. In this paper, we address the problem of cooperative query answering on Semantic Web Databases ($\mathcal{SWDB}s$) in order to overcome the problem of empty or unsatisfactory answers. While most existing approaches propose an automatic relaxation process based on RDF-S entailment, our proposition consists in endowing $\mathcal{SWDB}$ query languages with relaxation and approximation operators. The main interest of this approach is that the relaxation process can be finely controlled by the user or implicitly by the system. Experiments on real data are conducted to evaluate (1) the scalability of the proposed operators and (2) the interest of the similarity measure defined to rank the alternative answers.

1 Introduction

Semantic Web Databases ($\mathcal{SWDB}s$) have been designed to provide an efficient management of semantic data. As users are often not aware of the contents of the queried $\mathcal{SWDB}s$, their queries may return an empty set of answers. Query relaxation is one cooperative technique proposed to solve this problem. It aims at expanding the scope of a query so that more information can be gathered in the answers. As the main novelty in this context is the fact that $\mathcal{SWDB}s$ store ontologies to explicit the semantics of data, one can leverage such ontologies for query relaxation. As illustration, let us consider a database of hotels (*Hotelsbase*) borrowed from the site `http://www.hotelsbase.org/`. These hotels are organized in a hierarchy of different types (see Figure 1) and described by a set of properties such as *price*, *stars* or *rating*. If a user wants to find a *Bed & Breakfast* hotel in Paris under 90 euros, (s)he will get an empty answer. This query can be relaxed either by weakening the predicate *price* or by changing the type of hotel desired by leveraging the subsumption relationships defined in the ontology. To choose the most relevant relaxation techniques that should be applied, most approaches use a function for ranking the possible relaxed queries. If this solution presents the advantage of being completely automatic, it relies heavily on the ranking function used. Moreover, this function can be inappropriate for the targeted domain. Another drawback of the above approaches is that the relaxation process can not be controlled. If we come back to our query example, a user may want to relax his/her price condition up to 100 euros and, if there is still no answer, change the type of hotels desired. In order to not

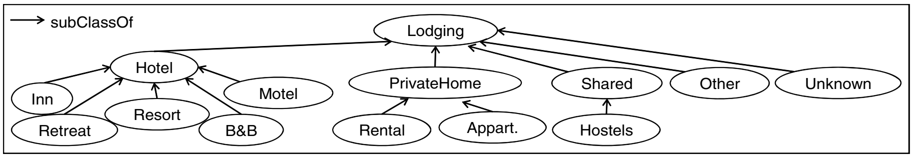


Fig. 1. Ontology example based on Hotelsbase

violate this intention, the system designer needs a tool to guide and control the relaxation process.

In this paper, we propose to extend the query language of $\mathcal{SWDB}$s. This extension consists of three operators that can be combined to finely relax an $\mathcal{SWDB}$ query and control the relaxation process according to the user preferences and/or the targeted domain [1]. These operators belong to two families: predicate and conceptual relaxations. In the former, we operate only on the (fuzzy/crisp) predicates involved in the query at hand by applying some tolerance borrowed from fuzzy set theory [2]. As for the latter, we leverage a domain ontology to substitute a concept present in the failing query by another concept by using hierarchical relations. A similarity measure is also defined to rank-order the answers of a relaxed or approximate query and return the top-k answers. The feasibility of our proposition is shown by describing its implementation on the OntoDB $\mathcal{SWDB}$.

This paper is organized as follows. Section 2 presents the three proposed operators while Section 3 shows how they can be used in an $\mathcal{SWDB}$ query language. Section 4 provides some experiments conducted to show the feasibility of our proposal and evaluate the scalability of our operators on real data. Section 5 discusses related works and Section 6 concludes the paper.

2 $\mathcal{SWDB}$ Query Approximation Operators

We consider ontologies defined as a set of classes C and properties P expressed in an ontology formalism such as RDF-S. $\mathcal{SWDB}$ queries are expressed as a graph pattern with $FILTER$ conditions composed of crisp or fuzzy predicates (fuzzy predicates are modeled thanks to trapezoidal membership functions expressed by the quadruplet (a, b, c, d) where $[b, c]$ (resp. $]a, d[$) stands for the core (resp. support) of the predicates).

2.1 The *PRED* Operator

$Signature : PRED : Q \times Pred \times Integer \times Interval \rightarrow Q.$
$PRED(q, p, \epsilon, i)$ relaxes the predicate p in the query q using a tolerance value of ϵ until the answer of the revised query is not empty or when the support is not included in the interval of validity i (see [3] for more details). The last two parameters are optional. If they are not specified, the tolerance value is set to

$\epsilon = 0.1$ and the generic interval of validity $V = [(\sqrt{5}-1)/2, (\sqrt{5}+1)/2]$ that constraints the tolerance relation of interest (expressed thanks to a particular closeness relation), is used [4].

Example. Let us consider the following query:
(?id, type, B&B) (?id, stars, 5)(?id, cityname, Belgrade) (?id, price, ?price)
FILTER **(?price, moderate)**APPROX PRED(?price)
where the semantics of the fuzzy predicate *moderate* is given by $(45, 50, 60, 65)$. As this query fails, the predicate *moderate* is relaxed by $T^{\uparrow}(moderate) = (40, 50, 60, 71.33)$ for a tolerance value $\epsilon = 0.1$. This process can be repeated n times until enough answers are found or when the interval of validity V is overstepped.

2.2 The *GEN* Operator

Signature : $GEN : Q \times C \cup P \times C \cup P \times Integer \rightarrow Q$
$GEN(q, c, c', i)$ relaxes the query q by replacing the class c by one of its superclasses c' and dropping the triples corresponding to properties not defined on c'. The parameter i specifies the maximum number of levels in the hierarchy that must be considered. The two last parameters are optional. If they are not defined, the superclass is determined using a similarity function and the process is repeated until the root class is reached (or when the result is satisfying). Considering the similarity function, both distance-based measure and information content measure have been used in previous work [5,6]. We have chosen the information content measure defined in [7] as it was more relevant to our data:

$$sim(c, c') = \frac{IC(msca(c,c'))}{IC(c)+IC(c')-IC(msca(c,c'))}$$

where $msca(c, c')$ is the concept that subsumes the two concepts being compared and $IC(c) = -logPr(c)$ corresponds to the information content of the class c which is defined according to the probability $Pr(c)$ of getting an instance of the class c in the $\mathcal{SWDB}$.

Example. Let us consider the following query: "retrieve five star Bed&Breakfast Hotels located in the city of Belgrade with a price less than 29":
(?id, type, B&B) (?id, stars, 5) (?id, cityname, Belgrade)
(?id, price, ?price) FILTER (?price <= 29) APPROX GEN(B&B)
This failing query is relaxed using the unique superclass of *B&B* i.e., *Hotel*. This relaxed query returns 1 result and thus the relaxation process stops.

2.3 The *SIB* Operator

Signature : $SIB : Q \times C \times 2^C \rightarrow Q$.
$SIB(q, c, [c_1, \cdots, c_n])$ modifies the query q by replacing the class c by its sibling class c_1. If the query does not return the desired answer, the siblings classes $c_2 \cdots c_n$ are successively used. The list of sibling classes is an optional parameter. If it is not defined, all the siblings classes are used sorted by similarities with c.

Example. Let us consider the following query: "retrieve Resort Hotels located in the city of Istanbul whose rating is at least 8":
(?id,type,Resort) (?id, cityname, Istanbul) (?id, rating, ?rating)
FILTER (?rating >= 8) APPROX SIB(Resort, [B&B, Inn])
This failing query is relaxed with the sibling class *B&B*. This relaxed query returns 2 results and the relaxation process stops.

3 $\mathcal{SWDB}$ Query Language Extension

The proposed extension of $\mathcal{SWDB}$ query languages has been illustrated by several examples in the previous section. We define here precisely this extension and introduce our ranking function of approximate answers.

3.1 Syntax of the Approx Clause

The proposed extension of $\mathcal{SWDB}$ query languages consists of a new clause *APPROX*. The syntax of this clause is as follows.

```
⟨approx clause⟩     ::= APPROX ⟨approx operator expr⟩ [TOP ⟨integer⟩]
⟨approx operator⟩   ::= ⟨pred operator⟩ | ⟨gen operator⟩ | ⟨sib operator⟩
⟨pred operator⟩     ::= PRED ( ⟨var⟩[,real][,⟨interval⟩] )
⟨gen operator⟩      ::= GEN ( ⟨literal⟩[,⟨literal⟩][,⟨integer⟩] )
⟨sib operator⟩      ::= SIB ( ⟨literal⟩ [,[⟨literal⟩(,⟨literal⟩)*]])
```

This clause is used to specify the number of results expected by the user (the default value is 1) and the operators that must be applied to obtain these answers if the result of the initial query is empty. The previously presented operators can be combined (⟨*approx operator expr*⟩) by the AND and OR constructs whose semantics is defined in the next section.

3.2 Semantics of the *Approx* Clause

The *PRED*, *GEN* and *SIB* operators previously defined can be combined with the two following constructs. (1) *AND*: the two approximations expressed by the two operands are done simultaneously. (2) *OR*: the approximation expressed by the first operand is done and if a satisfying answer is still not found, the query is relaxed using the second operand. Formally, if Op_1 and Op_2 are two operators of relaxation or approximation that can be repeated m and n times respectively:

$$Op_1(q)\ AND\ Op_2(q) \equiv Op_1(Op_2(q))$$
$$Op_1(q)\ OR\ Op_2(q) \equiv Op_1(q)\ if\ |Answers(Op_1(q))| >= k\ else\ Op_2(q)$$

As individual operator, the approximation process resulting of a combination of operators can be repeated several times:

$$(Op_1(q)\ AND\ Op_2(q))^{(i)} \equiv Op_1^{(i)}(Op_2^{(i)}(q))\ where\ i < min(m, n)$$
$$(Op_1(q)\ OR\ Op_2(q))^{(i)} \equiv Op_1^{(i)}(q)\ if\ i < m\ else\ Op_2^{(i-m)}(q)$$

3.3 Distance of the Approximate Answers with the Original Query

Let us consider the following $\mathcal{SWDB}$ query q:
(?h, type, B&B) (?h, price, ?price) FILTER(?price is moderate)
APPROX GEN(B&B, Hotel) AND PRED(?price, 0.05)
where *moderate* is defined by the quadruplet (48, 50, 55, 57). If this query fails, the relaxed query will use a variant of *moderate* given by $p = (45.5, 50, 55, 59.8)$. We need to assess the extent to which each answer of q', denoted h_i, satisfies the original query q. In previous works [5,6], the proposed measures of satisfaction take only into account the similarity between q and q': the satisfaction degree of an answer is the maximum score among all of the relaxed queries it matches. Thus, direct instances of the class $Hotel$ and direct instances of a subclass of $Hotel$ (e.g., $Motel$), that are results of the query q', will have the same degree of satisfaction. The same score will also be given to other hotels even if their prices are quite different. To solve this problem and return the top-k results more accurately, our proposed measure of satisfaction takes into account both the similarity between q and q' and the satisfaction of h_i w.r.t q':

$$SatQ(h_i) = min(Sim(q', q), SatQ'(h_i)) \tag{1}$$

Where $Sim(q', q)$ is the similarity measure between q' and q, and $SatQ(h_i)$ stands for the satisfaction degree of h_i of the query q. Formula (1) means that the satisfaction $SatQ(h_i)$ is the least degree between $Sim(q', q)$ and $SatQ'(h_i)$. This pessimistic attitude is expressed by the min operator.

Similarity Measure Computing. $Sim(q', q)$ is the aggregation of the similarity of each triple of q' with its corresponding triple in q. Thus, we have:
$Sim(q', q) = min(Sim(Hotel, B\&B), Sim(p, moderate))$

We have seen in Section 2.2 how to compute $Sim(Hotel, B\&B)$. For $Sim(p, moderate)$, the distance of Hausdorff [8] can be used:
$Sim(p, moderate) = 1/(1 + Dist(p, moderate))$

See [9] for more details on the application of the distance of Hausdorff to fuzzy sets. In our example, we obtain $Dist(p, moderate) = 0.8$.

Computation of the Satisfaction Degree. The calculus of $SatQ'(h_i)$ boils down to compute the satisfaction degree of h_i w.r.t to the gradual predicate p and its type. Arguably the degree of satisfaction of an instance of a subclass of $Hotel$ (e.g., Inn) w.r.t q' should not be the same than a direct instance of $Hotel$. As a consequence, we compute $SatQ'(h_i)$ as follows:

$$SatQ'(h_i) = min(\max_{t \in directType(h_i)} Sim(t, Hotel), \mu_p(h_i.price))$$

where *directType* is the set of the most specific classes an instance belongs to.

To show the interest of our proposed measure, table 1 presents the satisfaction degree of several approximative answers of our illustrative query q. This example shows that the ranking of our measure ($SatQ(h_i)$) is more precise than a ranking based only on $Sim(Q, Q')$. As $Sim(B\&B, Hotel) > Sim(B\&B, C_{sub})$ $\forall C_{sub} \in subClassOf(Hotel)$, the priority is given to the direct instances of $Hotel$ and then, to instances of *Resort* (the closest to *Hotel*), and so on. As most hotels are classified in the generic class *Hotel* (see section 4), the similarities between the different classes are quite low. And, since we use a pessimistic approach (expressed by the *min* operator), the ranking of the answers is dominated by the type rather than the price. This fact is illustrated by the answers h_1 and h_2 which have the same ranking even if they have different prices. This problem could be solved by using another operator of aggregation such as the product.

Table 1. Satisfaction degree of approximate answers (dt = DirectType)

h_i	$Sim(Q,Q')$	$SatQ'(h_i)$	$SatQ(h_i)$
$h_1, dt = Hotel, price = 58$	0.09	0.37	0.09
$h_2, dt = Hotel, price = 57$	0.09	0.58	0.09
$h_3, dt = Resort, price = 59$	0.09	0.08	0.08
$h_4, dt = Retreat, price = 58$	0.09	0.04	0.04
$h_5, dt = Inn, price = 59$	0.09	0.07	0.07

4 Preliminary Experimentation

We have implemented our relaxation techniques on the OntoDB Semantic Database [10] and evaluated the performance of each proposed operator on the dataset provided by Hotelsbase. This dataset is composed of 500K hotels distributed as follow: *Hotel* (370361), *B&B* (18452), *Inn* (6560), *Retreat* (539), *Resort* (14410) and *Motel* (8853). For each conceptual relaxation technique, we have done two implementations: (1) a straight implementation (2) an optimized version where the ontology is put in main memory and thus do not need to be read in the database during the relaxation algorithm. We have considered a set of queries that requires only one relaxation step and we have compared the cost of the initial query with the one of the relaxed query. Due to space limitation, we only provide results about vertical relaxation.

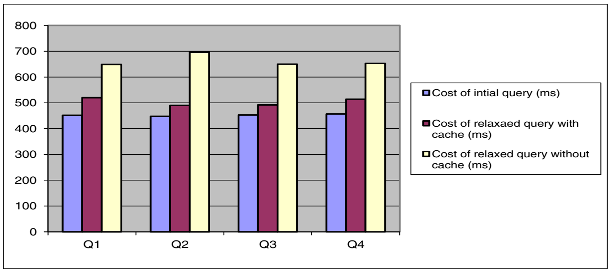


Fig. 2. Results of the experiments on vertical relaxation

Vertical Relaxation (VR). Four failing queries are considered (see below). In the right part of Figure 2, we provide the execution time of each query and its relaxed variant according to the implementation with cache (IWC) and the implementation without cache (IWOC).

$\mathtt{Q}_1$: (?id, type, B&B) (?id, stars, 5) (?id, cityname, Belgrade)
(?id, price, ?price) FILTER (?price <= 29) APPROX GEN (B&B)
$\mathtt{Q}_2$: (?id, type, INN) (?id, stars, 5) (?id, cityname, Shanghai)
(?id, price, ?price) FILTER (?price <= 15) APPROX GEN (INN)
$\mathtt{Q}_3$: (?id, type, Motel) (?id, stars, 5) (?id, cityname, Istanbul)
(?id, price, ?price) FILTER (?price <= 80) APPROX GEN (Motel)
$\mathtt{Q}_4$: (?id, type, Resort)(?id, rating, 3)(?id, stars, 3)(?id, state, California)
(?id, price, ?price) FILTER (?price < 85) APPROX GEN (Resort)

The results reported in Figure 2 illustrate the feasibility of the conceptual relaxation approach proposed, in the one hand, and confirm the claim that IWC is better than IWOC, on the other hand. One can also observe that the extra cost resulting from the execution of relaxed queries in IWC is not very important (the average of this cost amounts to *56 ms* for one relaxation step). This means that if the ontology hierarchy contains a maximum of 10 levels, the relaxation process does not exceed 5s (500ms x 10 where the average execution time of a query is 450 ms).

5 Related Work

Hurtado *et al.* [11] have introduced the relaxation of RDF queries through RDF-S entailment. This work has been extended in [12] to combine query approximation and relaxation and to consider conjunctive regular path queries. We note that the types of relaxation encompassed by these techniques do not include a fine-grained relaxation of filters (a value in a triple can only be replaced by a variable and not by an approximate value). Moreover few parameters are available in the proposed operators to control precisely the relaxation or approximation process. Dolog *et al.* [13] proposed a method for automatically relaxing over-constrained RDF queries based on background knowledge about the domain model and user preferences. The type of relaxation proposed consists in rewriting the original query by replacing some of its parts by applying specific rules. For instance, replacing a highly preferred value by a less preferred one. Compared to our approach, this approach requires to define domain and user preferences and do not use similarity relations between concepts of the ontology. Huang *et al.* have addressed the problem of relaxing RDF queries in [6] and [5]. These approaches rely on the relaxation model of [11]. (with the same limitation to relax filters) Two particular contributions are made in these papers: (i) ranking the relaxed queries using a distance-based method [6] or a measure based on information content [5] and (ii) optimizing the relaxation process to retrieve the top-k answers. We have shown in Section 3.3 that our proposed measure refines the one proposed in these papers.

6 Concluding Remarks

This paper addressed the need to control and guide the relaxation process when a $\mathcal{SWDB}$ query fails. Our proposition consists of three cooperative operators that can be called and combined in an $\mathcal{SWDB}$ query language such as SPARQL. In addition to these operators, we have defined a similarity measure to compute the distance between an approximate answer and the original query. The originality of this measure is to take into account not only the similarity between the initial and relaxed queries but also the degree of satisfaction of an answer to the relaxed query. By this way, one can distinguish between the answers of a relaxed query and thus obtain for instance the top-k approximate answers. To show the feasibility of our approach, we have implemented it on the OntoDB $\mathcal{SWDB}$ and we have made some preliminaries experiments to test the scalability of the three proposed operators. As for future work, we plan to achieve more extensive experimentations on the efficiency and effectiveness of our operators.

References

1. Tapucu, D., Jean, S., Ait-Ameur, Y., Unalir, M.O.: An extention of ontology based databases to handle preferences. In: Proc. ICEIS 2009, pp. 208–214 (2009)
2. Zadeh, L.A.: Fuzzy sets. Information and Control 8(3), 338–353 (1965)
3. Bosc, P., Hadjali, A., Pivert, O.: Incremental controlled relaxation of failing flexible queries. Journal of Intelligent Information Systems 3(3), 261–283 (2009)
4. Hadjali, A., Dubois, D., Prade, H.: Qualitative reasoning based on fuzzy relative orders of magnitude. IEEE Transactions on Fuzzy Systems 1(1), 9–23 (2003)
5. Huang, H., Liu, C., Zhou, X.: Approximating query answering on RDF databases. Journal of World Wide Web 15(1), 89–114 (2012)
6. Huang, H., Liu, C., Zhou, X.: Computing Relaxed Answers on RDF Databases. In: Bailey, J., Maier, D., Schewe, K.-D., Thalheim, B., Wang, X.S. (eds.) WISE 2008. LNCS, vol. 5175, pp. 163–175. Springer, Heidelberg (2008)
7. Pirró, G., Euzenat, J.: A feature and information theoretic framework for semantic similarity and relatedness. In: Patel-Schneider, P.F., Pan, Y., Hitzler, P., Mika, P., Zhang, L., Pan, J.Z., Horrocks, I., Glimm, B. (eds.) ISWC 2010, Part I. LNCS, vol. 6496, pp. 615–630. Springer, Heidelberg (2010)
8. Nutanong, S., Jacox, E.H., Samet, H.: An incremental hausdorff distance calculation algorithm. Proceedings of VLDB (PVLDB 2011) 4(8), 506–517 (2011)
9. Chaudhuri, B.B., Rosenfeld, A.: A modified hausdorff distance between fuzzy sets. Journal of Information Sciences 118(1-4), 159–171 (1999)
10. Jean, S., Dehainsala, H., Xuan, D.N., Pierra, G., Bellatreche, L., Aït-ameur, Y.: OntoDB: It is Time to Embed your Domain Ontology in your Database. In: Kotagiri, R., Radha Krishna, P., Mohania, M., Nantajeewarawat, E. (eds.) DASFAA 2007. LNCS, vol. 4443, pp. 1119–1122. Springer, Heidelberg (2007)
11. Hurtado, C.A., Poulovassilis, A., Wood, P.T.: Query Relaxation in RDF. In: Spaccapietra, S. (ed.) Journal on Data Semantics X. LNCS, vol. 4900, pp. 31–61. Springer, Heidelberg (2008)
12. Poulovassilis, A., Wood, P.T.: Combining Approximation and Relaxation in Semantic Web Path Queries. In: Patel-Schneider, P.F., Pan, Y., Hitzler, P., Mika, P., Zhang, L., Pan, J.Z., Horrocks, I., Glimm, B. (eds.) ISWC 2010, Part I. LNCS, vol. 6496, pp. 631–646. Springer, Heidelberg (2010)
13. Dolog, P., Stuckenschmidt, H., Wache, H., Diederich, J.: Relaxing RDF queries based on user and domain preferences. Journal of Intelligent Information System 33(3), 239–260 (2009)

The AgreementMakerLight Ontology Matching System

Daniel Faria[1], Catia Pesquita[1], Emanuel Santos[1],
Matteo Palmonari[2], Isabel F. Cruz[3], and Francisco M. Couto[1]

[1] LASIGE, Department of Informatics,
Faculty of Sciences of the University of Lisbon, Portugal
[2] Department of Informatics, Systems and Communication,
University of Milan Bicocca, Italy
[3] ADVIS Lab, Department of Computer Science,
University of Illinois at Chicago, USA

Abstract. AgreementMaker is one of the leading ontology matching systems, thanks to its combination of a flexible and extensible framework with a comprehensive user interface. In many domains, such as the biomedical, ontologies are becoming increasingly large thus presenting new challenges. We have developed a new core framework, AgreementMakerLight, focused on computational efficiency and designed to handle very large ontologies, while preserving most of the flexibility and extensibility of the original AgreementMaker framework. We evaluated the efficiency of AgreementMakerLight in two OAEI tracks: Anatomy and Large Biomedical Ontologies, obtaining excellent run time results. In addition, for the Anatomy track, AgreementMakerLight is now the best system as measured in terms of F-measure. Also in terms of F-measure, AgreementMakerLight is competitive with the best OAEI performers in two of the three tasks of the Large Biomedical Ontologies track that match whole ontologies.

1 Introduction

Ontology matching is essential for a full realization of the Semantic Web vision, by providing a means to link concepts from different ontologies. The ontology matching process takes as input two ontologies and outputs a set of correspondences between semantically related ontology concepts, also called an alignment [13]. Ontologies are fast becoming an integral part of many domains, such as biomedicine and geography, and in the last few years they have become increasingly large and complex. As a consequence the more recent ontology matching systems incorporate more elaborate approaches including scaling strategies [14,15,18], ontology repair techniques to ensure the coherence of the alignments [15], and the use of external resources and ontologies to increase the amount of available knowledge to support matching [14].

AgreementMaker has been one of the leading systems in the field of ontology and schema matching since the beginning of its development in 2001 [3,7].

It combines a powerful, flexible and extensible framework with a comprehensive user interface that enables alignment visualization and manual editing. However, AgreementMaker was not originally designed to match ontologies with more than a few thousand concepts. It relies on memory-intensive complex ontology representations and similarity matrices that store the correspondence scores between all concepts in both ontologies. These structures hinder the scalability of AgreementMaker's matching process. Furthermore, AgreementMaker is optimized for visualization, which is critical to support user interaction for semi-automatic ontology matching, but computationally intensive.

The Ontology Alignment Evaluation Initiative (OAEI) is an annual competition that provides a benchmark evaluation for ontology alignment systems [12]. In recent years the AgreementMaker system ranked among the top systems, and in particular was the top system on the Anatomy track in 2010 and 2011, with the best results ever for this track [9,10]. In 2012, reflecting the needs of the biomedical domain, the OAEI competition introduced the Large Biomedical Ontologies track. This track consists on matching ontologies with tens of thousands of terms, which poses new challenges to ontology matching systems. While AgreementMaker did not participate in OAEI 2012, it would have struggled to match ontologies of this size.

Given the increasing importance of matching very large ontologies, particularly in the biomedical domain, we have developed a novel ontology matching framework, derived from AgreementMaker and focused on the efficient matching of very large ontologies — the AgreementMakerLight (AML) framework. Unlike AgreementMaker it is not designed to support user interaction, but it maintains the flexibility and extensibility of the original framework. In this paper, we present the AML framework and its evaluation on the Anatomy and Large Biomedical Ontologies tracks of the OAEI 2012.

This paper is organized as follows: Section 2 presents some basic concepts in ontology matching, Section 3 presents the AML framework in depth, Section 4 shows the details of its evaluation, Section 5 presents and discusses the results of the evaluation, and Section 6 presents the main conclusions of our work.

2 Ontology Matching Concepts

In this section, we introduce some of the nomenclature commonly used in ontology matching, which we will use throughout this paper.

The process of *matching* or *aligning* two input ontologies (one *source* ontology and one *target* ontology) consists in finding semantic relationships between the classes of the *source* ontology and the classes of the *target* ontology. In the context of this paper, these semantic relationships are restricted to equivalence relationships, and are called *mappings*. The set of mappings between two ontologies is called an *alignment* [3].

Ontology matching systems use matching algorithms, called *matchers*, which assign a numerical value to each mapping. This numerical value reflects the semantic similarity between terms. These matchers can function at different levels, including the element level and the structural level [13].

Element-level matchers analyze concepts or their instances in isolation, ignoring their relations with other concepts or instances. These matchers can use internal knowledge only, that is, information contained in the ontology itself, or incorporate external knowledge in the form of reusable alignments, upper or domain ontologies, and other linguistic resources. A popular internal-knowledge element-level matching technique is based on the lexical matching of the labels associated with ontology concepts.

Structure-level techniques compare ontology concepts or their instances based on their relationships with other concepts or instances. They can also use external knowledge, such as instances that are not part of the ontology or previous alignments.

Most ontology systems aggregate several distinct matchers:

- sequential composition, where the results of one matcher are fed to the next
- parallel composition, where distinct matchers are run independently, and their results are combined following specific criteria,
  - homogeneous, in which the different kinds of data are processed by appropriate matchers
  - heterogeneous, in which the same input is used by distinct matchers

Furthermore, the similarity between two ontology concepts may involve the ontologies as a whole, so that the final similarity between two concepts may ultimately depend on all of them. Several approaches use this notion to propagate similarities throughout the ontology [16,11,8].

After the similarities between ontology concepts have been computed, it is necessary to use a global strategy to arrive at a final optimized alignment. These techniques can include trimming, which applies thresholds to ensure only the best matches are considered; or maximal weight matching (or weaker variants like stable marriage), which optimize the global similarity [4].

Recently, ontology matching systems have begun to include approaches to ensure the ontological quality of their outputs, such as the application of rules to prune out illogical mappings [14] or the use of full-fledged repair approaches that strive to ensure the coherence of the final alignment, that is, that all classes are satisfiable [15].

3 AgreementMakerLight Framework

3.1 Overview

The AML framework was programmed in Java and developed in Eclipse. The core framework includes two modules: the ontology loading module and the ontology matching module. The ontology loading module is responsible for loading the input ontology files and constructing ontology objects (Figure 1). It is also responsible for loading external ontologies used as background knowledge. The ontology matching module is responsible for aligning ontology objects by combining one or more matching algorithms and one or more selection steps (Figure 2).

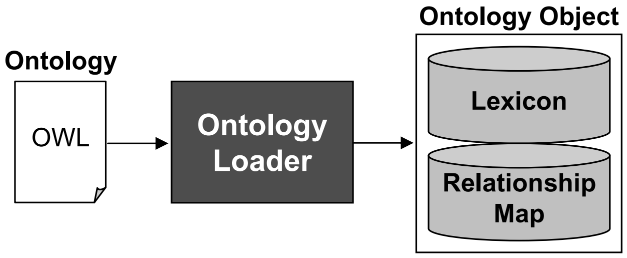


Fig. 1. Schema of the AgreementMakerLight Ontology Loading Module

Like AgreementMaker, the AML ontology matching module was designed with flexibility and extensibility in mind, and thus allows for the inclusion of virtually any matching algorithm.

The AML framework also includes three key data structures: the *Lexicon*, the *RelationshipMap* and the *Alignment*. The first two data structures are the main components of Ontology Objects, i.e., representations of ontologies used in AML to support the matching process. As their names suggest, the *Lexicon* stores the lexical information of an ontology (i.e., the labels and synonyms of each term) and the *RelationshipMap* stores the structural information of an ontology (i.e., the relationships between all terms). The *Alignment* stores the set of mappings between two ontologies produced by one or more matching algorithms.

3.2 Ontology Loading Module

As is the case in AgreementMaker, the AML ontology loading module is currently based on the Jena2 ontology API (Jena), and the first step in the loading process is to read the ontology file into memory as a Jena OntModel. The difference between the two systems is that AML stores in internal data structures all the information from the OntModel that is necessary for ontology matching, whereas AgreementMaker keeps the OntModel in memory throughout the matching process.

After using Jena to read an input ontology file, the AML ontology loading module extracts from the OntModel the following information, which is linked together under an Ontology Object:

- The URI or the ontology.
- A list of URI of all named classes in the ontology that belong to the ontology's namespace.
- A list of local names of all listed classes.
- A list of the synonym properties used in the ontology (e.g., *hasExactSynonym, hasRelatedSynonym*).

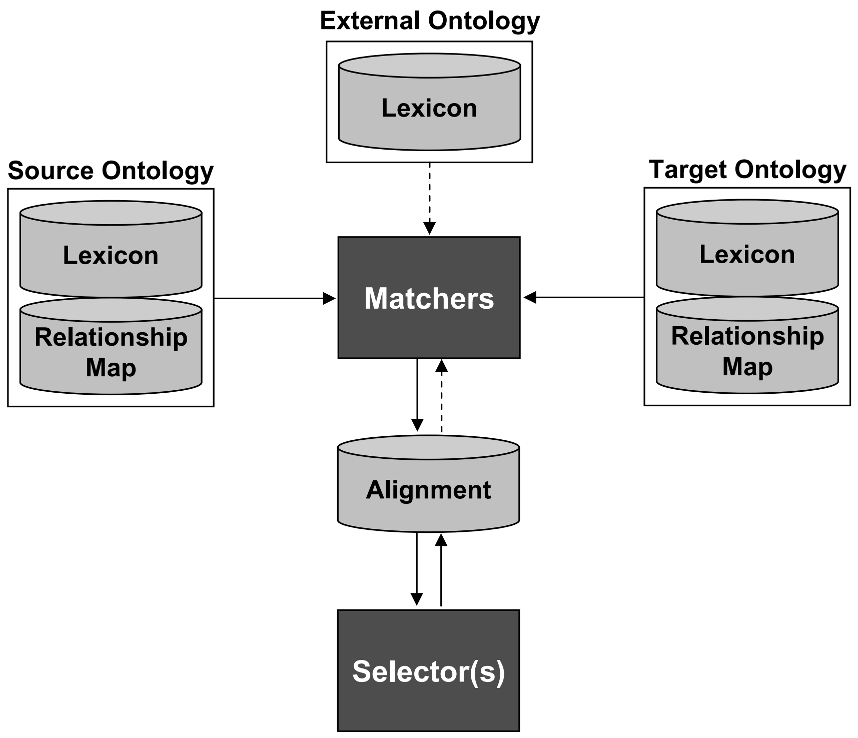


Fig. 2. Schema of the AgreementMakerLight Ontology Matching Module

- A *Lexicon* that contains the local names of all listed classes (when they are not alphanumeric codes), their labels, and all their synonyms (as declared in synonym property statements).
- A *RelationshipMap* that contains the *is a* and *part of* relationships between all listed classes (with transitive closure) plus all disjoint clauses between the listed classes (without transitive closure).

Building the *RelationshipMap* is optional, since the relationships are unnecessary for many matching algorithms, and this step takes approximately 60% of the total ontology loading time. In particular, the *RelationshipMap* is not built when loading external ontologies to use as background knowledge, as only their *Lexicon* is necessary for this purpose.

Even when the *RelationshipMap* is built, the whole ontology loading process is fairly quick: in our 4 core CPU server with 16 GB total RAM, small ontologies (under 10,000 classes) are loaded in 1 or 2 seconds and even very large ontologies (approximately 120,000 classes) are loaded in under 150 seconds.

3.3 Data Structures

A key difference between AML and AgreementMaker is that in the former ontologies are represented exclusively by internal data structures, whereas in the latter internal data structures are used in addition to the Jena OntModel. While building the internal data structures from the OntModel takes time, if those structures are designed with the efficiency of the matching process in mind, they will reduce the total processing time considerably. Furthermore, the internal data structures take up less memory than the OntModel, so in not keeping the latter in memory, we effectively increase the available memory for the matching process. Last but not least, this setup means that AML's ontology matching module is not tied to Jena or any specific ontology-reading API. Thus AML can work with any ontology-reading API by simply changing the ontology loading module.

Lexicon is a data structure that links each class in an ontology with its "names" (i.e., local names, labels, and synonyms) and the provenance of those names (i.e., whether they come from a local name or label, or from which type of synonym property statement). While an equivalent data structure already existed in AgreementMaker, it was built after the ontology loading process and only used by some matching algorithms. In AML, the *Lexicon* is a primary data structure used by all matching algorithms that require lexical information.

A novel aspect that was incorporated in the AML *Lexicon* was a system of weights to reflect the reliability of each provenance. For instance, synonyms obtained from *hasExactSynonym* statements are in principle more reliable than synonyms obtained from *hasRelatedSynonym* statements, as they should be closer in meaning to the concept described by a class. Thus, local names were given a weight of 1.0, labels a weight of 0.95, exact synonyms a weight of 0.9 and other synonyms a weight of 0.85. These weights may be used by any matching algorithm that uses the *Lexicon*.

The internal structure of the *Lexicon* consists on two MultiMaps (which are HashMaps of HashMaps) containing classes, names and provenances, with one having the class as key and the other having the name as key. Thus, the *Lexicon* can be queried by both class and name at virtually no computational cost.

RelationshipMap is a data structure that links each class to the classes related to it through *is a* or *part of* relationships or disjoint clauses. It complements the *Lexicon*, and is a very efficient alternative to the node-based tree structure used in AgreementMaker to represent each ontology.

The *RelationshipMap* stores all *is a* and *part of* paths in an ontology with transitive closure, and includes the distance of each path in number of edges. It also stores all direct disjoint clauses in an ontology (without transitive closure). Like the *Lexicon*, the *RelationshipMap* is based on MultiMaps. It includes two MultiMaps for relationships which contain ancestors, descendents and relationship (i.e., type and distance), with one having the ancestor as key and the other one having the descendent as key. It also includes a HashMap of Sets for disjoint clauses, linking each class to all classes that are disjoint with it. Thus the RelationshipMap can be queried to obtain all descendents of a class, all ancestors of a class, and all classes disjoint with a class at virtually no computational cost.

Alignment is a data structure used by the ontology matching module to store mappings between the input ontologies. This structure was already used by AgreementMaker, and was ported directly from it. However, in AgreementMaker, *Alignment* was used only to store the final output of a matching algorithm or combination of algorithms. During the matching procedure, the primary data structure used by AgreementMaker is a matrix that stores the similarities between all concepts of the source ontology against all concepts of the target ontology, which we abbreviate in the rest of the paper as a *all-against-all* strategy. The problem with this structure is that it takes $O(m \times n)$ memory (where m and n are the number of concepts of the source and target ontologies, respectively) and therefore does not scale for large ontologies. For instance, the matrix that results from matching two ontologies with 50,000 classes would occupy 18.6 GB of memory, which is beyond the capacity of our server. Since the number of mappings in a matching problem with cardinality one-to-one is $O(min(m, n))$, the vast majority of the values in the similarity matrix is very small or zero, thus making their storage unnecessary. In AML we opted for storing similarities directly in the *Alignment* and discarding similarities that are below a given threshold.

The internal structure of *Alignment* in AML is identical to that of AgreementMaker. It includes two MultiMaps that contain the source class, target class and similarity, with one having the source class as key and the other one having the target class as key. This enables efficient querying of mappings by class, and means that *Alignment* corresponds to a sparse matrix. In addition, *Alignment* also includes a list structure that enables sorting and thus facilitates selection.

3.4 Ontology Matching Module

The AML ontology matching module contains three components: *Matchers* (i.e., matching algorithms), *Selectors* (i.e., selection algorithms), and the previously described *Alignment* data structure (see Figure 2).

Matchers are algorithms that compare two ontologies and return an *Alignment* between them. Like in AgreementMaker, any kind of matching algorithm can be implemented as an AML *Matcher*, requiring only the implementation of a matching algorithm (which receives as input two ontology objects and returns an *Alignment* between them) and an extension method (which receives as input two ontology objects and an *Alignment* between them, and extends that *Alignment* by mapping only unmapped classes). However, AML divides *Matchers* into *Primary Matchers* and *Secondary Matchers* according to their efficiency.

Primary Matchers are matching algorithms that rely on HashMap cross-searches (i.e., searches where a key in a HashMap is used to query another HashMap directly) and thus take $O(n)$ time. *Secondary Matchers* are matching algorithms that make non-literal comparisons between terms and thus require explicitly comparing each term in an ontology with each term in the other ontology, which takes $O(n^2)$ time. This means that *Secondary Matchers* cannot be used efficiently to match large ontologies. Taking this into account, we stipulated that the extension method implemented by *Secondary Matchers* should be less extensive than the one implemented by *Primary Matchers*. Thus, in *Secondary Matchers* the extension method only tries to map classes in the vicinity of previously mapped ones (e.g., children, parents, siblings), whereas in *Primary Matchers* the extension methods tries to map all unmapped classes. In general, *Primary Matchers* in AML are used in match mode whereas *Secondary Matchers* are used in extension mode.

One of the unique features of AgreementMaker is its strategy for combining the output of multiple *Matchers* to obtain a better final alignment, the linear weighted combination [3]. However, this strategy relies on the all-against-all similarity matrix produced by each *Matcher*, and thus is not directly portable to AML. Currently AML combines the output of multiple *Matchers* by simply joining the alignments and keeping the highest similarity in case of repeated mappings, which was a strategy previously used in AgreementMaker [5,6].

Selectors are algorithms used to trim an *Alignment* by excluding mappings below a given similarity threshold and excluding competing mappings (i.e., multiple mappings that include the same class) to obtain the desired cardinality. As already mentioned, the desired cardinality in ontology matching is typically one-to-one. The problem of trimming a many-to-many *Alignment* to obtain a one-to-one *Alignment* corresponds to a bipartite mapping problem. The selection algorithm implemented in AgreementMaker finds the maximum weighted bipartite mapping, thus maximizing the sum of the similarities of the selected mappings [4]. However, this algorithm relies on the all-against-all similarity matrix, and thus is not directly portable to AML. Nevertheless, it is not clear that the maximum mapping is the "optimal" mapping for an ontology alignment problem, since the goal of the problem is to maximize the number of correct mappings while minimizing the number of incorrect mappings. Thus, assuming that the similarity value given by the matching algorithms is correlated with the probability that they are correct, then selecting a high-quality alignment (90% similarity) might be better than selecting two competing medium-quality alignments (60% similarity), but the maximum mapping would include the latter.

Taking the above considerations into account, we developed a greedy *Ranked Selector* algorithm for AML, that selects mappings based on their similarity. This simple algorithm starts by sorting the mappings in the *Alignment* in descending order of their similarity values, then selects mappings in that order, as long as they do not include classes that were present in previously selected mappings. This ensures that each class appears in at most one mapping, but favors stronger individual mappings (in terms of similarity) instead of trying to maximize the total similarity of the selected mappings.

3.5 Implemented Matchers

In porting matchers from AgreementMaker, a paramount idea was that, whenever possible, we would eliminate the need for an all-against-all comparison and instead convert them into *Primary Matchers* whenever possible. We implemented four matchers: the *Lexical Matcher*, the *Mediating Matcher*, the *Word Matcher* and the *Parametric String Matcher*. The first three were implemented as *Primary Matchers* and the last one was implemented as a *Secondary Matcher*.

Lexical Matcher is one of the simplest and most efficient matching algorithms, which looks for literal name matches in the *Lexicons* of the input ontologies. It is derived from the *Lexical Synonym Weighted Matcher* of AgreementMaker, but uses a more streamlined weight system. For two input ontologies *source* and *target*, the *Lexical Matcher* algorithm is as follows:

```
set A = empty alignment
set list = names(source)
for each name in list
  if target contains name
    set sourceList = sourceClasses(name)
    set targetList = targetClasses(name)
    for each sourceClass in sourceList
      set weightSource = weight(name,sourceClass)
      for each targetClass in targetList
        set sim = weightSource * weight(name,targetClass)
        add (sourceClass,targetClass,sim) to A
end
```

Note that, regardless of the order in which the input ontologies are given, the *Lexical Matcher* always iterates through the *Lexicon* of the ontology that has the smallest number of names, so as to minimize the number of iterations.

Mediating Matcher is also a simple and efficient matching algorithm based on literal name matches, but it uses a third (external) ontology as a mediator between the input ontologies. It uses the *Lexical Matcher* to compute "bridge" alignments between the input ontologies and the mediating ontology, then cross-searches the bridge alignments to map the input ontologies. For two input ontologies *source* and *target* and a mediating ontology *med*, the *Mediating Matcher* algorithm is as follows:

```
set A = empty alignment
compute Bs = LexicalMatcher(med,source)
compute Bt = LexicalMatcher(med,target)
for each medClass in Bs
  if Bt contains medClass
    for each sourceClass in Bs(medClass)
      set simSource = similarity(medClass,sourceClass)
      for each targetClass in Bt(medClass)
        set sim = simSource * similarity(medClass,targetClass)
        add (sourceClass,targetClass,sim) to A
end
```

Word Matcher is a word-based string similarity algorithm that measures the similarity between two classes through a weighted Jaccard index between the words present in their names. It is derived from the *Vector-based Multi-word Matcher* of AgreementMaker, but is based on a lexical cross-search whereas the latter requires an all-against-all comparison.

The first step in the *Word Matcher* is the derivation of a *Word Lexicon* from the *Lexicon* of each ontology and computing the frequency and evidence content (EC) of each word. The EC of each word is given by the inverse logarithm of its frequency [2]. After computing the *Word Lexicons*, the *Word Matcher* algorithm is as follows:

```
set A = empty alignment
set list = words(source)
for each word in list
  if target contains word
    set sourceList = sourceClasses(word)
    set targetList = targetClasses(word)
    for each sourceClass in sourceList
      set weightS = sourceEC(word) * weight(word,sourceClass)
      for each targetClass in targetList
        set weightT = targetEC(word) * weight(word,targetClass)
        set sim = sqrt(weightS * weightT)
        if A contains (sourceClass,targetClass)
          set A(sourceClass,targetClass) += sim
        else
          add (sourceClass,targetClass,similarity) to A
for each (sourceClass,targetClass) in A
  set itr = similarity(sourceClass,targetClass)
  set uni = weight(sourceClass) + weight(targetClass) - itr
  set A(sourceClass,targetClass) = itr/uni
end
```

Despite also being based on a lexical cross-search, the *Word Matcher* has a steeper memory requirement than the previous two matchers. The reason for this is that it needs to store temporary mappings between all classes that share at least one word, since it can only filter out low similarity mappings after iterating through all words. In the worst case, its memory requirement will be $O(n^2)$ for very verbose ontologies. Thus, despite being a primary matcher, the *Word Matcher* requires that very large ontologies be partitioned due to memory constraints.

Parametric String Matcher is a string similarity algorithm that implements a variety of similarity metrics and was directly ported from AgreementMaker. Since it makes non-literal string comparisons, it requires an all-against-all comparison of the ontologies and therefore is a secondary matcher in AML.

4 Evaluation

We evaluated AML on the Anatomy and Large Biomedical Ontologies tracks of the OAEI 2012 [1].

The Anatomy track consists on a medium-size ontology matching task between the Adult Mouse Anatomy Dictionary (with 2737 classes) and part of the NCI thesaurus describing human anatomy (with 3298 classes). It is evaluated using a manually created reference alignment that has been extensively tested. We were interested in evaluating AML in this track given that AgreementMaker was the top system in it, both in 2010 and 2011. Thus, we wanted to assess whether the improvement in efficiency of AML came at the cost of performance in terms of precision, recall, and F-measure. As such, we selected a configuration for AML that is as close as possible to the configuration used by AgreementMaker in OAEI 2011. It combines the *Lexical Matcher*, the *Mediating Matcher* using UBERON [17] as the background ontology, the *Word Matcher* and the *Parametric String Matcher* in extension mode. All *Matchers* were used with a minimum similarity threshold of 0.6 except the *Parametric String Matcher*, which was used with a higher threshold of 0.7.

The Large Biomedical Ontologies track includes a total of 9 ontology matching tasks between three ontologies: the Foundational Model of Anatomy (FMA), the NCI thesaurus and the SNOMED Clinical Terms. For each pairwise combination of these ontologies, there is a task for matching small overlapping fragments, a task for matching extended fragments, and a task for matching the whole ontologies. The reference alignments used in OAEI 2012 to evaluate these tasks were obtained automatically from UMLS, and then repaired with ALCOMO and/or LogMap. We were interested in evaluating AML on the three whole ontologies tasks given that AgreementMaker was not able to handle them. Thus, we wanted to assess whether the improvement in efficiency of AML was sufficient for it to compete with the top systems in OAEI in terms of run time, and what was its baseline performance. Thus we selected a simple configuration for AML that combined only the *Lexical Matcher* (with a threshold of 0.7) with the *Parametric String Matcher* in extension mode (with a threshold of 0.7). We evaluated AML using the reference alignments repaired by ALCOMO for the

FMA-NCI and FMA-SNOMED tasks, and the reference alignment repaired by LogMap for the SNOMED-NCI task.

AML and AgreementMaker run times were measured in our local server, with 4 core CPU and 16 GB total RAM. They are not directly comparable to the run times reported for the OAEI 2012 Anatomy track, which were measured on a less powerful server (2 core CPU with 3 GB allocated RAM), but can be compared to the run times for the Large Biomedical Ontologies track, which were measured on a more powerful server (16 core CPU with 15 GB allocated RAM).

5 Results and Discussion

The AML results for the Anatomy track are presented in Table 1 together with those of AgreementMaker and the top three systems in OAEI 2012 [1]. Surprisingly, the performance of AML in terms of F-measure was slightly better than that of AgreementMaker, and also slightly better than the top system in OAEI 2012. The comparison with AgreementMaker is interesting because the configuration of the systems was very similar, but AgreementMaker has a more exaustive matching process (including the *Parametric String Matcher* used for global matching and the LWC combination strategy [4]) and therefore we were not expecting AML to perform better than AgreementMaker. While AgreementMaker did have a higher recall than AML, this was compensated by the higher precision of the latter. The main reason behind this increase in precision is likely the new weighting system used in the *Lexicon*, as this is the only substantial change made over AgreementMaker. In any case, it is clear that the features of AgreementMaker that were excluded from AML for the sake of efficiency did not affect AML's performance. They did, however, have a significant effect on efficiency, as the run time of AML was only 5% of the run time of AgreementMaker on the same server and conditions. While we cannot compare our run time directly with those reported for the Anatomy track in OAEI 2012 (since our server is more powerful than the server used in this track) a run time of 10 seconds is nevertheless a clear indication of the efficiency of AML.

Regarding the three tasks of the Large Biomedical Ontologies track, AML was able to match all ontologies in considerably less time than the best systems in OAEI 2012 (see Table 2). Furthermore, the OAEI 2012 run times were obtained

Table 1. Evaluation of the AgreementMakerLight system in the OAEI 2012 Anatomy track

System	Precision	Recall	F-Measure	Run Time (s)
AgreementMakerLight	96.1%	88.9%	92.4%	10*
GOMMA-bk	91.7%	92.8%	92.3%	15
AgreementMaker	95.2%	89.4%	92.2%	200*
YAM++	94.3%	85.8%	89.8%	69
CODI	96.6%	82.7%	89.1%	880

* Run times obtained using a more powerful server than the one used for OAEI 2012.

Table 2. Evaluation of the AgreementMakerLight system on the OAEI 2012 Large Biomedical Ontologies Track

System	Precision	Recall	F-Measure	Run Time (s)
FMA-NCI				
YAM++	86.2%	83.8%	85.0%	1304
GOMMA	82.9%	83.6%	83.3%	217
ServOMapL	84.4%	80.8%	82.6%	251
AgreementMakerLight	84.7%	71.8%	78.0%	89*
FMA-SNOMED				
ServOMapL	85.1%	69.1%	76.3%	517
ServOMap	84.2%	65.5%	73.7%	517
YAM++	79.1%	68.5%	73.4%	23900
AgreementMakerLight	88.0%	18.8%	40.7%	224*
SNOMED-NCI				
YAM++	78.5%	60.4%	68.3%	30155
ServOMapL	78.5%	59.8%	67.9%	738
LogMap	81.2%	57.7%	67.4%	955
AgreementMakerLight	91.8%	49.1%	67.1%	231*

* Run times obtained using a less powerful server than the one used for OAEI 2012.

in a more powerful server, so the difference between AML and these systems should be even greater than revealed by the results. In terms of quality (as measured by the F-measure), the results of AML are behind those of the top systems in OAEI 2012. This was expected, since we used only a simple configuration with two matching algorithms, as our main goal was assessing the efficiency of AML. Nevertheless, the results obtained in the FMA-NCI and SNOMED-NCI tasks are competitive, with a high precision (higher than the top OAEI 2012 systems, in the case of the SNOMED-NCI task) and a reasonable recall. We expect to improve these baseline results significantly once we focus on quality, namely by introducing adequate external information, which is a common feature of all the top OAEI 2012 systems. Given AML's advantage in terms of run time, we expect to obtain results comparable to those of the top OAEI 2012 systems in these tasks, while taking less time.

The results for the FMA-SNOMED task were excellent in terms of run time and precision, but not in terms of recall. However, we identified the problem behind the low recall, which can be overcome without difficulty: we noticed that SNOMED includes the word "structure" in the labels of most anatomical structures (e.g., "structure of hair of trunk", "spinal nerve structure") whereas FMA names the same structures directly (e.g., "hair of trunk", "spinal nerve"). Since our matching configuration for this task was based on the *Lexical Matcher*, which is a literal name matcher, any such mappings will not be found (they may be found by the *Parametric String Matcher*, but only if they lie in the neighborhood of mappings found by the *Lexical Matcher*). One simple strategy to circumvent this problem is to use a word matching algorithm such as our *Word Matcher*. Thus, we expect to be able to obtain a significantly higher recall once we optimize the *Word Matcher*.

6 Conclusions

Overall, the AML framework has fulfilled the goals for which it was designed. On the one hand, it was able to match very large ontologies efficiently, in a competitive time when compared with top ontology matching systems. On the other hand, it was able to produce high quality results in the Anatomy track, that surpassed even those of AgreementMaker. In comparison with AgreementMaker, AML represents a substantial improvement in terms of efficiency without sacrificing performance, as measured in terms of precision, recall, and F-measure. Furthermore, AML shares the focus of AgreementMaker on flexibility and extensibility, which are part of its design strengths [3].

In the Anatomy track, the results obtained by AML place it above the systems that competed in OAEI 2012. We believe that AML can improve even further upon these results, considering that scalability to large and very large ontologies has been, until now, the main focus of the work behind AML. Regarding the Large Biomedical Ontologies track, AML obtained excellent run times, but it is clear that performance can be improved—especially recall—once we enlarge our focus. In fact, even without focusing on performance, the results obtained by AML on the FMA-NCI and SNOMED-NCI tasks were competitive. Thus, we expect AML to place among the very best systems in these tasks, with the inclusion of suitable external data and a more complete matching configuration. As for the FMA-SNOMED task, we identified that the cause for the low recall obtained by AML was the unusual naming convention of the anatomical structures in SNOMED. We expect to circumvent this problem by using a word-based matching algorithm, which together with the inclusion of external data, will enable AML to obtain competitive results in this task while remaining among the fastest systems.

Further improvements to the AML framework will include innovative strategies for using external resources, repairing alignments, and using semantic similarity in the context of structural matching [19].

Acknowledgments. DF, CP, ES and FMC were funded by the Portuguese FCT through the SOMER project (PTDC/EIA-EIA/119119/2010) and the multi-annual funding program to LASIGE. The research of IFC was partially supported by NSF Awards 0812258, 1143926, and 1213013, and by UIC Area of Excellence and Civic Engagement Research Awards.

References

1. Aguirre, J.L., Eckert, K., Euzenat, J., Ferrara, A., van Hage, W.R., Hollink, L., Meilicke, C., Nikolov, A., Ritze, D., Scharffe, F., Shvaiko, P., Sváb-Zamazal, O., dos Santos, C.T., Jiménez-Ruiz, E., Grau, B.C., Zapilko, B.: Results of the Ontology Alignment Evaluation Initiative 2012. In: ISWC International Workshop on Ontology Matching (OM). CEUR Workshop Proceedings, vol. 946, pp. 73–115 (2012)
2. Couto, F., Silva, M., Coutinho, P.: Finding genomic ontology terms in text using evidence content. BMC Bioinformatics 6(suppl. 1), S21 (2005)

3. Cruz, I.F., Palandri Antonelli, F., Stroe, C.: AgreementMaker: Efficient Matching for Large Real-World Schemas and Ontologies. PVLDB 2(2), 1586–1589 (2009)
4. Cruz, I.F., Palandri Antonelli, F., Stroe, C.: Efficient Selection of Mappings and Automatic Quality-driven Combination of Matching Methods. In: ISWC International Workshop on Ontology Matching (OM). CEUR Workshop Proceedings, vol. 551, pp. 49–60 (2009)
5. Cruz, I.F., Palmonari, M., Caimi, F., Stroe, C.: Towards "On the Go" Matching of Linked Open Data Ontologies. In: IJCAI Workshop Discovering Meaning on the Go in Large & Heterogeneous Data (LHD), pp. 37–42 (2011)
6. Cruz, I.F., Palmonari, M., Caimi, F., Stroe, C.: Building Linked Ontologies with High Precision Using Subclass Mapping Discovery. Artificial Intelligence Review (2012) (to appear)
7. Cruz, I.F., Stroe, C., Caimi, F., Fabiani, A., Pesquita, C., Couto, F.M., Palmonari, M.: Using AgreementMaker to Align Ontologies for OAEI 2011. In: ISWC International Workshop on Ontology Matching (OM). CEUR Workshop Proceedings, vol. 814, pp. 114–121 (2011)
8. Cruz, I.F., Sunna, W.: Structural Alignment Methods with Applications to Geospatial Ontologies. Transactions in GIS 12(6), 683–711 (2008)
9. Euzenat, J., Ferrara, A., Meilicke, C., Pane, J., Scharffe, F., Shvaiko, P., Stuckenschmidt, H., Sváb-Zamazal, O., Svátek, V., dos Santos, C.T.: Results of the Ontology Alignment Evaluation Initiative 2010. In: ISWC International Workshop on Ontology Matching (OM). CEUR Workshop Proceedings, vol. 689, pp. 85–117 (2010)
10. Euzenat, J., Ferrara, A., van Hage, W.R., Hollink, L., Meilicke, C., Nikolov, A., Ritze, D., Scharffe, F., Shvaiko, P., Stuckenschmidt, H., Sváb-Zamazal, O., dos Santos, C.T.: Results of the Ontology Alignment Evaluation Initiative 2011. In: ISWC International Workshop on Ontology Matching (OM). CEUR Workshop Proceedings, vol. 814, pp. 85–113 (2011)
11. Euzenat, J., Loup, D., Touzani, M., Valtchev, P.: Ontology alignment with OLA. In: Proceedings of the 3rd International Workshop on Evaluation of Ontology based Tools (EON), Hiroshima, Japan (2004)
12. Euzenat, J., Meilicke, C., Stuckenschmidt, H., Shvaiko, P., Trojahn, C.: Ontology alignment evaluation initiative: Six years of experience. In: Spaccapietra, S. (ed.) Journal on Data Semantics XV. LNCS, vol. 6720, pp. 158–192. Springer, Heidelberg (2011)
13. Euzenat, J., Shvaiko, P.: Ontology matching. Springer-Verlag New York Inc. (2007)
14. Groß, A., Hartung, M., Kirsten, T., Rahm, E.: GOMMA results for OAEI 2012. In: Ontology Matching Workshop, International Semantic Web Conference 2012 (2012)
15. Jiménez-Ruiz, E., Grau, B.C., Zhou, Y.: LogMap 2.0: towards logic-based, scalable and interactive ontology matching. In: Proceedings of the 4th International Workshop on Semantic Web Applications and Tools for the Life Sciences, pp. 45–46 (2011)
16. Melnik, S., Garcia-Molina, H., Rahm, E.: Similarity flooding: a versatile graph matching algorithm and its application to schema matching. In: Proceedings of the 18th International Conference on Data Engineering, pp. 117–128 (2001)
17. Mungall, C.J., Torniai, C., Gkoutos, G.V., Lewis, S., Haendel, M.A.: Uberon, an Integrative Multi-species Anatomy Ontology. Genome Biology 13(1), R5 (2012)
18. Ngo, D., Bellahsene, Z.: Yam++: A multi-strategy based approach for ontology matching task. In: ten Teije, A., Völker, J., Handschuh, S., Stuckenschmidt, H., d'Acquin, M., Nikolov, A., Aussenac-Gilles, N., Hernandez, N. (eds.) EKAW 2012. LNCS, vol. 7603, pp. 421–425. Springer, Heidelberg (2012)
19. Pesquita, C., Stroe, C., Cruz, I.F., Couto, F.: BLOOMS on AgreementMaker: Results for OAEI 2010. In: ISWC International Workshop on Ontology Matching (OM). CEUR Workshop Proceedings, vol. 689, pp. 135–141 (2010)

Flexible Matchmaking for RESTful Web Services

Fatma Slaimi, Sana Sellami, Omar Boucelma, and Ahlem Ben Hassine

National School of Computer Science (ENSI), University of Manouba, Tunisia
Aix-Marseille Univ, LSIS, UMR CNRS 7296, Marseille, France
{fatma.slaimi,ahlembh}@gmail.com,
{sana.sellami,omar.boucelma}@lsis.org

Abstract. This paper describes a flexible matchmaking approach that relies on an extensible set of schema/ontology matching techniques and primitives that may be combined in various ways, accordingly with the specification of a service request. The approach has been implemented and tested against a collection of RESTFul web services described in hRESTS microformat.

Keywords: Semantic Web Services Discovery, RESTful Services.

1 Introduction

With the proliferation of web services (WS), WS discovery has been identified as one of the key challenges for achieving efficient service oriented computing. As the number of (publicly) available web services keeps growing, discovering the right service(s) is still an issue although a significant amount of work has been done during the last decade.

With the advent of the Semantic Web, new semantic techniques and systems has been proposed leading a new research topic known as Semantic Web Service Discovery. A number of solutions has been proposed in the literature, and resulted in different semantic matchmaking techniques and systems [1]. Despite commonalities shared with the schema matching research area [1], (semantic) web services discovery techniques have been proposed along a separate and independent pathway. Nevertheless, during the past few years, some schema matching approaches have been proposed to address WS discovery [2].

In this paper, we advocate an "unbundled" approach (versus a monolithic one) for semantic web services discovery. The approach relies on an extensible set of schema/ontology matching techniques and primitives that may be combined in various ways, accordingly with the specification of a service request. Matching techniques have been primarily developed to (semi)automatically allow the discovery of correspondences between (database/XML) schemas, mostly in the context of Information Integration. Matching approaches depend on the description of services and can be applied at the element or structure levels; they may be terminological, structural, or semantic [15].

[1] http://www-ags.dfki.uni-sb.de/~klusch/s3/index.html

The contribution of the paper is twofold: (1) a RESTFul matchmaking approach based on schema matching, and (2) an experimental evaluation performed in using the well known hRESTS-TC1 service retrieval test collection [2].

The remainder of this paper is organized as follows: In Section 2, we review the semantic web services concepts that are mandatory for the comprehension of the rest of the paper; in Section 3 we survey some related work, before we detail our approach in Section 4. Section 5 describes and discusses our evaluation environment. Finally we conclude in Section 6.

2 Semantic Web Services

Several languages for semantic annotation of Web services are proposed in the literature. The most known are OWL-S (Web Ontology Language for Web Services), WSMO (Web Service Modeling Ontology), WSML (Web Service Modeling Language), and annotation languages like WSDL-S (Web Service Semantics), SAWSDL (Semantic Annotations for WSDL and XML Schema) and SA-REST . These languages differ in their complexity and expressive power. Languages such as OWL-S and WSMO aim to define an ontology for semantic web service description. However, the annotation languages propose to add annotations as an extension of WSDL description and connect the annotated elements with semantic information. The main goal of semantic annotation approaches is to enhance the syntactic description of web services with semantic information. The most important elements of web services that need to be annotated are functional properties related to the input, output, conditions and effects. These elements are processed during service discovery and composition. Instead of traditional SOAP-based services, which can use different standards to be described both syntactically that semantically, there are no standard for REST based services. Rather, in almost all cases, RESTful services are described with full text documentation which details the functionality of the service and the way to use it. hRESTS (HTML for RESTful Services) [3] is a micro format that enables the creation of machine-processable Web API descriptions based on available HTML documentation. hRESTS is complemented by the MicroWSMO microformat, which supports the semantic annotation of service properties in a SAWSDL-like manner. MicroWSMO introduces additional HTML classes, in order to enable the linking of ontological elements and the provisioning of machinery for transforming data exchanged between two services used in a service composition.

The main components of hRESTS services are:

- service : interface that the client deals with,
- operation : an action that the client can perform,
- resources : the address (URI) to invoke the operation
- method : detect the HTTP method
- input/outputs messages :request and response are the messages sent as input and output of the operation

[2] Available at `http://www.semwebcentral.org/`

– links : in the output messages, make a run-time hypertext graph of related resources.

hRESTS service functional model can be presented by the following Fig.1. This model is very similar to WSDL model [3], hRESTS aims to provide a machine-readable representation of common Web service description.

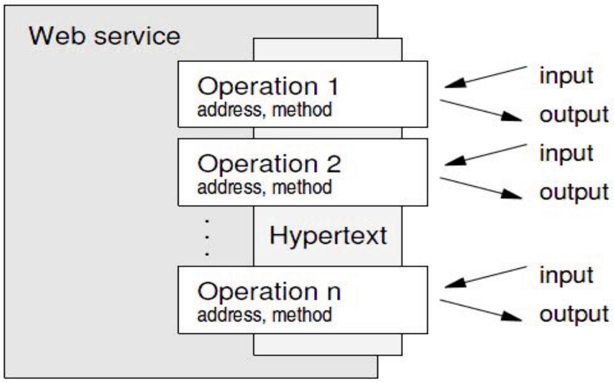


Fig. 1. hRESTS Service Model ([3])

hRESTS web service description represent only syntactic information, it doesn't include semantics. To semantically annotate this description, the MicroWSMO microformat can be used. It has been proposed in [3]. It define links to semantic concepts (ontology represented in the Web Ontology Language) to describe web service components.

3 Related Work

During the last decade, several Semantic Web Service Discovery approaches have been proposed in the literature. Most of them have been devoted to the matchmaking of SAWSDL[4] [5] [6] and OWL-S services [7] [8]. These matchmakers can be logic-based, non logic-based or hybrid, i.e., a combination of logic and non logic according to the classification proposed in [9].

Discovering RESTful services approaches has been recently proposed in the literature [10] [11] [12]. The first work XAM4SWS [10] compares operation, input, output and service similarities. These similarities are combined in the single one for each pair of operations. Semantic matchmaking considers hRESTS characteristics and determines verb (get, post, put, delete) similarity values. The second work [11] proposed a graph-theoretic approach for matching RESTful services. This approach matches RESTful web services functionalities based on their WADL(Web Application Description Language) elements. It uses linguistic knowledge and domain-specific heuristics.

[3] //www.w3.org/TR/wsdl20/

4 SR-REST: A Matchmaking Approach for RESTful Services

We propose SR-REST[4] a matchmaking approach for RESTFul web services. Given a repository of web services and a user's request our aim is to automatically perform matching of this query with services as a means to find out a mapping indicating the services' elements corresponding to the underlying request. The result of the matching process is a set of mapping elements of both the schema of matched elements and the plausibility of their correspondence. The latter is defined using a similarity value between 0, i.e. strong dissimilarity; and 1, i.e strong similarity. The overall similarity between two services should be computed based on the degree of matching of their underlying operations.

The main idea of the approach consists in comparing the services with the requested service, i.e., operations, inputs and outputs, based on several SWS matchmaking strategies. Our aim is to take leverage of existing matchers [9] [13] such as, logic, syntactic, structural and semantic matchers, in providing suitable combination of these latters, leading to a flexible and efficient matchmaking algorithms. The approach is tested against RESTFul web services encoded in hRESTS. The proposed matchmaker, SR-REST, exploits information from all levels of the hRESTS Web service description in the matching process, i.e., service, operation and input/output levels.

4.1 Logic-Based Matchers

Usually, the inputs and the outputs are highly important in the service description due to the fact that they are considered as the only annotated part of the service enabling the use of the semantic relations between services' parameters. Therefore, SR-REST performs a logic matching based on the subsumption reasoning. It computes the logic match between inputs (respectively outputs) of the service offer and the service request using different logic filters proposed by Paolucci and al. [14]. Given two concepts C (from the service request) and D (from the service offer), the degrees of logic match (DoM) are defined as bellow:

- **exact:** if $C \equiv D$
- **plug-in:** if $C \sqsubseteq D$
- **subsume:** if $C \sqsupseteq C$
- **fail:** if C is disjoint with D

These filters allow us to exploit subsumption relations e.g. generalization, specification between concepts of an ontology. However, there are other semantic relations between concepts that could be useful in the matchmaking process (fig. 2) such as neighborhood relations. For that reason, we propose a new filter to detect such kind of relation called *intersect*. Assuming that we have two sets LG and LN of direct generalization and direct neighbors respectively, where:

[4] Service (Entity) Resolution RESTful web services

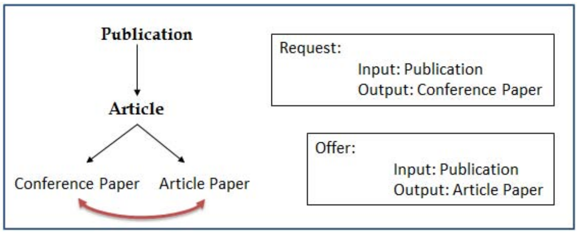


Fig. 2. Example of sibling relation between concepts

- **LG (C)**is the set of all direct generalization C' of the concept C, i.e. C is a sub concept of C'.

- **LN (C)**is the set of all direct neighbors C' of the concept C, i.e. LG (C) = LG (C')

- ∀(A, B) pair of concept, A *intersect* B, (denoted intersect(A, B)) if and only if ∃ a concept C where C ∈ LN (A) and C ∈ LN (B).

As a means to rank the previously mentioned filters, SR-REST attributes a quantitative equivalence (as in [5]) for each DoM relation based on the distance between concepts in the ontology. So, we mapped the pre-cited discrete DoMs into a constant numerical scale, varying from zero (no similarity) to one (exact similarity). Formally, we define PL(C,D) as the shortest path between two concepts C and D in an ontology and E the numerical equivalent for the DoM filter (exact, plugin, subsume, intersect, fail). Then, the logic similarity Ls(C,D), between the two concepts C and D is given by the following measure:

$$Ls(C, D) = E/PL(C, D). \tag{1}$$

4.2 Syntactic-Based Matchers

SR-REST can perform syntactic matching by computing the similarity between names, operation and descriptions of the service. For each pair of, service offer operation and request operation, it measures the similarity between texts based on functions proposed in simpack[15] such as: the Extended Jaccard , the Loss-of-Information , the Jensen-Shannon and the Cosine similarity measures.

Name Similarity. Names of services are defined by programmers. Usually, programmers respect some conventions, the names of variables and methods generally have meanings. A service name can summarize the functionality of the service. Using a tokenizer, some available similarity measure methods can be used to determine the similarity value between an offered service name and a requested service name, i.e., levensthein edit distance, average string, Dice's coefficient, Jaro coefficient, TF-IDF. SR-REST deploys some string based measurements described in Simpack [15] to compute the similarity value between service names.

The performance of names based matching depends on the qualities of service names. Note that, some service names describe perfectly the functionalities of their services while others cannot.

Text Similarity. Text description consists of a description of the web service written by developers in natural language using simple phrases and technical terms in an easily understandable way. Text description uses simple sentences instead of complicated ones leading to an easy process. Two similar services may use, in their text descriptions, similar terms, abbreviations, domain specific terminology, and phrases. Thus, it is possible to discover similar services by matchmaking their text descriptions. Hence, several web services matchmakers proposed in the literature [6] [16] consider text descriptions of Web services as a crucial information for the discovery process. These matchmakers use Tokenizer, they first convert words into lowercases, then use a porter stemmer for stemming and finally filter words using special characters, e.g. stop words, separators, etc. Text description of a hRESTS Web service document is represented by a vector where each dimension expresses a pair: the term and its frequency in the document. This matchmaker supports various vectors based on the similarity measurement functions implemented by Simpack.

4.3 Semantic Structural-Based Matcher

Semantic Structural based matchmakers are graph algorithms that consider the inputs as labeled graphs. They analyze the position of nodes in the graph. The importance of the structural similarities can be vindicated by the fact that same elements may appear in divers contexts and hence need to be differentiated to ensure a correct matching. In this way, SR-REST is also based on structural matching to determine the structural similarities between Input/Output elements. This similarity is based on the context elements and on the hypothesis that two elements are structurally similar if theirs structural neighbors are similar. The context includes: ancestor, sibling, immediate Child and leaf nodes.

The problem of semantic relatedness in an ontology or taxonomy is well known in the literature [17]. Generally, existing approaches represent the ontology as a graph for the semantic structural measure where nodes represent the different concepts and the edges represent relationships between concepts. However, the distance between concepts (graph nodes) can be used to compute the degree of their similarity. One of the most used approach for the semantic relatedness measure is Resnik [18], Resnik's measure function calculates the semantic similarity between ontology's concepts. It is based on the information content of a concept, which uses the instance probability. Using Resnik, different concepts in the ontology are annotated with values of the probability of their appearance in the service description.

$$sim_{Res}(A,B) = \max_{c \in S(A,B)} -logp(c) \tag{2}$$

where S(A,B) is the set of concepts that subsume both A and B in the ontology.

4.4 Structural-Based Matcher

A hRESTS Web service document can be given the structure of a labeled tree with the service at the root. Thereby, the problem of structural matching can be converted into a tree matching problem. Based on an XML schema, we create a subtree for each operation. The root of a subtree is the operation itself where the other nodes describe the inputs and outputs parameters. We use a WSDL Analyzer [19] to find the structural similarities between Input/output elements. This analyzer measures the similarities and the differences between two Web services. In addition, it exploits different types of data in the WSDL schema (names, type of information, structure). It proceeds first by identifying the similarities and differences over a set of operations, then by examining the structure of the Input/Output parameters. In the tree adopted depiction of a hRESTS description Web service document, the leaf nodes are the basic data types given by the XML schema.

– Definition (Service Tree): A labeled tree TS = (N,E,R,F) is an acyclic, connected graph where N = $\{n_1, n_2,..., n_p\}$ a set of nodes and E a set of edge. The function F : N $\longrightarrow$ L assigns a label to each node with basic data types DT $\in$ L (the set of labels) and R is the root of the tree TS.

The measurement of the similarity between two trees of services TS1 and TS2 begins from the root and recursively pass through the tree. For every node c $\in$ N_{TS1} and d $\in$ N_{TS2}.

$$Sim(c,d) = \begin{cases} wn * simn(F(c), F(d)) \\ +ws * Max\oplus i, jF(c), F(d) \notin D \\ simt(F(c), F(d))F(c), F(d) \in D \end{cases}$$

where (c, n_i)$\in$ E_{TS1} and (d,m_j)$\in$ E_{TS2}, $\bigoplus i, j$ is the sum of pairs sim(n_i,m_j) for $1 \leq i < card(n)$ and $1 \leq j < card(m)$, every n_i and m_j occur a once in the sum. If card(n) = card(m), there are some nodes that cannot be matched. Weights w_n and w_s are employed to reflect the effect of the structural or name similarities. simt is a function to measure the Similarity between types, it is based on the use of a compatibility table which assigns a degree to each pair of basic data types. The labels similarity, simn can be measured with different functions: edit distance, substring containment or similarity in WordNet (measure the semantic proximity).

4.5 SR-REST Algorithm

SR-REST takes into consideration all web service components (service, operation, inputs and outputs) during the matchmaking process. The underlying approach applies multiple matching strategies for each component: syntactic matching for service level and text description (when available) and operations names, logic and structural matching for inputs and outputs parameters. For this reason, we propose four different functions, Sim_{Ser} (Service similarity), Sim_{Op}

(operations similarity), Sim_{in} (inputs similarity), and Sim_{out}(outputs similarity). These similarities are combined using weights in an aggregated function *simA* defined as follows:

$$\mathrm{simA(a,b)} = \mathrm{sim}_{Ser}\mathrm{(a,b)} * \mathrm{w}_{Ser} + \mathrm{sim}_{Op}\mathrm{(a,b)} * \mathrm{w}_{Op} + \mathrm{sim}_{in}\mathrm{(a,b)} * \mathrm{w}_{in} + \mathrm{sim}_{out}\ \mathrm{(a,b)} * \mathrm{w}_{out}$$

$$\text{where } w_{Ser} + w_{Op} + w_{in} + w_{out} = 1 \quad (3)$$

```
program Service Matcher (Output)
  {
   var
     reqServ, offServ: Services
   begin
     Similarity(reqServ.ser,offServ.ser);

  ForEach (OpR[i] In reqServ.operations)
    { ForEach(OpO[j] in offServ.operations)
    {   simOp(i,j) = Similarity(OpR [i], OpO [j]);
        simIn(i,j) = matchParameters(OpR [i].inputs , OpO [j].inputs);
        simOut(i,j) = matchParameters( OpR [i].outputs , OpO [j].outputs);
        simA(i,j) = wSer * simSer(i,j) + wOp * simOp(i,j)
                    + wIn * simIn(i,j) + wOut * simOut(i,j);
        MSim(k)= simA(i,j);
        }
     }
      simg =SUM( MSim )/| reqService.operations|;
      return simg
    }
  }
end.
```

Fig. 3. *SR-REST Algorithm*

5 Evaluation

In this section, we present a preliminary evaluation of our approach. The hRESTS-TC1[5] test collection has been used to assess SR-REST Matchmaker. This collection is derived from SAWSDL-TC1 collection [10] and consists of 25 queries (from different domains: communication, food, economy, medical, travel and education), 895 services and 24 ontology used to semantically annotate the inputs and outputs parameters of web services. A relevance set is provided for each query, which is used to evaluate the precision of the matchmaker. SR-REST is implemented in Java and Pellet 2.0. We used SME2 the Semantic Matchmaker Evaluation Environment [6] to evaluate the performance of SR-REST.

[5] http://semwebcentral.org/projects/hRESTS-tc/
[6] http://projects.semwebcentral.org/projects/sme2/

For our evaluation, we use the Precision and Recall defined as follows:

Precision= $\frac{|A \cap B|}{B}$, Recall= $\frac{|A \cap B|}{A}$

where A is the set of all relevant services for a request and B the set of all retrieved services for a request.

SME2 evaluates the retrieval performance of matchmakers by measuring the average-precision (AP) and the macro-averaged precision[20]. Average precision AP is the average of precisions computed at each point of the relevant documents in the returned ranked list. However Macro-Average corresponds to the precision values for answer sets returned by the matchmaker for all queries in the test collection at standard recall levels. As the evaluation of a single query is not sufficient to make a significant observation, macro average precision is computed over many queries giving equal weight to each user query. Ceiling interpolation is used to estimate precision values at each standardized recall level, since each query likely has a different number of relevant services.

The Macro-Average precision is defined as follows:

$\text{Prec}_i = \frac{1}{Q} . \sum q \in Q \max\{P_o | R_o \geq Rec_i \wedge (R_o, P_o) \in O_q\}$

where O_q is the set of observed pairs of recall/precision values for query q when scanning the ranked services in the answer set for q stepwise for true positives in the relevance sets of the test collection.

5.1 Matcher Configuration

SR-REST combines different matchers: logic, syntactic and structural. We have performed different evaluation runs using several settings. Based on the work proposed in [10], we applied three configurations of level weights : interface, operations and parameters($w_{interface}$,$w_{operation}$,w_{inputs},$w_{outputs}$). The first configuration *config 1*(0, 0,0.5, 0.5) considers only inputs and outputs in the matching process. The second configuration *config 2* (0.25, 0.25, 0.25, 0.25) assigns the same weight to the different parameters. In the third configuration *config 3* (0.1, 0.1, 0.4, 0.4), we attribute a weight of 0.1 to the interface and operation and 0.4 to the inputs and outputs.

We assigned numerical equivalents for each DoM matching filter, exact=1, plugin=0.5, sub=0.5, intersect=0.25 and fail =0.

5.2 Experimentation

Table 1 summarizes the average precision (AP) for all previously described configurations of SR-REST. We assessed two versions of SR-REST. The first one without semantic structural match of parameters and the second one using Resnik for structural similarity measure.

Table 1. Averaged AP values

	config 1	config 2	Config 3
average.AP	0.70	0.73	0.75
	config 1+resnik	config 2 +resnik	Config 3+resnik
average.AP	0.71	0.754	0.77

We compare SR-REST with XAM4SWS [10] which is the only matchmaker to work on hRESTS services. SR-REST and XAM4SWS apply the same matchers: logic and syntatic ones. They attribute numerical equivalent for each DoM relation and aggregate the different measures in a global function. Unlike XAM4SWS, SR-REST deploys a semantic structural matching of input and outputs parameters using Resnik measure and considers the textual description of the service.

Table 2. SR-REST vs XAM4SWS Average Precision

Level weights	Num Dom Equivalents	AP SR-REST	AP XAM4SWS
config 1(0,0,0.5,0.5)	(1,0.5,0.5,0)	071	0.718
config 2 (0.25,0.25,0.25,0.25)	(1,0.5,0.5,0)	075	0.725
config 3(0.1,0.1,0.4,0.4)	(1,0.5,0.5,0)	0.77	0.747

5.3 Discussion

Based on the experimental results, we can make the following remarks:

– We used three different configurations to evaluate the performance of SR-REST approach. As illustrated in Fig. 4, in taking into account the service and operation levels in the matchmaking process, we did improve both precision and recall. This can be explained by the fact that, in some cases, a service name can reflect the functionality of the service and thus some pertinent services can be discovered while taking into account the name and the textual description of the service.
– Semantic structural matching (Resnik) improves the precision and recall of SR-REST (Fig. 5). This matcher considers the different relations between concepts of the ontology. So, indirect subsumption relations between concepts can be identified and consequently some relevant results can be retrieved.
– The structure of hRESTS documents may play an important role in the matchmaking process. In some cases, false positives and false negatives can be caused by the difference in the cardinality of input and output parameters sets. Then, we used WSDL-Analyzer to determine structural matching of hRESTS documents based on a labeled tree matching. In fact a hRESTS document can be considered as a WSDL file after elimination of its concretes parts (binding, ports, methods).

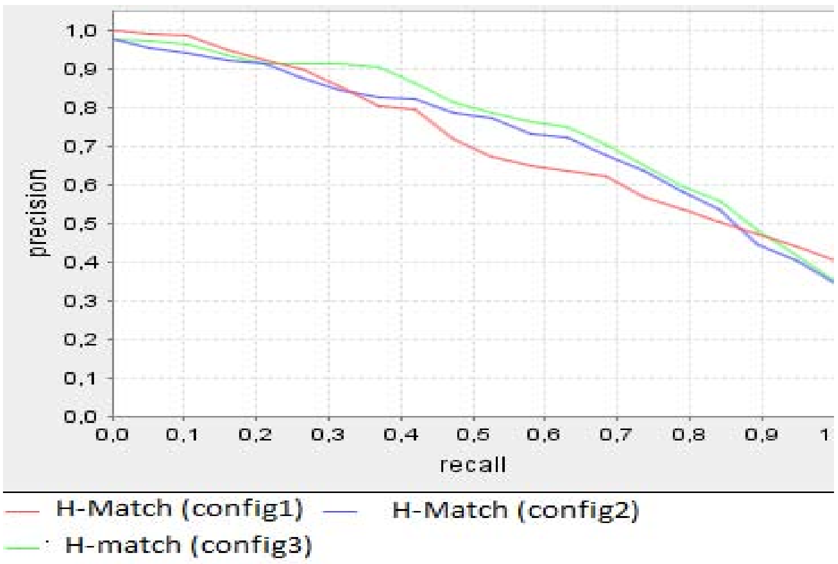


Fig. 4. Recall/Precision Macro Averaged

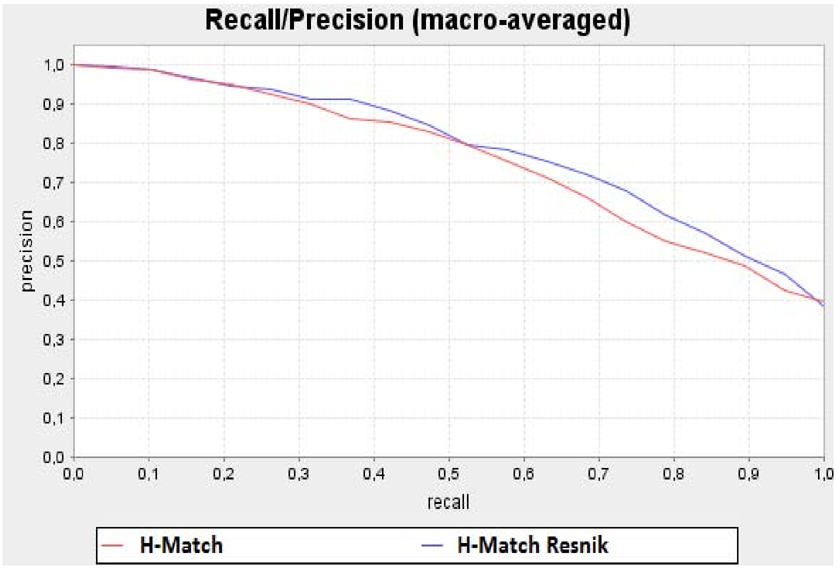


Fig. 5. Recall/Precision Macro Averaged(config3)

– Matcher tuning is a hard task and is highly dependent on the collection tests. While the combination of logic, syntactic, structural and semantic matchers yield to good results, an improvement of these results can be done in using other matchers or techniques. Most of the existing matchmakers use machine learning or approximate matching techniques to this aim. Then, we believe that we can improve SR-REST in considering more features like synonym relations and in applying an approximative logic matching of annotated parameters to compute the approximated concept subsumption relations and the corresponding approximated signature matching. In addition a learning machine approach can be used to aggregate results of different matchers. As in XAM4SWS, we could use an estimator such as OLS (Ordinary Least Squares) that maps the logic filter to its numerical counterpart.

6 Conclusion

In this paper, we described SR-REST a flexible matchmaking approach that allows the combination of several (schema) matching/matchmaking techniques to achieve semantic web service discovery. The approach has been implemented and successfully tested against a collection of RESTFul web services described in hRESTS.

Although we used the hRESTS description of web services, SR-REST is generic enough to be applied for the discovery of other web resources (e.g., Cloud APIs) that maybe (more or less) semantically annotated.

Finally, applicability and importance of the approach could be also discussed with respect to the availability of a significant number of hRESTS web services. One should notice, that an important number of REST services are being made available (see *programmableweb* website[7] for instance). Hence, assuming one can (semi)automatically annotate those REST APIs, the SR-REST approach described in this paper could be tested and evaluated against REST services listed in the *programmableweb* directory.

References

1. Bellahsene, Z., Bonifati, A., Rahm, E. (eds.): Schema Matching and Mapping. Springer (2011)
2. Algergawy, A., Nayak, R., Siegmund, N., Köppen, V., Saake, G.: Combining schema and level-based matching for web service discovery. In: Benatallah, B., Casati, F., Kappel, G., Rossi, G. (eds.) ICWE 2010. LNCS, vol. 6189, pp. 114–128. Springer, Heidelberg (2010)
3. Kopecký, J., Gomadam, K., Vitvar, T.: hRESTS: An HTML microformat for describing RESTful web services. In: Web Intelligence, pp. 619–625. IEEE (2008)
4. Klusch, M., Kapahnke, P., Zinnikus, I.: Adaptive hybrid semantic selection of sawsdl services with sawsdl-mx2. Int. J. Semantic Web Inf. Syst. (2010)
5. Schulte, S., Lampe, U., Eckert, J., Steinmetz, R.: Log4sws.kom: Self-adapting semantic web service discovery for sawsdl. In: The 6th World Congress on Services, SERVICES 2010, Miami, Florida, USA, July 5-10, pp. 511–518. IEEE Computer Society (2010)

[7] `www.programmableweb.com`

6. Wei, D., Wang, T., Wang, J., Bernstein, A.: Sawsdl-imatcher: A customizable and effective semantic web service matchmaker. Web Semantics: Science, Services and Agents on the World Wide Web (2011)
7. Klusch, M., Kapahnke, P.: Adaptive signature-based semantic selection of services with owls-mx3. Multiagent and Grid Systems 8(1), 69–82 (2012)
8. Klusch, M., Kapahnke, P.: iSeM: Approximated reasoning for adaptive hybrid selection of semantic services. In: Aroyo, L., Antoniou, G., Hyvönen, E., ten Teije, A., Stuckenschmidt, H., Cabral, L., Tudorache, T. (eds.) ESWC 2010, Part II. LNCS, vol. 6089, pp. 30–44. Springer, Heidelberg (2010)
9. Klusch, M.: Semantic web service coordination. In: CASCOM: Intelligent Service Coordination in the Semantic Web, pp. 59–104 (2008)
10. Lampe, U., Schulte, S., Siebenhaar, M., Schuller, D., Steinmetz, R.: Adaptive matchmaking for restful services based on hrests and microwsmo. In: Proceedings of the 5th Workshop on Emerging Web Services Technology, WEWST 2010, Ayia Napa, Cyprus, December 1. ACM International Conference Proceeding Series, pp. 10–17. ACM (2010)
11. Khorasgani, R.R., Stroulia, E., Zaïane, O.R.: Web service matching for restful web services. In: 13th IEEE International Symposium on Web Systems Evolution, WSE 2011, Williamsburg, VA, USA, September 30, pp. 115–124. IEEE (2011)
12. John, D., Rajasree, M.S.: Restdoc: Describe, discover and compose restful semantic web services using annotated documentations. International Journal of Web and Semantic Technology (IJWseT) 4(1) (2013)
13. Shvaiko, P., Euzenat, J.: A survey of schema-based matching approaches. In: Spaccapietra, S. (ed.) Journal on Data Semantics IV. LNCS, vol. 3730, pp. 146–171. Springer, Heidelberg (2005)
14. Paolucci, M., Kawamura, T., Payne, T.R., Sycara, K.: Semantic matching of web services capabilities. In: Horrocks, I., Hendler, J. (eds.) ISWC 2002. LNCS, vol. 2342, pp. 333–347. Springer, Heidelberg (2002)
15. Bernstein, A., Kaufmann, E., Kiefer, C., Bürki, C.: SimPack: A generic Java library for similiarity measures in ontologies. Technical report, Department of Informatics, University of Zurich, Zurich, Switzerland (2005)
16. Becker, J., Müller, O., Woditsch, M.: An ontology-based natural language service discovery engine - design and experimental evaluation. In: Alexander, P.M., Turpin, M., van Deventer, J.P. (eds.) ECIS (2010)
17. Patwardhan, S., Banerjee, S., Pedersen, T.: Using measures of semantic relatedness for word sense disambiguation. In: Gelbukh, A. (ed.) CICLing 2003. LNCS, vol. 2588, pp. 241–257. Springer, Heidelberg (2003)
18. Resnik, P.: Semantic similarity in a taxonomy: An information-based measure and its application to problems of ambiguity in natural language. Journal of Artificial Intelligence Research 11(1), 95–130 (1999)
19. Klusch, M., Kapahnke, P., Zinnikus, I.: Hybrid adaptive web service selection with SAWSDL-MX and WSDL-analyzer. In: Aroyo, L., et al. (eds.) ESWC 2009. LNCS, vol. 5554, pp. 550–564. Springer, Heidelberg (2009)
20. Raghavan, V.V., Bollmann, P., Jung, G.S.: Retrieval system evaluation using recall and precision: Problems and answers. In: Proceedings of the 12th International Conference on Research and Development in Information Retrieval, SIGIR 1989, Cambridge, Massachusetts, USA, June 25-28, pp. 59–68 (1989)

Mix'n'Match: An Alternative Approach for Combining Ontology Matchers

(Short Paper)

Simon Steyskal[1,2] and Axel Polleres[1]

[1] Siemens AG, Siemensstrasse 90, 1210 Vienna, Austria
[2] Vienna University of Technology, Favoritenstraße 9, 1040 Vienna, Austria

Abstract. The existence of a standardized ontology alignment format promoted by the Ontology Alignment Evaluation Initiative (OAEI) potentially enables different ontology matchers to be combined and used together. Along these lines, we present a novel architecture for combining ontology matchers based on iterative calls of off-the-shelf matchers that exchange information in the form of reference mappings in this standard alignment format. However, we argue that only a few of the matchers contesting in the past years' OAEI campaigns actually allow the provision of reference alignments to support the matching process. We bypass this lacking functionality by introducing an alternative approach for aligning results of different ontology matchers using simple URI replacement in the aligned ontologies. We experimentally prove that our iterative approach benefits from this emulation of reference alignments.

1 Introduction

Mapping between the internal models of software systems is a crucial task in almost all data and system integration scenarios, keeping computer scientists busy for the last decades with still no "silver bullet" in sight.

In quest for the said "silver bullet" [9] to solve such alignment problems, ontologies and ontology matching seem to be a good starting point particularly for for aligning object-oriented models, since ontologies can represent (and be extracted from) object-oriented models fairly naturally, where the hope is that the increasing number of off-the-shelf ontology matching tools can be used "out of the box" to propose reference alignments that can be tailored in a semi-automatic process. In the past decade, the field of ontology matching has matured with many different ontology matchers having participated in the last years' OAEI [1] campaigns, exposing different strengths and weaknesses. The different tracks provided by the OAEI target different matching problems and since every matching tool has its specific special techniques and features, good performance in a specific sub-track often comes in hand with a bad performance in another one. So, intuitively, combining the results of a heterogeneous set of ontology matchers,

[1] `http://oaei.ontologymatching.org/`

taking the “best off-the-shelf matcher for the problem at hand” seems to be a promising approach for a “universal matcher” without the need for designing a new matcher from scratch.

Unfortunately, (i) neither deciding which matcher is best suitable for a set of given ontologies nor (ii) using ontology matchers in a semi-automatic human-guided matching process is trivial: for one, selecting the most suitable matcher requires some preparatory work like interviews or tests [18] or background knowledge in terms of training sets for learning a classifier [7, 8].

On the other hand, for an interactive matching process, one needs to be able to guide a matcher by providing (or confirming) reference alignments, either provided by a domain expert directly or iteratively added by confirming matching results from an ontology matcher to be used as reference for supporting another call of the same or another ontology matcher.

In the present paper, we demonstrate the feasibility of a system that can support such a process in one go; that is, we present a combined ontology matcher which performs better on average than a single matcher on a number of heterogeneous ontology matching problems (from the OAEI campaign), without the need to choose a single matcher, but by iteratively combining results of different matchers, and feeding them as support into subsequent runs of those matchers.

Structure of the Paper. We start our paper with section 2 presenting a framework, which defines an architecture for a combined iterative matcher, that allows to plug existing matchers flexibly together and iteratively combines their results. Preliminary evaluation results presented in Section 3 are very encouraging showing that our approach performs very stable, outperform each single matcher in some cases, or scoring comparably well to the best single matchers in other cases. We discuss ideas for future improvements in Section 4 before we conclude in Section 5.

2 The Mix’n’Match Framework

In this section, we present our combined approach to ontology matching, which we call “Mix’n’Match”. Our goal is an approach that combines the above existing matchers with all their strengths and weaknesses in a unified manner. Instead of finding the one “best off-the-shelf matcher for the problem at hand” we aim at combining matcher results of different matchers. Intuitively, the idea of Mix’n’Match is very simple: starting from an empty set of alignments, we aim at iteratively supporting in each round, matchers with the combined results of other matchers found in previous rounds, somehow aggregating the results of a heterogeneous set of ontology matchers.

Two main obstacles need to be overcome to make this approach feasible.

Result Aggregation: In each round we need to decide which alignments to keep and which ones to ignore from the union of new alignments found by all considered matchers.

Reference Alignments: as mentioned above only one of the considered ontology matchers supports reference alignments as inputs, so we need to find another way to "inject" alignments in a non-obtrusive way[2] into existing matchers.

As for result aggregation, for the moment, we have chosen a straightforward strategy here, based on a simple majority vote, that is, all alignments confirmed by a majority of matchers above a certain threshold prevail. Adding support for reference alignments (without touching the matchers' source code) was not that easy. Our first idea was to add reference alignments explicitly in the form of OWL equivalences, using properties `owl:equivalentClass` (for matching concepts), `owl:equivalentProperty` (for matching properties), or `owl:sameAs` (for matching individuals). However, this approach to "emulating" reference alignments proved unsatisfactory, since we would be referring in $O1$ to entities from $O2$ and vice versa. First, such referring would need additional definitions (`owl:Class`, `owl:DatatypeProperty`, or `owl:ObjectProperty` of the referred entities in the respective other ontology of all aligned entities, since otherwise the such enriched ontology would no longer satisfy the syntactic restrictions of OWL DL: obviously, this is inconvenient, because it would lead to significant overhead essentially redefining already existing entities. Somewhat surprisingly, an alternative, much simpler approach to emulating reference alignments proved to be much more effective than adding OWL equivalence axioms: instead of "axiomatizing" alignments using OWL, we just created for each reference alignment a combined new URI for the matched entities, which was then replaced within both $O1$ and $O2$. Let us illustrate this replacement in a short example.

Example 1. Based on found alignments we modify both ontologies by replacing the URIs of the two classes *Teacher* and *Professor* by a unified one <*http://example.org/MixMatch/Teacher___Professor*>, cf. Listings 1+2.

Listing 1. Ontology $O1$ after enrichment

```
@prefix mm: <ex.org/MMatch
    />.
...
mm:Teacher_Professor a owl:
    Class.

ont1:name a owl:
    DatatypeProperty;
     rdfs:domain mm:
         Teacher_Professor;
         rdfs:range xsd:string
...
```

Listing 2. Ontology $O2$ after enrichment

```
@prefix mm: <ex.org/MMatch
    />.
...
mm:Teacher_Professor a owl:
    Class.

ont2:nameIs a owl:
    DatatypeProperty;
     rdfs:domain mm:
         Teacher_Professor;
         rdfs:range xsd:string
...
```

A schematic overview of the overall Mix'n'Match framework is shown in Figure 1; we briefly explain each step in the following.

[2] In order to keep our framework general and extensible, we do not want to modify the code of the ontology matchers or interfere in some other tool-specific way with the matchers used in our framework.

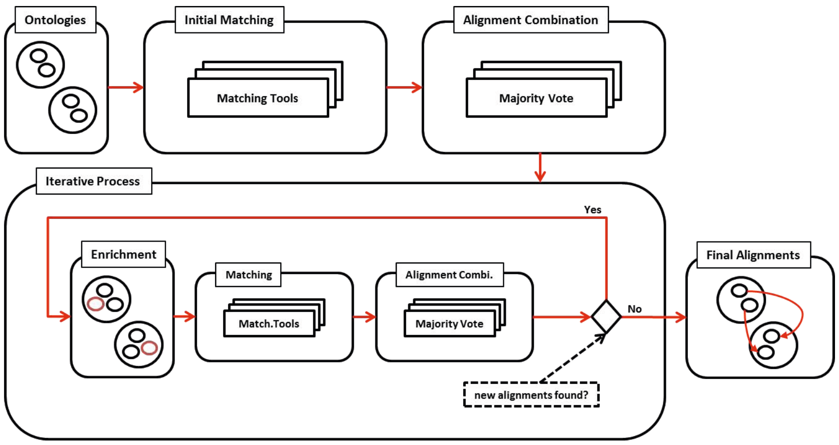


Fig. 1. Framework of Mix'n'Match

Initial Matching: In the initial matching step all participating ontology matchers try to find alignments of the two unaltered base ontologies.

Alignment Combination: The combination of the alignments, especially the choice of those which are used for the enrichment step is – as mentioned above – based on majority votes. By only accepting alignments which were found by a large number of heterogeneous matching tools we aim to ensure a high precision of the found alignments and therefore emulate reference alignments provided by a human domain expert. Although Mix'n'Match would support the definition of an alignment confidence threshold as additional parameter (i.e, only allowing alignments over a specific threshold to pass) we set this threshold per default to 0 in our experiments: since the calculation of confidence values is not standardised across matchers and some matchers only produce boolean confidence values. Other result aggregation methods may be conceivable here and will need more detailed investigations in future versions of Mix'n'Match.

Enrichment: After mixing of the alignments, enrichment of the ontologies takes place, implementing the simple URI replacement sketched above: for every pair of matched entities in the set of aggregated alignments, a merged entity URI is created and will replace every occurrence of the matched entities in both ontologies (cf. Example 1). This approach is motivated by the assumption that if two entities were stated as equal by the majority of ontology matchers, their URI can be replaced by an unified URI, stating them as equal in the sense of URIs as global identifiers. Note that, despite the above-mentioned fact that most matchers seem to ignore URIs as unique identifiers of entities, our experiments showed that URI replacement was effective in boosting the confidence in such asserted alignments in almost all considered matchers.[3] Again, other enrichment techniques – as mentioned before, for instance, for matchers supporting OWL, reference alignments could be encoded by enrichment with respective OWL axioms – could be conceivable and would fit into the generic framework of Mix'n'Match. Note also that in the current version of Mix'n'Match only one-to-one and equality alignments were considered.

Iteration: After enrichment (which imitates the behavior of reference alignments) we start a new matching round, i.e., we restart the individual matchers on the enriched ontologies, gathering new reference alignments, followed by another alignment combination step, and so forth, until a fix point is reached, in the sense that the alignment combination step produces no further new alignments.

Implementation: We have implemented Mix'n'Match using Java for the general framework, and integrated the existing matchers either by importing the respective Java sources, if available, or including .jar files (some of which were available through the SEAL[4] project).

3 Evaluation

We evaluated our framework using datasets provided by the ontology alignment evaluation initiative. In the following sections we show, that our approach is either able to outperform current ontology matching tools or enhance the results of the in the Mix'n'Match process participated ones.

As evaluation system we used an Intel Core Duo 3GHz with 4GB RAM and Windows 7 64 Bit as OS.

3.1 Evaluation Datasets

Our initial intention was to compare Mix'n'Matchs results to those retrieved by a machine learning approach of combining ontology matchers. [7] Therefore we based our evaluations on the same datasets, i.e. #301 to #304 of the benchmark track consisting of four real world ontologies which were matched against a base ontology (#101), all describing a bibliographic domain and using a subset of the OAEIs conference track, but in contrast to the evaluation of the machine learning approach we used the updated reference alignment set *ra1* which consists of seven ontologies namely *cmt, conference, ekaw, edas, conference, iasted* and *sigkdd*. In contrast to the benchmark track, all seven ontologies were matched against each other resulting in 21 possible combinations and for which reference alignments were available. Regardless their respective participation on either the benchmark or the conference track, we included *AROMA [5], AnchorFlood [11], Eff2Match [1], FalconAO [14], HotMatch [4]* and *Hertuda [12]* for the

[3] Deliberately leaving out internal details of the respective matchers, we may only conjecture here that other features like string-matching were triggered after reference alignments being "asserted" by URI replacement, which consequently lead to further new alignments being found in subsequent calls.

[4] `http://www.seals-project.eu/`

final Mix'n'Match process. Based on performance issues we excluded *Lily [21]* from our matching process and therefore were able to decrease the computation time of Mix'n'Match about more than 50%, two matching tools where excluded because of compatibility problems (*AUTOMSv2 [16], ServOMap2 [22]*). To prove the robustness of our approach, *LogMap2 [15] and YAM++ [19]* were evaluated for comparison purposes but didn't participate in the Mix'n'Match process.

3.2 Evaluation Results

Regarding the Benchmark track, Mix'n'Match was able to outperform all tested ontology matchers although it needed more than twice as much time to complete than the second slowest matcher *Lily*. Matchers marked with † were not able to retrieve alignments for testcase 303 in our test setting. As stated in Table 1 Mix'n'Match was able to increase the F-Measure value of the best used matcher by 2.72%. Nevertheless LogMap2 and Yam++ retrieved better results, but since our approach is primarily based on using majority vote for choosing alignments, single outstanding ontology matchers don't significantly affect the results if we considering their alignments for Mix'n'Match.

Table 1. Aggregated results of the Benchmark and Conference Tracks

	Benchmark			**Conference**		
Matcher	Prec./Rec.(%)	F-Meas.(%)	Time(ms)	Prec./Rec.(%)	F-Meas.(%)	Time(ms)
Mix'n'Match	**89,70/76,40**	**82,51**	**229795**	**68,78/57,50**	**62,64**	**1215747**
Eff2Match	86,97/74,26	80,12	35123	34,43/64,59	44,91	90649
FalconAO	87,24/73,42	79,73	10424	56,32/57,46	56,89	10176
YAM++	89,86/70,01	78,70	51057	77,29/68,95	72,88	394041
Anchor-Flood†	68,86/57,46	62,64	1021	45,16/57,73	50,68	2451
AUTOv2†	68,50/48,40	56,73	12751	76,90/45,13	56,88	30654
Aroma†	59,06/50,15	54,24	1723	30,85/45,97	36,92	5667
Lily†	54,57/51,68	53,09	86390	35,72/46,03	40,23	337861
Hertuda	56,95/42,90	48,94	1077	72,62/51,01	59,92	1953
LogMap2	83,62/ 30,41	44,60	7255	78,81/57,91	66,76	22072
HotMatch	61,58/33,94	43,76	2553	69,66/51,60	59,29	8994

Remark. We recognized slight differences between the evaluation values measured by the OAEI and those measured by us. But since our approach relies on the results of other matchers, the relative improvement of the values should stay the same.

4 Discussion

Related Work: The approach of using an iterative process which reruns matching techniques till no further alignment pairs were found is not new per se. Tools like Anchor-Flood [11] or ASMOV [13] are based on such a framework but in contrast to our approach neither one of them use off-the-shelf matchers for retrieving alignments. The benefit of using whole tools instead of specific matching techniques is obvious, on the one hand we can highly increase the flexibility of Mix'n'Match since single off-the-shelf matcher can easily be integrated and segregated from the matching process and on the other hand since matching tools evolve over time and perform better and better by themselves, Mix'n'Match will also benefit from this evolution based on its framework. Besides of combining ontology matchers during the matching step, some approaches only focus on the combination of alignment sets of already matched ontologies. But these approaches doesn't rerun the matching process with the gathered additional knowledge again, although they received very good results especially by using machine learning techniques for the combination of the alignments [6,7,17]. Furthermore there exist some other approaches than a majority voite for combining the results of different ontology matching techniques which could be fairly easily adopted to work with our approach like in [10] where the authors propose an automated approach for weighting individual ontology matchers based on their importance on different matching tasks or in [2] were an automatic alignment evaluation by using quality measures is described. In [3] the authors provide an approach for automatically configuring the parameters of off-the-shelf matchers and therefore optimize their performance; together with the approach proposed in [20] where unsupervised learning techniques are used to find similarity parameters these approaches could be used to improve the performance and flexibility of Mix'n'Match.

Future Work: Several topics will be part of future investigations like using other combination approaches than majority vote or the distributed execution of the used ontology matchers. As alternative combination approach, an advanced machine learning technique like discussed in [7] could be used. Therefore we could increase the *recall* of the reference alignments generated in each round and also remove the common drawbacks of majority votes like ignoring single good results. Furthermore a self-learning classifier must be developed to ensure the full automation of our approach.

5 Conclusions

In this paper we have presented an alternative approach for combining ontology matchers, using their alignment results for iteratively including the knowledge these alignments provide into the ontologies and then rerun the matching process. We have shown that it is basically possible to emulate the additional knowledge eventual input alignments could provide and furthermore eliminate the common drawbacks like loss of automation, which usually comes in hand with the use of reference alignments as support for matching processes.

Nevertheless we discovered that not all ontology matchers consider entities with the exact same URI as 100% identical, but since string similarity measures seems to be part of nearly every investigated ontology matcher it may helped to exceed some internal confidence plateaus of found alignments and therefore accept them as correct alignments. We also pointed out that the more information for the matching process is available, either in terms of additional axioms in the ontology or by providing reference alignments, the better results can be obtained.

References

1. Wei Khong Chua, W., Kim, J.-J.: Eff2match results for oaei 2010. In: OM 2010. CEUR Workshop Proceedings (2010)
2. Cruz, I., Palandri Antonelli, F., Stroe, C.: Efficient selection of mappings and automatic quality-driven combination of matching methods. In: OM 2009. CEUR Workshop Proceedings (2009)
3. Cruz, I.F., Fabiani, A., Caimi, F., Stroe, C., Palmonari, M.: Automatic configuration selection using ontology matching task profiling. In: Simperl, E., Cimiano, P., Polleres, A., Corcho, O., Presutti, V. (eds.) ESWC 2012. LNCS, vol. 7295, pp. 179–194. Springer, Heidelberg (2012)
4. Dang, T.T., et al.: Hotmatch results for oeai 2012. In: OM 2012. CEUR Workshop Proceedings (2012)
5. David, J., Guillet, F., Briand, H.: Matching directories and owl ontologies with aroma. In: CIKM 2006, pp. 830–831. ACM, New York (2006)
6. Doan, A., Madhavan, J., Dhamankar, R., Domingos, P., Halevy, A.Y.: Learning to match ontologies on the semantic web. VLDB J. 12(4), 303–319 (2003)
7. Eckert, K., Meilicke, C., Stuckenschmidt, H.: Improving ontology matching using meta-level learning. In: Aroyo, L., et al. (eds.) ESWC 2009. LNCS, vol. 5554, pp. 158–172. Springer, Heidelberg (2009)
8. Ehrig, M., Staab, S., Sure, Y.: Bootstrapping ontology alignment methods with APFEL. In: Gil, Y., Motta, E., Benjamins, V.R., Musen, M.A. (eds.) ISWC 2005. LNCS, vol. 3729, pp. 186–200. Springer, Heidelberg (2005)
9. Fensel, D.: Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce. Springer (2001)
10. Gulić, M., Magdalenić, I., Vrdoljak, B.: Automated weighted aggregation in an ontology matching system. Int'l Journal of Metadata, Semantics and Ontologies 7(1), 55–64 (2012)
11. Seddiqui Hanif, M., Aono, M.: An efficient and scalable algorithm for segmented alignment of ontologies of arbitrary size. J. Web Sem. 7(4), 344–356 (2009)
12. Hertling, S.: Hertuda results for oeai 2012. In: OM 2012. CEUR Workshop Proceedings (2012)
13. Jean-Mary, Y.R., Shironoshita, E.P., Kabuka, M.R.: Ontology Matching with Semantic Verification. J. Web Sem. 7(3), 235–251 (2009)
14. Jian, N., Hu, W., Cheng, G., Qu, Y.: Falcon-ao: Aligning ontologies with falcon. In: K-Cap 2005 Workshop on Integrating Ontologies, pp. 87–93 (2005)
15. Jiménez-Ruiz, E., Cuenca Grau, B., Zhou, Y.: Logmap 2.0: towards logic-based, scalable and interactive ontology matching. In: 4th Int'l Workshop on Semantic Web Applications and Tools for the Life Sciences, SWAT4LS 2011, pp. 45–46. ACM (2012)
16. Kotis, K., Katasonov, A., Leino, J.: Automsv2 results for oaei 2012. In: OM 2012. CEUR Workshop Proceedings (2012)
17. Maio, P., Bettencourt, N., Silva, N., Rocha, J.: Evaluating a confidence value for ontology alignment. In: Proceedings of the 2nd International Workshop on Ontology Matching (OM 2007), Busan, Korea, November 11. CEUR Workshop Proceedings, vol. 304. CEUR-WS.org (2007)
18. Mochol, M., Jentzsch, A.: Towards a rule-based matcher selection. In: Gangemi, A., Euzenat, J. (eds.) EKAW 2008. LNCS (LNAI), vol. 5268, pp. 109–119. Springer, Heidelberg (2008)

19. Ngo, D.H., Bellahsene, Z.: YAM++: (not) Yet Another Matcher for Ontology Matching Task. In: Bases de Données Avancées, France, p. 5 (2012)
20. Nikolov, A., d'Aquin, M., Motta, E.: Unsupervised learning of link discovery configuration. In: Simperl, E., Cimiano, P., Polleres, A., Corcho, O., Presutti, V. (eds.) ESWC 2012. LNCS, vol. 7295, pp. 119–133. Springer, Heidelberg (2012)
21. Wang, P., Xu, B.: Lily: Ontology alignment results for oaei 2009. In: OM 2009. CEUR Workshop Proceedings (2009)
22. Wang, Z., Wang, Y., Zhang, S., Shen, G., Du, T.: Matching large scale ontology effectively. In: Mizoguchi, R., Shi, Z.-Z., Giunchiglia, F. (eds.) ASWC 2006. LNCS, vol. 4185, pp. 99–105. Springer, Heidelberg (2006)

Evaluation of Technologies for Mapping Representation in Ontologies

Olga Kovalenko[1], Christophe Debruyne[2], Estefanía Serral[1], and Stefan Biffl[1]

[1] Vienna University of Technology, CDL-Flex,
Favoritenstrasse 9-11/E188, A-1040 Vienna
`firstname.lastname@tuwien.ac.at`
[2] Vrije Universiteit Brussel STARLab, Pleinlaan 2, B-1050 Brussel
`firstname.lastname@vub.ac.be`

Abstract. Ontology mapping is needed to explicitly represent the relations between several ontologies, which is an essential task for applications such as semantic integration and data transformation. Currently, there is no standard for representing mappings. Instead, there are a number of technologies that support the representation of mappings between the ontologies. In this paper we introduce a set of mapping categories that were identified based on requirements for the data integration projects of an industry partner. An evaluation of available technologies for mapping representation regarding the support for introduced mapping categories has been performed. The results of the evaluation show that the SPARQL Inference Notation would fit the best in the described use case scenario.

1 Introduction

Ontologies are widely used to obtain semantic integration and semantic interoperability among different systems. For many applications it is necessary to explicitly specify the relations and correspondences between different ontologies. The common technique here is to define a set of mappings that bind a set of entities in one ontology to the entities in another ontology.

To our knowledge there is no standard or conventional technique on how to represent mappings in ontologies. The Web Ontology Language[1] (OWL) provides some support to express mappings between the entities of different ontologies [5]. These mapping are stored as a part of the ontology and are coupled with it [10]. However, the expressivity of OWL is restricted to rather simple mappings, such as one-to-one mapping for ontology concepts and properties. Differences in granularity or understanding of the domain semantics require support for more sophisticated mappings and cannot be expressed in pure OWL [5]. In order to support defining more complex mappings between ontologies various technologies have been developed, e.g., the Datalog language, the Semantic Web Rule Language (SWRL)[2], SPARQL CONSTRUCT queries[3], and the SPARQL

[1] OWL overview: `http://www.w3.org/TR/owl-features/`
[2] SWRL specification: `http://www.daml.org/2003/11/swrl/`
[3] SPARQL specification: `http://www.w3.org/TR/rdf-sparql-query/`

Inference Notation (SPIN)[4]. The degree of support for defining mappings between ontologies provided by these technologies vary in many aspects, e.g., by the expressivity for mapping representation and on how mappings are stored, which must be taken into account for choosing the appropriate technology for a specific task. It is important to note that the selection of an optimal technique for mapping representation will strongly depend on the application at hand, e.g., the best fitting technology for ontology merging or for semantic integration can be different [11].

In this paper we create a benchmark for comparing the abilities of technologies to express different categories of mappings. We investigate categories of complex mappings between ontologies needed in a typical data integration scenario in multi-disciplinary engineering projects [1]. In these projects the data from the ontologies of the disciplines involved in the project must be transformed into a common ontology to allow the integration of the local discipline ontologies. The contribution of the paper is twofold: a) we categorize mappings that are required in the projects of an industry partner to fulfill effective and efficient data integration and data transformation; and b) we evaluate the available technologies for mapping representation regarding the support provided for introduced mapping categories. We focus on OWL and RDF(S)[5] ontologies, as these are currently prevailing ontology languages.

The rest of the paper is organized as follows: Section 2 describes the related work on ontology mapping; in Section 3 we present the application scenario from our industry partners that is used across the paper to illustrate introduced mapping categories and determine a set of mapping categories that may be needed in the described use case scenario; and in Section 4 we evaluate the existing technologies for mapping representation regarding the support for defined mapping categories and discuss which technology would be a better choice in the presented use case scenario.

2 Related Work

Mapping captures the semantic and structural relationships between the entities of two ontologies. Or in other words, mappings specify the connection between entities in enough details that it will be possible to apply or execute them for a certain task [3].

According to Noy et al three dimensions can be distinguished in ontology mapping research: a) mappings discovery; b) mappings representation (MR) and c) reasoning with mappings [9]. In this paper we focus on the second dimension - representation of mappings.

Several authors have addressed the problem of MR in their works. Noy at al. distinguish three main ways on how mappings can be represented: a) as first-order logic axioms serving as semantic bridges between ontologies; b) using views

[4] SPIN overview: `http://www.w3.org/Submission/spin-overview/`

[5] RDF(S) specification: `http://www.w3.org/TR/rdf-schema/`

that link global ontology to local ontologies; and c) mappings themselves are instances in a mappings ontology [9]. An approach using the Semantic Bridges between entities of two ontologies was presented by Maedche et al. [8]. A language to specify mappings between ontologies and requirements for such language are described by Scharffe and de Bruijn in [10]. An interesting approach was presented by Brockmans et al. [2], which is based on Meta Object Facility (MOF)[6] and the Unified Modeling Language (UML)[7] to support formalism independent graphical modeling of mappings between OWL ontologies.

The general problem is that there are many systems developed by different vendors, which represent mappings in dissimilar formats. A standard language for this purpose would highly facilitate the reusability and interoperability of the results of such systems, however there is no such single standard established yet [10]. Up to date there is also a lack of work investigating the abilities and appropriateness of current technologies for MR in different application scenarios. We believe that this problem has high practical importance as the specific application will influence the choice of optimal MR technique for the task at hand [11]. Focused on the data integration and data transformation applications, this paper provides a state of the art on existing technologies for MR between ontologies, and evaluates their support for a set of essential mapping categories.

3 Use Case Scenario and Mappings Description

As a running example we present a data integration scenario from an industry partner, a hydro power plant systems integrator.

As is shown on the Figure 1 ontologies are applied to integrate data from all disciplines involved in the project engineering. The data model of each discipline is represented by its local ontology. The common data model, which includes only concepts that are relevant to at least two disciplines, is built on the top of local ontologies. For simplicity reasons we limit the set of concepts and attributes shown to the minimal set necessary to illustrate all the identified mapping categories introduced further. On the top of the figure 1, the local ontologies representing data models from three different domains (software engineering (SE), mechanical engineering (ME) and project management (PM)) are shown. The concept *PLC* describes the controllers' behavior of the devices; the concept *PhysicalComponent* describes the components of the plant, and the concept *Connection* captures direct connections between the devices; the concept *Project* describes the important characteristics of the project under development. The figure 1 also shows relations between each local model and the common model represented by mappings. Different mapping categories are identified as M1-M7.

Value Processing (M1-M4). Often a relation between entities in two ontologies is not "exactly-the-same", but can be represented by some function that

[6] OMG's MetaObject Facility specification: `http://www.omg.org/mof/`

[7] Unified Modeling Language specification: `http://www.omg.org/spec/UML/`

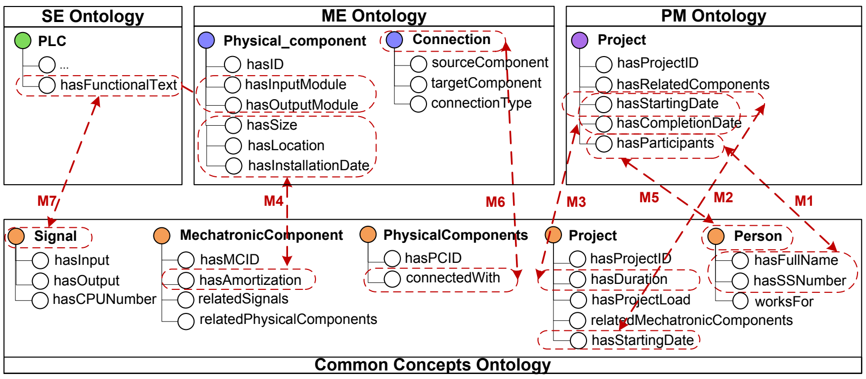


Fig. 1. Mapping categories: M1 - string processing; M2 - data type transformation; M3 - math functions; M4 - user-defined functions; M5 - structural differences → granularity; M6 - structural differences → schematic differences; M7 - conditional mappings. Mappings M1-M2 could be also considered as bidirectional mappings (M8).

takes a value of a property in source ontology as an input and returns a value of a property in the target ontology as an output. Below 4 types of suchlike mappings are described.

String Processing (M1). In the PM ontology an employee is represented by the attribute *hasParticipants* of the *Project* concept, which value is a list of strings in the “name/surname/social security number” format. In the CC ontology the concept *Person* represents employee. Thus, corresponding mapping defines that initial string must be split into 3 parts, which then will be used as values for the *hasFullName* and the *hasSocialSecurityNumber* attributes.

Data Type Transformation (M2). It can happen that in a proprietary data model the data type of a certain concept was not modeled in an optimal way, e.g. dates are represented as strings. In the example, the value of the *hasStartingDate* attribute in the PM ontology may be string in the “DD/MM/YYYY” format. If the data type of corresponding attribute in the CC ontology is *Date* then the data type transformation is needed.

Math Functions (M3). Here the relation between the entities represents some mathematical expression. For an instance, to obtain a value of the *hasDuration* attribute in the CC ontology a substraction of *hasStartingDate* from *hasEnding-Date* in PM ontology must be performed.

User-Defined Functions (M4). This category comprises functions that are not supported by the used technology, but must be additionally implemented. In the example, a value of the *hasAmortization* attribute of an mechatronic component will depend on its location and size and also on its installation date.

Granularity (M5). In the example, each employee in the PM ontology is represented by a string value of the hasParticipants attribute, while in the CC ontology the concept *Person* serves the same purpose.

Schematic Differences (M6). This type of mappings can arise when two ontologies on a similar domain were created separately and thus, the same semantics was modeled differently. In the example, a connection between physical devices in the ME ontology is represented by the *Connection* concept with the *sourceComponent* and *targetComponent* attributes, while in the CC ontology the same semantics is expressed with the *connectedWith* attribute of the *PhysicalComponent* concept.

Conditional Mappings (M7). One of the mapping aspects we wish to examine is to what extent technologies allow for the transformation of descriptions of one type of thing into descriptions of another type. In the example, the concept "signal" models a connection between physical devices and their controllers and corresponding PLC code. A value of the *hasFunctionalText* attribute of the *PLC* concept from the SE ontology is a string in "CPU number/input module/output module" format. Corresponding mapping states that for any pair of PLC and physical component, such that their values of *hasFunctionalText* and *hasInputModule* and *hasOutputModule* attributes have certain conformity an instance of *Signal* concept must be created in the CC ontology with appropriate values in its *hasInput*, *hasOutput* and *hasCPUNumber* attributes.

Bidirectional Mappings (M8). Usually mappings are specified in a way that knowledge/data flow can occur in one direction only. However, for some applications it is beneficial to define bidirectional mappings between the entities in order to reduce total amount of mappings and facilitate their maintenance [4].

4 Technology Evaluation for Mapping Categories

Below we describe the technologies that can be applied to create mappings between OWL ontologies before providing an overview on how well these technologies cover the examples in Fig. 1.

The Web Ontology Language (OWL) is an ontology language where one can declare relations between concepts such as equivalence, subsumption, etc. and allows one to infer additional information about instances by reasoning over the properties of classes and relations. OWL itself neither supports the representation of mappings between ontologies nor provides means for translating instances from one ontology to another. One way to represent mappings is to use the **SPARQL CONSTRUCT** query form, which returns an RDF graph based on template instantiated with he results of a query.

A second option is to use rules, which can be declared on top of OWL ontologies. Apache Jena[8] includes a rule-based inference engine called **Jena Rules** for reasoning with RDF and OWL data sources based on a Datalog implementation. Datalog is a declarative logic programming language that is becoming increasingly popular for data integration tasks [6]. **SPARQL Inference Notation** (SPIN) [7] is currently submitted to W3C and provides – amongst others – means to link class definitions with SPARQL queries (ASK and CONSTRUCT)

[8] Apache Jena: `http://jena.apache.org/`

Table 1. Evaluation of Technologies for Different Mapping Categories

Legend: Y = yes, P = partial, N = no * = depends on provider	**String**	**Data Type**	**Math**	**User-defined**	**Granularity**	**Schematic**	**Conditional**	**Bi-directional**	
	M1	**M2**	**M3**	**M4**	**M5**	**M6**	**M7**	**M8**	**Standard**
Jena-Rules	P	P	P	*	Y	Y	Y	P	-
SPIN	Y	Y	Y	Y*	Y	Y	Y	P	W3C Sub.
SWRL	Y	Y	Y	*	Y	Y	Y	P	W3C Rec.
SPARQL CONSTRUCT	P	Y	P	*	Y	Y	Y	N	W3C Rec.

to infer triples. The **Semantic Web Rule Language**, (SWRL) is a W3C recommendation for a Semantic Web rules language, combining OWL DL – a decidable fragment of OWL – with those of the Rule Markup Language. Rules are thus expressed in terms of OWL concepts. Note that SPARQL CONSTRUCT is *not* a rule language and "merely" allows one to make a transformation from one graph match to another graph; i.e., a one-step transformation.

Now we provide an overview on how well above mentioned technologies cover the examples in Fig. 1. This comparison is depicted in Table 1. Where necessary, we provide some extra information on certain caveats of the technologies.

String Processing (M1). The built-in string functions for Jena-Rules are limited; there exists for instance no function for obtaining substrings or string replacement. SPARQL supports some additional basic string functions. We therefore consider the string processing coverage of Jena-Rules and SPARQL partial. SWRL and SPIN offer additional string functions. Some frameworks – such as Jena – allows one to create custom functions (cfr. M4).

Data Type Transformation (M2). There are two aspects to be analyzed: xsd data type transformations and transforming values into other values. SPARQL 1.1 – and hence, also SPIN – allows one to cast the value of a variable to another xsd data type. The Jena-Rules and SWRL data type transformations are limited; e.g., Jena-Rules has a built-in primitive for creating a URI out of a set list of strings. As SWRL and Jena-Rules are used to infer additional triples to query, the SPARQL queries built on top of these rules can do the casting. The second type of transformation is covered by M1 and M4.

Math Functions (M3). All considered technologies support arithmetic, albeit Jena-Rules implementation has some limitations as mentioned in the previous Section. SWRL and SPIN even provide some additional functions (e.g., *sin*, *cos*, etc.), which are not defined in the SPARQL specification and their existence can depend on the provider. Again, when necessary, some function can be provided as a user-defined function (cfr. M4).

User-Defined Functions (M4). The implementation of user-defined functions is interesting to look at. Some simple functions can be built via rules and others might need to be implemented or already exist. Defining functions as part of the mapping can only be done in SPIN. For Jena Rules, SWRL and SPARQL, one can include custom functions if the software allows so. Jena, for instance, allows one to create Java classes representing functions that can be imported.

Jena thus covers the inclusion of custom functions, albeit via code. This is why we denoted this possibility in the table with an asterisk. SPIN "inherits" this aspect by depending on Jena.

Granularity (M5) and Schematic Differences (M6) tackled the problem of structural differences. All technologies provide means for transforming information into different representations. The limits actually depend on the specific requirements of a domain. In the presented use case scenario, one could use the social security number to create a URI for that person. Otherwise, blank nodes have to be introduced. The application domain of the knowledge bases could also have business rules that tell us how to construct URIs; which can be accomplished with aforementioned mappings.

Conditional Mappings (M7). Unlike OWL, which only allows relations between classes and properties via class- and property relations, both rules and SPARQL CONSTRUCT allow the transformation of descriptions of one type of concept into descriptions of other types of (related) concepts.

Bidirectional Mappings (M8). Ideally, mappings are bidirectional and maintained as such in one artifact. However, this can only be guaranteed if there is a bijective mapping between models of two ontologies. Take, for example, the mapping from one entity containing with a height and width as an attribute to another entity with a surface. Computing the surface by multiplying the height and width is easy, but the inverse yields more than one possibility. None of the studied technologies allow for bidirectional rules. However, we can examine the problem from a different angle and see to what extent we can "simulate" bidirectionality by storing both mappings in one artifact. When possible, bidirectional mappings can be declared with Jena-Rules, SWRL and SPIN. This is not possible with SPARQL CONSTRUCT queries, as a SPARQL CONSTRUCT queries allows one to populate a graph based on the result set of a query.

For our use case, SPIN and SPARQL CONSTRUCT are suitable candidates to represent mappings. SPARQL has the advantage of being W3C recommendation for quite a while and therefore different software solutions exist. Some of these solutions support the definition of custom functions. The downside is that mappings that depend on other mappings have to be executed in a certain order and are not a part of the knowledge base. Using SPIN, rules are part of the knowledge base. SPIN even has a notion of function that can be reused in different parts of the knowledge base. The downside is that SPIN is still in consideration by W3C. An implementation of SPIN is available from TopBraid[9] that depends on Jena. Given the fact that it is more desirable to have the mappings as part of the knowledge base, SPIN proves to be the more suitable candidate.

5 Conclusion and Future Work

In this paper we examined available technologies that support mapping representation between ontologies regarding the ability to represent complex mappings

[9] `http://www.topquadrant.com/`

that were identified based on requirements for the data integration projects of an industry partner. We evaluated existing technologies for mapping representation regarding the support provided for introduced mapping categories and discussed which technology would fit the best for the described use case scenario. The results show that in the introduced use case scenario the SPIN would be the optimal choice.

Possible directions for future work could include: a) Investigation on how different applications scenarios will influence the choice of optimal mapping representation technology; b) Following the work of Shvaiko et al. [12], focusing on the construction of mappings and considering it as a collaborative (and therefore social) process.

Acknowledgements. This work has been supported by the Christian Doppler Forschungsgesellschaft and by the Erasmus Mundus Action 2 Programme of the European Union.

References

1. Biffl, S., Moser, T., Winkler, D.: Risk assessment in multi-disciplinary (software+) engineering projects. International Journal of Software Engineering and Knowledge Engineering 21(02), 211–236 (2011)
2. Brockmans, S., Haase, P., Stuckenschmidt, H.: Formalism-independent specification of ontology mappings – A metamodeling approach. In: Meersman, R., Tari, Z. (eds.) OTM 2006. LNCS, vol. 4275, pp. 901–908. Springer, Heidelberg (2006)
3. Ehrig, M.: Ontology alignment: bridging the semantic gap, vol. 4. Springer Science+ Business Media (2007)
4. Ghidini, C., Serafini, L., Tessaris, S.: On relating heterogeneous elements from different ontologies. In: Kokinov, B., Richardson, D.C., Roth-Berghofer, T.R., Vieu, L. (eds.) CONTEXT 2007. LNCS (LNAI), vol. 4635, pp. 234–247. Springer, Heidelberg (2007)
5. Haase, P., Motik, B.: A mapping system for the integration of owl-dl ontologies. In: Proceedings of the First International Workshop on Interoperability of Heterogeneous Information Systems, pp. 9–16. ACM (2005)
6. Huang, S.S., Green, T.J., Loo, B.T.: Datalog and emerging applications: an interactive tutorial. In: Sellis, T.K., Miller, R.J., Kementsietsidis, A., Velegrakis, Y. (eds.) SIGMOD Conference, pp. 1213–1216. ACM (2011)
7. Knublauch, H., Handler, J.A., Idehen, K.: SPIN - Overview and Motivation. W3C Member Submission, W3C (February 2011), `http://www.w3.org/Submission/2011/SUBM-spin-overview-20110222/`
8. Maedche, A., Motik, B., Silva, N., Volz, R.: MAFRA – A MApping FRAmework for distributed ontologies. In: Gómez-Pérez, A., Benjamins, V.R. (eds.) EKAW 2002. LNCS (LNAI), vol. 2473, pp. 235–250. Springer, Heidelberg (2002)
9. Noy, N.F.: Semantic integration: a survey of ontology-based approaches. ACM Sigmod Record 33(4), 65–70 (2004)
10. Scharffe, F., de Bruijn, J.: A language to specify mappings between ontologies. In: Proc. of the Internet Based Systems IEEE Conference (SITIS 2005). Citeseer (2005)
11. Scharffe, F., Fensel, D.: Correspondence patterns for ontology alignment. In: Gangemi, A., Euzenat, J. (eds.) EKAW 2008. LNCS (LNAI), vol. 5268, pp. 83–92. Springer, Heidelberg (2008)
12. Shvaiko, P., Euzenat, J.: Ten challenges for ontology matching. In: Meersman, R., Tari, Z. (eds.) OTM 2008, Part II. LNCS, vol. 5332, pp. 1164–1182. Springer, Heidelberg (2008)

Incremental Maintenance of RDF Views of Relational Data

Vania Maria P. Vidal[1], Marco Antonio Casanova[2], and Diego Sá Cardoso[1]

[1] Federal University of Ceará, Fortaleza, CE, Brazil
{vvidal,dcardoso}@lia.ufc.br
[2] Department of Informatics – Pontifical Catholic University of Rio de Janeiro, RJ, Brazil
casanova@inf.puc-rio.br

Abstract. This paper proposes an incremental maintenance strategy, based on rules, for RDF views defined on top of relational data. The first step relies on the designer to specify a mapping between the relational schema and a target ontology and results in a specification of how to represent relational schema concepts in terms of RDF classes and properties of the designer's choice. Using the mappings obtained in the first step, the second step automatically generates the rules required for the incremental maintenance of the view.

Keywords: RDF View, Incremental View Maintenance, Correspondence Assertions.

1 Introduction

As RDF becomes the facto standard for publishing structured data over the Web and since most business data is currently stored in relational database systems, the problem of publishing relational data in RDF format has special significance. A general and flexible way to publish relational data in RDF format is to create RDF views of the underlying relational data. The contents of views can be materialized to improve query performance and data availability. However, to be useful, a materialized view must be continuously maintained to reflect dynamic source updates. Basically, there are two strategies for materialized view maintenance. *Re-materialization* re-computes view data at pre-established times, whereas *incremental maintenance* periodically modifies part of the view data to reflect updates to the database. It has been shown that incremental maintenance generally outperform full view re-computation [1,2,6,9, 13,17,18,20].

This paper proposes an incremental maintenance strategy, based on rules, for RDF views defined on top of relational data. The strategy has two major steps: The *mapping generation step* relies on the designer to specify a mapping between the relational schema and a target ontology and results in a specification of how to represent relational schema concepts in terms of RDF classes and properties of the designer's choice. The mapping induces a RDF view that is exported from the data source. The *view maintenance rules generation step* automatically generates the rules required for incremental maintenance of the RDF view from the mappings.

Our solution has the following major points. First, we propose correspondence assertions as a convenient way to specify customized mappings between target RDF vocabularies and base relational schemas. The benefits of using declarative formalisms for schema mappings are well known [3,12]. The concept of correspondence assertions was used in an earlier paper [20] to investigate XML views, but it proved to be much simpler to apply the concept to the context discussed in this paper. It is important to pointing out that the problem of generating schema mappings is outside the scope of this paper.

Second, the views that we address are focused on schema-directed RDF publishing. As such, the correspondence assertions induce schema mappings defined by the class of projection-selection-equijoin queries, which supports most types of data restructuring that are common in data publishing. We make a compromise in constraining the expressiveness of mappings to obtain an algorithm that is very efficient. Furthermore, the views are self-maintainable.

Third, our rules can be implemented using triggers. Hence, no middleware system is required, since triggers are responsible for directly propagating the changes to the materialized views.

Fourth, most of the work is done at view definition time. Based on the mappings, at view definition time, we are able to: (i) identify all source updates that are relevant to the view; and (ii) define the view maintenance statements required to maintain the view w.r.t a given relevant update. We emphasize that the view maintenance statements are defined based solely on the source update and current source state and, hence, no access to the materialized view is required. This is important when the view is maintained externally [18], because accessing a remote data source may be too slow.

Lastly, the view maintenance statements propagated by the rules do not require any additional queries over data source to maintain the view. Again, this becomes important when the view is maintained externally [18].

The use of rules is therefore a very effective solution for incremental maintenance of external views. However, creating rules that correctly maintain an RDF view can be a complex process, which calls for tools that automate the rule generation process. In this paper, we show that, based on a set of correspondence assertions [20], one can generate, automatic and efficiently, all the rules required to maintain a materialized view. Our formalism allows us to justify that the rules generated by our approach correctly maintain the view.

The remainder of this paper is organized as follows. Section 2 summarizes related work in the area of incremental view maintenance. Section 3 introduces the correspondence assertions. Section 4 presents the example used throughout the paper. Section 5 describes our approach for the automatic generation of incremental view maintenance rules, based on the correspondence assertions. Section 6 contains the conclusions.

2 Related Work

The problem of Incremental View Maintenance has been extensively studied for relational view [6, 11] as well as for object-oriented view [2, 13]. There have been also incremental maintenance algorithms for semi-structured views [1, 14, 21] and XML views [7, 9, 17, 20]. Different data models and view specification languages have been assumed by a number of researchers. The algorithms in [1, 14, 21] are developed for views defined with a query over graph structures. The views considered in [7, 9] are defined using an XML algebra over XML trees, and the views in [17] are defined using path expressions over XML documents.

The incremental algorithm in [4] maintains XML documents produced by an ATG, a formalism for mapping a relational schema to a predefined (possibly recursive) DTD. In their approach, a middleware system interacts with the underlying DBMS and maintains a hash index and a sub tree pool for the external XML view. The main problem with this approach, not to mention the high complexity of the algorithm, is that it requires several round-trips between the middleware and the DBMS. Therefore, the view is not self maintainable, which is a desirable feature for external views (view stored outside the DBMS). Other drawbacks are that the use of in-memory hash table limits the technique for large documents cached in a middleware, and it is not possible to detect irrelevant updates.

The algorithm in [20] incrementally maintains materialized XML views of relational data, in the context of the SQL/XML [8] standard. The algorithm has four major steps: first, it identifies the view paths that are relevant to a base update; second, it identifies all elements in a relevant path that are affected by the base update; third, it generates the list of updates required to maintain the affected elements; and, finally, it sends the list of updates to the view.

None of the above techniques can be directly applied to RDF views of relational data.

3 Correspondence Assertions

3.1 Basic Concepts and Notation

As usual, we denote a *relation scheme* as $R[A_1,....,A_n]$ and adopt *mandatory* (or *not null*) *attributes*, *keys*, *primary keys* and *foreign keys* as relational constraints. In particular, we use $F(R{:}L,S{:}K)$ to denote a foreign key, named F, where L and K are lists of attributes from R and S, respectively, with the same length. We also say that F *relates R and S*.

A *relational schema* is a pair $\boldsymbol{S}=(\boldsymbol{R},\boldsymbol{\Omega})$, where $\boldsymbol{R}$ is a set of relation schemes and $\boldsymbol{\Omega}$ is a set of relational constraints such that: (i) $\boldsymbol{\Omega}$ has a unique primary key for each relation scheme in $\boldsymbol{R}$; (ii) $\boldsymbol{\Omega}$ has a mandatory attribute constraint for each attribute which is part of a key or primary key; (iii) if $\boldsymbol{\Omega}$ has a foreign key of the form $F(R{:}L,S{:}K)$, then $\boldsymbol{\Omega}$ also has a constraint indicating that K is the primary key of S. The *vocabulary* of $\boldsymbol{S}$ is the set of relation names, attribute names, etc. used in $\boldsymbol{S}$. Given a relation scheme $R[A_1,....,A_n]$ and a *tuple variable t* over R, we use $t.A_k$ to denote the *projection* of t over A_k. We use *selections* over relation schemes, defined as usual.

Let $\boldsymbol{S}=(\boldsymbol{R},\boldsymbol{\Omega})$ be a relational schema and R and T be relation schemes of $\boldsymbol{S}$. A list $\varphi=[F_1,...,F_{n-1}]$ of foreign key names of $\boldsymbol{S}$ is a *path from R to T* iff there is a list $R_1,...,R_n$ of relation schemes of $\boldsymbol{S}$ such that $R_1=R$, $R_n=T$ and F_i *relates* R_i *and* R_{i+1}. We say that tuples of *R reference tuples of T through* φ. A path φ is an *association path* iff $\varphi=[F_1,F_2]$, where the foreign keys are of the forms $F_1(R_2{:}L_2,R_1{:}K_1)$ and $F_2(R_2{:}M_2,R_3{:}K_3)$. For example, consider the relational schema *ISWC_REL* in Figure 1. The list of foreign keys names *[Fk_Publications,Fk_Authors]* is an association path from *Papers* to *Persons*, but *[Fk_Publications,Fk_Persons]* is not even a path.

We also recall a minimum set of concepts related to ontologies. A *vocabulary* $\boldsymbol{V}$ is a set of *classes*, *object properties* and *datatype properties*. An *ontology* is a pair $\boldsymbol{O}=(\boldsymbol{V},\Sigma)$ such that $\boldsymbol{V}$ is a vocabulary and Σ is a finite set of formulae in $\boldsymbol{V}$, the *constraints* of $\boldsymbol{O}$. Among the constraints, we consider those that define the *domain* and *range* of a property, as well as *cardinality constraints*, defined in the usual way.

3.2 Definition of the Correspondence Assertions

This section introduces the notion of correspondence assertion, leaving examples to Section 3. Let $\boldsymbol{S}=(\boldsymbol{R},\boldsymbol{\Omega})$ be a relational schema and $\boldsymbol{O}=(\boldsymbol{V},\Sigma)$ be an ontology and assume that Σ has constraints defining the domain and range of each property.

Definition 3.1: A *class correspondence assertion (CCA)* is an expression of one of following forms:

(i) $\Psi\colon C \equiv R[A_1,...,A_n]$

(ii) $\Psi\colon C \equiv R[A_1,...,A_n]\sigma$

where Ψ is the *name* of the CCA, C is a class of $\boldsymbol{V}$, R is a relation name of $\boldsymbol{S}$, $A_1,...,A_n$ are the attributes of the primary key of R, and σ is a selection over R. We also say that Ψ *matches* C with R.

Definition 3.2: An *object property correspondence assertion (OCA)* is an expression of one of following forms:

(i) $\Psi\colon P \equiv R$

(ii) $\Psi\colon P \equiv R\,/\,\varphi$

where Ψ is the *name* of the OCA, P is an object property of $\boldsymbol{V}$ and φ is a path from R.

Definition 3.3: A *datatype property correspondence assertion (DCA)* is an expression of one of following forms:

(i) $\Psi\colon P \equiv R\,/\,A$

(ii) $\Psi\colon P \equiv R\,/\,\{A_1,...,A_n\}$

(iii) $\Psi\colon P \equiv R\,/\,\varphi\,/\,B$

(iv) $\Psi\colon P \equiv R\,/\,\varphi\,/\,\{B_1,...,B_n\}$

where Ψ is the name of the DCA, P is a datatype property of $\boldsymbol{V}$, R is a relation name of $\boldsymbol{S}$, A is an attribute of R, $A_1,...,A_n$ are attributes of R, φ is a path from R to T, B is an attribute of T, and $B_1,...,B_n$ are attributes of T.

Definition 3.4: A correspondence assertion *is relevant to a relation R* iff it is of one of the following forms:

(i) $\Psi\colon C \equiv R[A_1,...,A_n]$

(ii) $\Psi\colon C \equiv R[A_1,...,A_n]\sigma$

(iii) $\Psi\colon P \equiv R$

(iv) $\Psi: P \equiv R / \varphi$
(v) $\Psi: P \equiv R / A$
(vi) $\Psi: P \equiv R / \{A_1,...,A_n\}$
(vii) $\Psi: P \equiv R / \varphi / B$
(viii) $\Psi: P \equiv R' / \varphi$ where φ has a foreign key of R.
(ix) $\Psi: P \equiv R' / \varphi / \{B_1,...,B_n\}$ φ has a foreign key of R.
(x) $\Psi: P \equiv R' / \varphi / B$, φ has a foreign key of R.

Definition 3.5: A *mapping* between ***V*** and ***S*** is a set ***A*** of correspondence assertions such that:

(i) If ***A*** has an OCA of the form $P \equiv R$, then ***A*** must have a CCA that matches the domain of *P* with *R*, and a CCA that matches the range of *P* also with *R*.

(ii) If ***A*** has an OCA of the form $P \equiv R/\varphi$, where φ is a path from *R* to *T*, then ***A*** must have a CCA that matches the domain of *P* with *R* and a CCA that matches the range of *P* with *T*.

(iii) If ***A*** has a DCA that matches a datatype property *P* in ***V*** with a relation name *R* of ***S***, then ***A*** must have a CCA that matches the domain of *P* with *R*.

3.3 Transformation Rules Generated by Correspondence Assertions

In this section, we introduce the notion of transformation rule and show how to interpret correspondence assertions as transformation rules. Let $\boldsymbol{O}=(\boldsymbol{V},\boldsymbol{\Sigma})$ be an ontology and $\boldsymbol{S}=(\boldsymbol{R},\boldsymbol{\Omega})$ be a relational schema, with vocabulary ***U***. Let ***X*** be a set of *scalar variables* and ***T*** be a set of tuple variables, disjoint from each-other and from ***V*** and ***U***.

A *literal* is a *range expression* of the form *R(t)*, where *R* is a relation name in ***U*** and *t* is a tuple variable in ***T***, or a *built-in predicate* of one of the forms shown in Table 1. A *rule body B* is a list of literals. When necessary, we use *"*$B[x_1,...,x_k]$*"* to indicate that the tuple or scalar variables $x_1,...,x_k$ occur in *B*. We also use *"*$R(t), B[t,x_1,...,x_k]$*"* to indicate that the rule body has a literal of the form *R(t)*.

A *transformation rule*, or simply a *rule*, is an expression of one of the forms:

- $C(x) \leftarrow B[x]$, where *C* is a class in ***V*** and *B[x]* is a rule body
- $P(x,y) \leftarrow B[x,y]$, where *P* is a property and *B[x,y]* is a rule body

Let ***A*** be a set of correspondence assertions that defines a mapping between ***V*** and ***S***, that is, ***A*** satisfies the conditions stated in Definition 2.5. Assume that each class *C* in ***V*** is associated with a namespace prefix.

Table 2 shows the transformation rules *induced* by the correspondence assertions in ***A***. For example, the rule on the right-hand side of Line 5 indicates that, for each tuple *t* of *R* such that *t.A* is not null, one should:

- Compute the URI *s* of the instance of domain *D* of *P* that *t* represents, using the class correspondence assertion Ψ_D: $D(s) \leftarrow R(t), B_D[t,s]$, where $B_D[t,s]$ stands for *"*$HasURI[\Psi](t,s)$*"*, if the CCA for *D* follows Line 1 of Table 2, or $B_D[t,s]$ stands for *"*$HasURI[\Psi](t,s), \sigma(t)$*"*, if the CCA for *D* follows Line 2;
- Translate the value of *A* in tuple *t*, generating the literal *v*;
- Associate *v* as the value for property *P* of *s*.

Table 1. Built-in predicates

Built-in predicate	**Intuitive definition**
nonNull(v)	*nonNull(v)* holds iff value *v* is not null
RDFLiteral(u, A, R, v)	Given a value *u*, an attribute *A* of *R*, a relation name *R*, and a literal *v*, *RDFLiteral(u, A, R, v)* holds iff *v* is the literal representation of *u*, given the type of *A* in *R*
HasReferencedTuples[φ](t,u) where *φ* is a path from *R* to *T*	Given a tuple *t* of *R* and tuple *u* of *T*, *HasReferencedTuples[φ](t,u)* holds iff *u* is referenced by *t* through path *φ*
HasURI[Ψ](t,s) where *Ψ* is a CCA for a class *C* of ***V***, using attributes $A_1,...,A_n$ of *R*	Given a tuple *t* of *R*, *HasURI[Ψ](t,s)* holds iff *s* is the URI obtained by concatenating the namespace prefix for *C* and the values of $t.A_1,...,t.A_n$
$concat([v_1,.., v_n],v)$	Given a list $[v_1,.., v_n]$ of string values, $concat([v_1,.., v_n],v)$ holds iff *v* is the string obtained by concatenating $v_1,..,v_n$

Table 2. Transformation Rules

	Correspondence Assertion	**Transformation Rule**
T1	$\Psi: C \equiv R[A_1,...,A_n]$	$C(s) \leftarrow R(t), HasURI[\Psi](t,s)$
T2	$\Psi: C \equiv R[A_1,...,A_n]\sigma$	$C(s) \leftarrow R(t), HasURI[\Psi](t,s), \sigma(t)$
T3	$\Psi: P \equiv R$ where: - ***A*** has a CCA Ψ_D that matches the domain *D* of *P* with *R* and Ψ_D has mapping rule $D(s)\leftarrow R(t),B_D[t,s]$ - ***A*** has a CCA Ψ_N that matches the range *N* of *P* with *R* and Ψ_N has mapping rule $N(o)\leftarrow R(t),B_N[t,o]$	$P(s,o) \leftarrow R(t), B_D[t,s], B_N[t,o]$
T4	$\Psi: P \equiv R / \varphi$ where: - *φ* is a path of *R* to *T* - ***A*** has a CCA Ψ_D that matches the domain *D* of *P* with *R* and Ψ_D has mapping rule $D(s)\leftarrow R(t),B_D[t,s]$ - ***A*** has a CCA Ψ_N that matches the range *N* of *P* with *T* and Ψ_N has mapping rule $N(o)\leftarrow T(u),B_N[u,o]$	$P(s,o) \leftarrow R(t), B_D[t,s],$ $HasReferencedTuples[\varphi](t,u),$ $T(u), B_N[u,o]$
T5	$\Psi: P \equiv R / A$ where: - ***A*** has a CCA Ψ_D that matches the domain *D* of *P* with *R* and Ψ_D has mapping rule $D(s)\leftarrow R(t),B_D[t,s]$ - *A* is an attribute of *R*	$P(s,v) \leftarrow R(t), B_D[t,s],$ $nonNull(t.A)$ $RDFLiteral(t.A, “A”, “R”,v)$
T6	$\Psi: P \equiv R / \varphi / A$ where: - *φ* is a path of *R* to *T* - ***A*** has a CCA Ψ_D that matches the domain *D* of *P* with *R* and Ψ_D has mapping rule $D(s)\leftarrow R(t),B_D[t,s]$ - *A* is an attribute of *T*	$P(s,v) \leftarrow R(t), B_D[t,s],$ $HasReferencedTuples[\varphi](t,u),$ $nonNull(u.A),$ $RDFLiteral(u.A, “A”, “T”,v)$

Table 2. (*Continued*)

T7	Ψ: $P \equiv R / \{A_1,...,A_m\}$ where: - ***A*** has a CCA Ψ_D that matches the domain D of P with R and Ψ_D has mapping rule $D(s) \leftarrow R(t), B_D[t,s]$ - $A_1,...,A_m$ are attributes of R	$P(s,v) \leftarrow R(t), B_D[t,s],$ $nonNull(t.A_1),...,nonNull(t.A_m),$ $RDFLiteral(t.A_1, \text{“}A_1\text{”}, \text{“}R\text{”}, v_1),$ $...,$ $RDFLiteral(t.A_m, \text{“}A_m\text{”}, \text{“}R\text{”}, v_m),$ $concat([v_1,..,v_m],v)$
T8	Ψ: $P \equiv R / \varphi / \{A_1,...,A_m\}$ where: - φ is a path from R to T - ***A*** has a CCA Ψ_D that matches the domain D of P with R and Ψ_D has mapping rule $D(s) \leftarrow R(t), B_D[t,s]$ - $A_1,...,A_m$ are attributes of T	$P(s,v) \leftarrow D(t), B_D[t,s],$ $HasReferencedTuples[\varphi](t,u),$ $nonNull(u.A_1),...,nonNull(u.A_m)$ $RDFLiteral(u.A_1, \text{“}A_1\text{”}, \text{“}T\text{”}, v_1),$ $...,$ $RDFLiteral(u.A_m, \text{“}A_m\text{”}, \text{“}T\text{”}, v_m),$ $concat([v_1,..,v_m],v)$

4 Running Example

In this section, we present the relational database schema *ISWC_REL* and the ontology *CONF_OWL*, which are used as a case study throughout the paper. We also present a set of correspondence assertions, which specifies the mapping between *CONF_OWL* and *ISWC_REL.*

Figure 1 depicts the relational schema *ISWC_REL*. Each table has a distinct primary key, whose name ends with 'ID'. *Persons* and *Papers* represent the main concepts. The attribute *conference* of *Papers* is a foreign key to *Conferences*. *Rel_Person_Paper* represents an N:M relationship between *Persons* and *Papers*. The labels of the arcs, such as *Fk_Publications*, are the names of the foreign keys.

Figure 2 depicts the ontology *CONF_OWL*, which reuses terms from four well-known vocabularies: *FOAF* (Friend of a Friend), *SKOS* (Knowledge Organization System), *VCARD* and *DC* (Dublin Core). We use the prefix "*conf*" for the new terms defined in the *CONF_OWL* ontology.

Table 3 shows a set of correspondence assertions that specifies a mapping between *CONF_OWL* and *ISWC_REL*, obtained with the help of the tool described in [19]. For example, the transformation rules induced by *CCA1*, *DCA1* and *OCA2* are (the translations from attribute values to RDF literals are omitted for simplicity):

- *CCA1* specifies that each tuple *t* in *Persons* produces one RDF triple:
  <*http://example.com/person/t.perID*> *rdf:type* *foaf:Person.*
- *DCA1* specifies that each tuple *t* in *Persons* produces one RDF triple:
  <*http://example.com/person/t.perID*> *foaf:name*
  concat(t.firstname, t.lastname).

- *OCA2* specifies that, for each tuple *t* in *Person*, for each tuple *t'* in *Topics* such that *t'* is referenced by *t* through path *[Fk_Authors, Fk_Publications, Fk_Papers, Fk_Topics]*, one triple of the form is generated:

  <http://example.com/person/t.perID> conf:ResearchInterests <http://example.com/org/t'.topicID>.

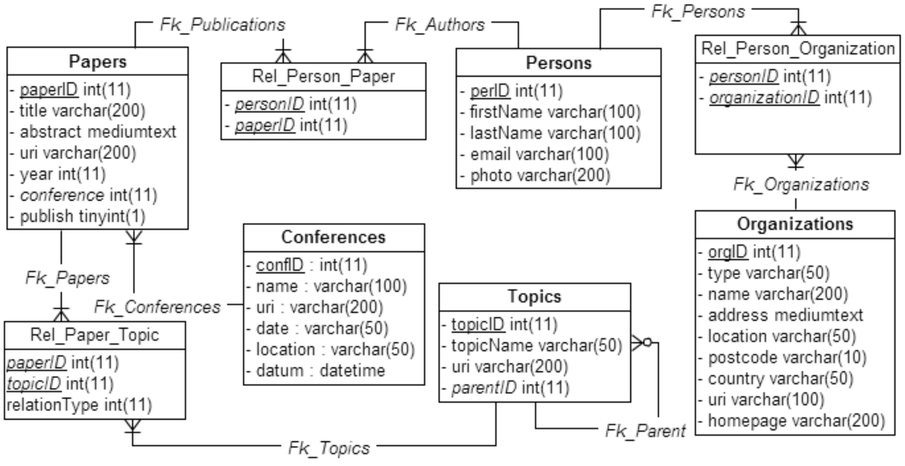


Fig. 1. ISWC_REL Database Schema

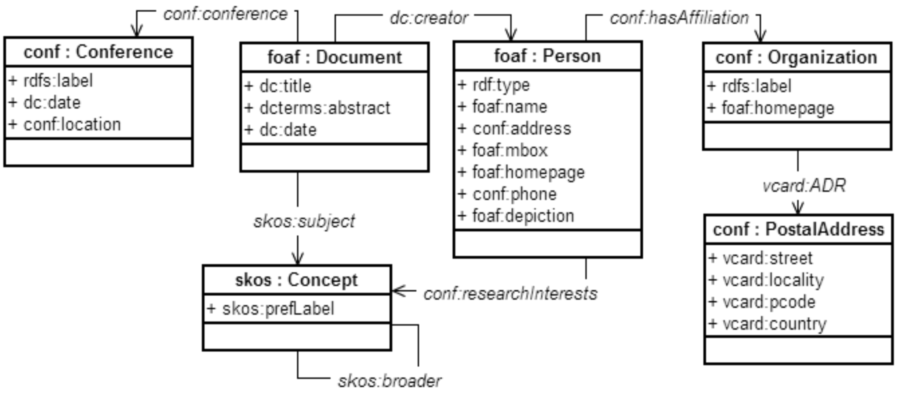


Fig. 2. CONF_OWL Target Ontology

Table 3. Correspondence Assertions

CCA1	*foaf:Person ≡ Persons[perID]*
CCA2	*foaf:Document ≡ Papers[paperID]*
CCA3	*conf:PostalAddress ≡ Organizations[orgID]*
CCA4	*conf:Organization ≡ Organizations[orgID]*
CCA5	*conf:Conference ≡ Conferences[confID]*
CCA6	*skos:Concept ≡ Topics[topicID]*
OCA1	*conf:hasAffiliation ≡ Persons / [Fk_Persons, Fk_Organizations]*
OCA2	*conf:researchInterests ≡ Persons / [Fk_Authors, Fk_Publications, Fk_Papers, Fk_Topics]*
OCA3	*vcard:ADR ≡ Organizations*
OCA4	*skos:subject ≡ Papers / [Fk_Papers, Fk_Topics]*
OCA5	*conf:conference ≡ Papers / Fk_Conferences*
OCA6	*skos:broader ≡ Topics / Fk_Parent*
DCA1	*foaf:name ≡ Persons / { firstName, lastName }*
DCA2	*foaf:mbox ≡ Persons / email*
DCA3	*rdfs:label ≡ Organization / name*
DCA4	*foaf:homepage ≡ Organization / homepage*
DCA5	*vcard:Street ≡ Organizations / address*
DCA6	*vcard:locality ≡ Organizations / location*
DCA7	*vcard:Pcode ≡ Organizations / postcode*
DCA8	*vcard:country ≡ Organizations /country*
DCA9	*dc:title ≡ Papers / title*
DCA10	*dcterms:abstract ≡ Papers / abstract*
DCA11	*dc:date ≡ Papers / year*
DCA12	*skos:prefLabel ≡ Topics / topicName*
DCA13	*rdfs:label ≡ Conferences / name*

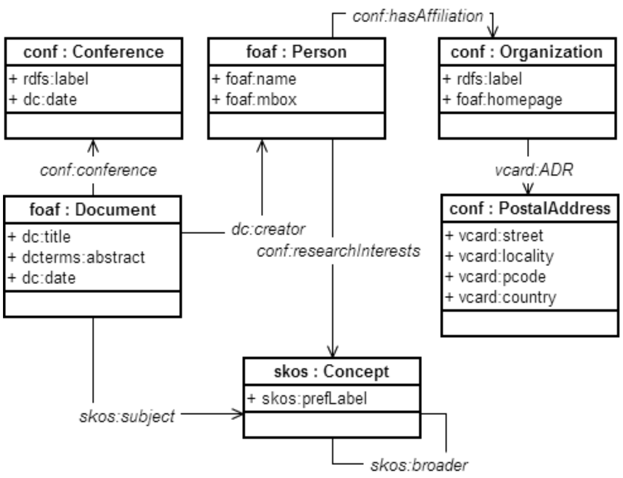


Fig. 3. ISWC_RDF Exported View Schema

5 Automatic Generation of Maintenance Rules

In this section, we present our process for generating a set of rules required for the incremental maintenance of materialized view data. The process inputs are:

- ***D** = (V_D,C_D)* is the target ontology, where V_D is the vocabulary of *D* and C_D is the set of constraints of ***D***
- ***S*** is the relational schema that is mapped to ***D***
- ***A*** is a mapping, that is, a set of correspondence assertions between V_D and ***S***

The process consists of two main steps. The first step involves the generation of the *exported RDF view schema V*, which is induced by the mapping ***A***, as described in [5]. Figure 3 shows the exported RDF view schema *ISWC_RDF* induced by the correspondence assertions in Table 3. The vocabulary of *ISWC_RDF* contains all the elements of the *CONF_OWL* ontology that match an element of *ISWC_REL.*

The second step involves the generation of a set of rules required for the incremental maintenance of the materialized view data. In following, we present how to use the view correspondence assertions to generate the set of rules required for the incremental maintenance of *V*.

In our approach, the process of generating the maintenance rules for *V* consists of the following steps:

- Obtain the set of all relations in ***S*** that are relevant to *V*. A relation *R* is relevant to *V* iff any correspondence assertion of *V* is relevant to *R* (see Def. 3.4).
- For each relation *R* that is relevant to *V*, three rules are generated (see Fig. 4). Rule (a) is triggered by deletions on *R*, Rule (b) by insertions on *R*, and Rule (c) by updates on *R*. An update is a deletion followed by an insertion.

When INSERT ON R **Then** U:= **GVU_INSERTonR (r**$_{new}$**)**; **ApplyUpdates(** V, U**)**; (a)	**When** DELETE ON R **Then** U:=**GVU_DELETEonR (r**$_{old}$**)**; **ApplyUpdates(** V, U**)**; (b)	**When** UPDATE ON R **Then** U:=**GVU_DELETEonR (r**$_{old}$**)**; **ApplyUpdates(** V, U**)**; U:= **GVU_INSERTonR (r**$_{new}$**)**; **ApplyUpdates(** V, U**)**; (c)

Fig. 4. Rules for maintenance of View *V* with respect to updates on relevant relation *R*

In Figure 4, the procedures *GVU_INSERT[R]* and *GVU_DELETE[R]* are automatically generated, at view definition time, based on the correspondence assertions of *V* that are relevant to *R*. The procedure *GVU_INSERT[R]* takes as input a tuple r_{new} inserted in *R* and returns the updates necessary to maintain the view *V*. The procedure *GVU_DELETE[R]* takes as input a tuple r_{old} deleted from *R*, and returns the updates necessary to maintain the view *V*. Appendix A shows the algorithm *Generate_ GVU_ INSERT[R]*, which takes as input the set of correspondence assertions of *V* that are relevant to *R*, and compiles the procedure *"GVU_ INSERT[R]"* with parameter r_{new}. The definitions of the built-in functions used in *Generate_GVU_ INSERT[R]* are defined in Table 4. The algorithm for compiling the procedure *"GVU_ DELETE[R]"* is very similar, and it is omitted here for brevity.

Table 5 shows the procedure *GVU_INSERT[Papers]*, which returns the updates necessary to maintain the view *ISWC_RDF* with respect to insertions on relation *Papers*. By Definition 4.5, we have that CCA2, DCA9 and OCA5 are relevant to relation *Papers*. The updates required for CCA2, DCA9 and OCA5 are defined according with transformation rules T1, T4 and T5 in Table2, respectively.

Table 6 shows the procedure *GVU_INSERT[Rel_Paper_Topic]*, which returns the updates necessary to maintain the view *ISWC_RDF* with respect to insertions on relation *Rel_Paper_Topic*. By Definition 4.5, we have that OCA2 and OCA4 are relevant to relation *Rel_Paper_Topic,* because the relation *Rel_Paper_Topic* is related by the foreign key *Fk_Topics* which occurs in the path of both OCA2 and OCA4. According to case 4 of the algorithm, the updates required by OCA2 for maintaining the view *ISWC_RDF* with respect to insertions on relation *Rel_Paper_Topic* reduce to: For each tuple t_1 in *Persons* that is referenced by r_{new} through path {*Fk_Papers, Fk_Publications*, *Fk_Author*}, recompute all *conf:research_interests* of t_1 according to transformation rule T5. Note that the property *conf:research_interests* of other tuples in *Papers* is not affected by insertions on relation *Rel_Paper_Topic.*

Table 4. Built-in Functions

Built-in function	Definition
GenerateRDFLiteral(u, A, R, v)	*v = GenerateRDFLiteral(u, A, R)* iff *RDFLiteral(u, A, R, v)*
ObtainsReferencedTuples[φ](t) where φ is a path from *R* to *T*	{ *u* / *HasReferencedTuples[φ](t,u)* }
GenerateURI[Ψ](t) where Ψ is a CCA for a class *C*.	*s = GenerateURI[Ψ](t)* iff *HasURI[Ψ](t, s)*
GenerateConcat([v_1*,.., *v_n*],v)*	*v = GenerateConcat(u, A, R)* iff *concat([*v_1*,.., *v_n*],v)*

Table 5. Procedure GVU_INSERT[Papers]

```
U := ∅;
/* Updates required by CCA2  (lines 2-4 of Generate_GVU_Insert[R] algorithm)
s := GenerateURI[CCA2]($r_{new}$);
U := U ∪ { "InsertTriple(s, "rdf:type", "foaf:Document");" };
 If  s≠ " "  then{
  /* Updates required by OCA5  (lines 11-16 of Generate_GVU_Insert[R] algorithm)
   Q = ObtainReferencedTuples[FK_Conferences]($r_{new}$);
   For each t in Q do {
      o := GenerateURI[CCA5](t);
      U := U ∪ { "If conf:conference(o) then InsertTriple(s, "conf:conference" , o);"};};
  /* Updates required by DCA9  (lines 21-24 of Generate_GVU_Insert[R] algorithm)
    If nonNull($r_{new}$.title ) then {
       v := GenerateRDFLiteral($r_{new}$.title, "title", "Papers");
       U := U ∪ {"InsertTriple(s, "dc:title", v);"};};
 }};
Return(U).
```

Table 6. Procedure GVU_Insert[Paper_Topic]_

```
U := ∅;
/* Updates required by OCA2  (lines 38-48 of Generate_GVU_Insert[R] algorithm)
Q1 = ObtainReferencedTuple[Fk_Papers, Fk_Publications,  Fk_Author]($r_{new}$);
For each $t_1$ in $Q_1$ do {
  s = GenerateURI[CCA1]($t_1$);
  U := U ∪ { "DeleteTriple(s, "conf:research_interests", ?o);"};
  Q2 = ObtainReferencedTuple[Fk_Authors, Fk_Publications, Fk_Papers, Fk_Topics ]($t_1$)};
  For each $t_2$ in $Q_2$ do {
      o = GenerateURI[CCA6]($t_2$);
      U := U ∪ { "If skos:concept(o) then InsertTriple(s, "conf:research_interests", o);"}};
/* Updates required by OCA4  (lines 38-48 of Generate_GVU_Insert[R] algorithm)
Q1 = ObtainReferencedTuple[FK_Papers]($r_{new}$);
For each $t_1$  in $Q_1$ do {
   s  = GenerateURI[CCA2]($t_1$);
   U := U ∪ { "DeleteTriple(s, "skos:Subject", ?o);"};
   Q2 = ObtainReferencedTuple[Fk_Papers, Fk_Topics]($t_1$);
   For each $t_2$ in $Q_2$ do {
          o = GenerateURI[CCA6]($t_2$);
          U := U ∪ { "If skos:concept(o) then InsertTriple(s, "skos:Subject", o);"}}}
Return(U).}
```

Table 7. Procedure GVU_Insert[Organizations]_

```
U := ∅;
/* Updates required by CCA3  (lines 2-4 of Generate_GVU_Insert[R] algorithm)
s := GenerateURI[CCA3]($r_{new}$);
U := U ∪ {"InsertTriple(s, "rdf:type",  "conf:PostalAddress");" };
If s≠ " "  then{
   /* Updates required by DCA5  (lines 17-20 of Generate_GVU_Insert[R] algorithm)
     If nonNull($r_{new}$.address) then {
        v := GenerateRDFLiteral($r_{new}$.address, "address", "Organizations");
        U := U ∪ {"InsertTriple(s, "vcard:Street", v);"};};
   /* Updates required by DCA6  (lines 21-24 of Generate_GVU_Insert[R] algorithm)
     If nonNull($r_{new}$.location) then {
        v := GenerateRDFLiteral($r_{new}$.location, "location", "Organizations");
        U := U ∪ {"InsertTriple(s, "vcard:locality", v);"};};
   /* Updates required by DCA7  (lines 21-24 of Generate_GVU_Insert[R] algorithm)
     If nonNull($r_{new}$.postcode) then {
        v := GenerateRDFLiteral($r_{new}$. postcode, " postcode", "Organizations");
        U := U ∪ {"InsertTriple(s, "vcard:Pcode", v);"};};
```

Table 7. (*Continued*)

```
/* Updates required by DCA8 (lines 21-24 of Generate_GVU_Insert[R] algorithm)
   If nonNull($r_{new}$.country) then {
      v := GenerateRDFLiteral($r_{new}$. country, "country", "Organizations");
      U := U ∪ {"InsertTriple(s, "vcard:country", v);"};};};
/* Updates required by CCA4 (lines 2-4 of Generate_GVU_Insert[R] algorithm)
s := GenerateURI[CCA4]($r_{new}$);
U := U ∪ { "InsertTriple(s, "rdf:type", "conf:Organization");" };
If $s \neq$ " " then{
  /* Updates required by OCA3 (lines 21-24 of Generate_GVU_Insert[R] algorithm)
     o := GenerateURI[CCA3]($r_{new}$);
     U := U ∪ {"If conf: PostalAddress(o) InsertTriple(s, "vcard:ADR", o);"};};
    /* Updates required by DCA3 (lines 21-24 of Generate_GVU_Insert[R] algorithm)
     If nonNull($r_{new}$.title ) then {
        v := GenerateRDFLiteral($r_{new}$.name, "name", "Organizations");
        U := U ∪ {"InsertTriple(s, "rdfs:label", v);"};};
  /* Updates required by DCA4 (lines 21-24 of Generate_GVU_Insert[R] algorithm)
     If nonNull($r_{new}$.homepage ) then {
        v := GenerateRDFLiteral($r_{new}$.homepage, "homepage", "Organizations");
        U := U ∪ {"InsertTriple(s, "foaf:homepage", v);"};};};};
Return(U).
```

6 Conclusions and Future Work

In this paper, we argued that, based on view correspondence assertions, we can automatic and efficiently identify all relations that are relevant to a view. Using view correspondence assertions, we can also define the rules required for maintaining a view with respect to updates on a relevant relation. Finally, we indicated how the rules can be automatically generated from the correspondence assertions.

We emphasize that, in our approach, the rules are responsible for directly propagating the changes to the materialized view. Our approach is effective for an externally maintained view because: the view maintenance rules are defined at view definition time; no access to the materialized view is required to compute the view maintenance statements propagated by the rules; and the propagated view maintenance statements do not require any additional queries over the data source to maintain the view.

We are currently working on the development of a tool to automate the generation of incremental view maintenance rules.

Acknowledgments. This work was partly supported by CNPq, under grants 301497/2006-0, 475717/2011-2 and 57128/2009-9, by FAPERJ, under grants E-26/170028/2008 and E-26/103.070/2011, and by CAPES, under grant PROCAD/NF 1128/2010.

References

1. Abiteboul, S., McHugh, J., Rys, M., Vassalos, V., Wiener, J.L.: Incremental Maintenance for Materialized Views over Semistructured Data. In: VLDB, pp. 38–49 (1998)
2. Ali, M.A., Fernandes, A.A., Paton, N.W.: Incremental Maintenance for Materialized OQL Views. In: DOLAP, pp. 41–48 (2000)
3. Bernstein, P.A., Melnik, S.: Model Management 2.0: Manipulating Richer Mappings. In: SIGMOD, pp. 1–12 (2007)
4. Bohannon, P., Choi, B., Fan, W.: Incremental evaluation of schema-directed XML publishing. In: SIGMOD, pp. 13–18 (2004)
5. Casanova, M.A., Beitman, K.K., Furtado, A.L., Vidal, V.M.P., Macedo, J.A.F., Gomes, R.V.A., Salas, P.E.R.: The Role of Constraints in Linked Data. In: Meersman, R., et al. (eds.) OTM 2011, Part II. LNCS, vol. 7045, pp. 781–799. Springer, Heidelberg (2011)
6. Ceri, S., Widom, J.: Deriving productions rules for incremental view maintenance. In: VLDB, pp. 577–589 (1991)
7. Dimitrova, K., El-Sayed, M., Rundensteiner, E.A.: Order-sensitive View Maintenance of Materialized XQuery Views. In: Song, I.-Y., Liddle, S.W., Ling, T.-W., Scheuermann, P. (eds.) ER 2003. LNCS, vol. 2813, pp. 144–157. Springer, Heidelberg (2003)
8. Eisenberg, A., Melton, J., Kulkarni, K., Michels, J.E., Zemke, F.: SQL:2003 has been published. In: SIGMOD, vol. 33(1), pp. 119–126 (2004)
9. El-Sayed, M., Wang, L., Ding, L., Rudensteiner, E.: An algebraic approach for Incremental Maintenance of Materialized Xquery Views. In: WIDM, pp. 88–91 (2002)
10. Fuxman, A., Hernandez, M.A., Ho, H., Miller, R.J., Papotti, P., Popa, L.: Nested mappings: schema mapping reloaded. In: VLDB, pp. 67–78 (2006)
11. Gupta, A., Mumick, I.S.: Materialized Views. MIT Press (2000)
12. Jiang, H., Ho, H., Popa, L., Han, W.: Mapping-Driven XML Transformation. In: WWW, pp. 1063–1072 (2007)
13. Kuno, H.A., Rundensteiner, E.A.: Incremental Maintenance of Materialized Object-Oriented Views in MultiView: Strategies and Performance Evaluation. IEEE Transaction on Data and Knowledge Engineering 10(5), 768–792 (1998)
14. Liefke, H., Davidson, S.B.: View Maintenance for Hierarchical Semi-structured Data. In: Kambayashi, Y., Mohania, M., Tjoa, A.M. (eds.) DaWaK 2000. LNCS, vol. 1874, pp. 114–125. Springer, Heidelberg (2000)
15. Miller, R.J.: Retrospective on Clio: Schema Mapping and Data Exchange in Practice. In: International Workshop on Description Logics (2007)
16. Popa, L., Velegrakis, Y., Miller, R.J., Hernandez, M.A., Fagin, R.: Translating Web Data. In: VLDB, pp. 598–609 (2002)
17. Sawires, A., Tatemura, J., Po, O., Agrawal, D., Candan, K.: Incremental Maintenance of Path-expression Views. In: SIGMOD, pp. 443–454 (2005)
18. Staudt, M., Jarke, M.: Incremental maintenance of externally materialized views. In: VLDB, pp. 75–86 (1996)
19. Vidal, V.M.P., Casanova, M.A., Neto, L., Monteiro, J.M.: 2013. R2RML by Assertion: A Semi-Automatic Tool for Generating Customized R2RML Mappings. In: ESWC Demos (2013)
20. Vidal, V.M.P., Lemos, F.C.L., Araújo, V., Casanova, M.A.: A Mapping-Driven Approach for SQL/XML View Maintenance. In: ICEIS, pp. 65–73 (2008)

Appendix

Algorithm Generate "GVU_Insert[R]"

Input: The set of correspondence assertions relevant to relation *R*.
Output: Procedure "*GVU_INSERT[R]*" with parameter r_{new}.

```
τ := {"U := ∅;"};
For each CCA $\Psi_C$ that matches a class C with R do {
  If $\Psi_C$: $C \equiv R[A_1,\dots, A_n]$ then
   τ := τ + "s := GenerateURI[$\Psi_C$]($r_{new}$); U := U ∪ {"InsertTriple(s, "rdf:type", "C");"};"
   else if $\Psi_C$ : $C \equiv R[A_1,\dots,A_n]\sigma$ then
      τ := τ + "if $\sigma(r_{new})$= true then { s := GenerateURI [$\Psi_C$] ($r_{new}$);
                                    U := U ∪ {"InsertTriple(s, "rdf:type", "C");"}
                              else s:= " ";";
  τ := τ + "If s≠ " " then {";
    For each OCA $\Psi_P$ : $P \equiv R/\varphi$ where C is the domain of P, $\varphi$ is a path from R to T,
     and $\Psi_N$ is the CCA that matches the range N of P with T do {
      τ := τ + "Q := ObtainReferencedTuples[$\varphi$]($r_{new}$);"
             + "For each t in Q do {
                 o := GenerateURI[$\Psi_N$](t);
                 U := U ∪ {"If N(o) then "InsertTriple(s, "P", o);"};};};
    For each OCA $\Psi_P$ : $P \equiv R$ where C is the domain of P, $\Psi_N$ is the CCA that
      matches the range N of P with R do {
       τ := τ + "o := GenerateURI[$\Psi_N$]($r_{new}$);"
              + "U := U ∪ {"If N(o) then "InsertTriple(s, "P", o);"};"};
For each DCA $\Psi_P$ : $P \equiv R/A$ where C is the domain of P do {
       τ := τ + "if nonNull($r_{new}$.A) then {
                  v = GenerateRDFLiteral($r_{new}$.A, "A", "R");
                  U := U ∪ {"InsertTriple(s, "P", v);"};";
    For each DCA $\Psi_P$:$P \equiv R/\varphi/A$ where C is the domain of P, $\varphi$ is a path from R to T,
      and $\Psi_N$ is the CCA that matches the range N of P with T do
       τ := τ + "Q := ObtainReferencedTuples[$\varphi$]($r_{new}$);";
              + "For each t in Q do {
                 "if nonNull(t.A) then {
                  v = GenerateRDFLiteral(t.A, "A", "T");
                  U := U ∪ {"InsertTriple(s, "P", v);"}};";

  };
};
```

```
**For each** OCA $\Psi_P$ : $P \equiv R_1/\varphi$ where $\varphi = [FK_1, ... ,FK_{n-1}]$ is a path from $R_1$ to $R_n$, and $R = R_j$, $2 \le j \le n$, **do {**
   Let $\varphi_1 = [FK_j, ... ,FK_1]$;
   Let $\Psi_D$ be the CCA that matches the domain D of P with $R_1$, and $\Psi_N$ be
   the CCA that matches the range $N$ of $P$ with $R_n$.
   $\tau := \tau$ + "*Q1 := ObtainReferencedTuples*[$\varphi_1$]($\mathbf{r}_{new}$);";
         + "For each $t_1$ in *Q1* do {
               *s := GenerateURI*[$\Psi_D$]($t_1$);
               U := U $\cup$ {"*DeleteTriple*(*s*, "*P*", *?o*);"};
              *Q2* := *ObtainReferencedTuples*[$\varphi$]($t_1$);
               For each $t_2$ in *Q2* do {
                *o* := *GenerateURI*[$\Psi_N$]($t_2$);
                U := U $\cup$ {"If N(o) then "*InsertTriple*(*s*, "*P*", *o*);"};};}";
**};**
**For each** DCA $\Psi_P$ : $P \equiv R_1/\varphi/A$ where $\varphi = [FK_1, ... ,FK_{n-1}]$ is a path from $R_1$ to $R_n$, and $R = R_j$, $2 \le j \le n$, **do**
   Let $\varphi_1 = [FK_j, ... ,FK_1]$;
   Let $\Psi_D$ be the CCA that matches the domain D of P with $\mathbf{R}_1$, and $\Psi_N$ be
    the CCA that matches the range $N$ of $P$ with $R_n$.
   $\tau := \tau$ + "*Q1 := ObtainReferencedTuples*[$\varphi_1$]($\mathbf{r}_{new}$);";
         + "For each $t_1$ in *Q1* do {
               *s := GenerateURI*[$\Psi_D$] ($t_1$);
               U := U $\cup$ {"*DeleteTriple*(*s*, "*P*", *?o*)"};
              *Q2* := *ObtainReferencedTuples*[$\varphi$]($t_1$);
               For each $t_2$ in *Q2* do {
                 "if *nonNull*($t_2$.A) then {
                  *v* = *GenerateRDFLiteral*($t_2$.*A*, "*A*", "$R_n$");
                  U := U $\cup$ {"*InsertTriple*(*s*, "*P*", *v*); "};};}";
};
**Return($\tau$).**
```

Semantically Interlinked Notification System for Ubiquitous Presence Management*

Qaiser Mehmood[1], Muhammad Intizar Ali[1], Ollie Fagan[2], Owen Friel[2], and Alessandra Mileo[1]

[1] DERI, National University of Ireland, Galway, Ireland
{qaiser.mehmood,ali.intizar,alessandra.mileo}@deri.org
[2] Cisco Systems, Galway, Ireland
{ofagan,ofriel}@cisco.com

Abstract. Presence based notification systems play a pivotal role in any collaborative working environment by providing near real time information about the status, locality and presence of the collaborators. Instant Messaging (IM) tools provide a simple low cost solution to support communication and collaboration in the working environment. With the wide adoption of smart phones, there is an increasing trend of using IM clients on mobile devices for exchanging and managing presence information. However, in modern organisations the use of collaborative tools is not limited only to IMs, but various other tools are also used, such as project management tools, meeting agenda, calendar and alike. In this paper we design, develop and implement an agile platform for ubiquitous presence management that relies on semantically interlinked vocabularies, existing information extraction tools for semantic enhancement of heterogeneous enterprise collaborative tools, continuous query processing for semantic integration and event detection in dynamic environment, and semantic transformation via query language for event-triggered notification actions based on the users context. We showcase our proposed solution by demonstrating a real world scenario for Cisco's Unified Presence System and we evaluate our solution in terms of requirements for Ubiquitous Presence Management System (UPMS) compared to related solutions, and overhead of each additional processing step, namely semantic transformation, continuous query processing and notification.

Keywords: Semantic Presence Management, Notification Systems, NFC, Instant Messaging, XMPP, XSPARQL, Stream Query Processing.

1 Introduction

Instant Messaging (IM) has been an effective way of communicating and collaborating in the modern enterprises [17]. Flexible and interoperable solutions for online IM-based communication systems rely on the standard XML-based

* This work is jointly funded by Science Foundation Ireland, Grant No. SFI/08/CE/l1380 (Lion2) and Cisco Systems, Galway, Ireland.

Extensible Messaging and Presence Protocol (XMPP) protocol for message exchange [19].

Advent of the modern communication technologies and smart phones have facilitated IM users by providing communication facilities on-the-go, increasing the number of mobile IM users. According to the forecast from Juniper Research, mobile IM users will exceed 1.3 billion by 2016[1]. This phenomenon that sees IM at the centre of the communication infrastructure is also observed in enterprise solutions for unified communications, where IM represents a flexible and easy-to-use tool for communication, presence management and collaboration.

The more IM becomes popular as a collaborative tool, the higher the need for solutions that can enhance collaboration by seamlessly integrating and correlating IMs messaging and semantic presence with information from other collaborative tools (including calendars, emails and project management tools) with sensors for physically contextualised presence [3] (including sensors embedded in mobile devices and other localised sensors).

In previous works, we focused on the use of semantics to enhance presence management and facilitate integration and exchange of semantic presence information [2,8]. Semantic enrichment in IM communication paves the way for ubiquitous presence management, because it provides a standard and interoperable way of interlinking and sharing heterogeneous presence knowledge.

However, in order to materialise the idea of Ubiquitous Presence Management System (UPMS), we need to design and implement an agile solution that can cater for the richness of physical contexts (location, ongoing meetings, available resources) and integrate these physical contexts with other potential sources of contextualised presence such as calendars and other collaborative tools [9]. Physical contexts are not only rich in terms of variety of the information produced, but they also have an intrinsic dynamic nature we need to cater for. While variety is addressed by using semantic technologies for data integration and linking, we tackle the contextual dynamicity by resorting to the paradigm of event-based notification systems: dynamic knowledge from physical and collaborative contexts is interlinked and captured as events, which trigger appropriate actions or notifications.

To facilitate a solution with high interoperability and flexibility, we need to rely on existing standards for IM communication and semantic enrichment. In other words, we need to be able to manipulate and use XMPP (XML-based) and RDF to capture, encode transform and generate IMs, detect semantic events and generate notification actions [9,21,4]. We enable these capabilities by relying on a complete mapping of XMPP constructs into RDF [7], a SPARQL-based continuous query processing engine for semantic event detection and aggregation [15,14], and a semantic query and transformation language that can deal with XML and RDF data (XSPARQL [1]).

The proposed UPMS architecture combines and applies the existing solutions for semantic information extraction from collaborative tools, continuous query processing and data transformation in the area of IM communication and based

[1] `http://www.juniperresearch.com/viewpressrelease.php?id=311&pr=248`

on the standard XMPP protocol, resulting in a flexible and interoperable solution for Ubiquitous Presence Management.

Our main contributions in this paper can be summarised as follows: (i) we identify the requirements for UPMS, (ii) we semantically enhance XMPP messages based on standard vocabularies for sensor data and personal information management, (iii) we design and implement an agile platform for ubiquitous presence management that can detect semantically relevant event and trigger actions over the standard XMPP communication channel solely by using SPARQL-like queries in both event detection (CQELS) and data transformation (XSPARQL), (iv) we implement and evaluate the requirements and the performances of our solution by using NFC-enabled smartphones for physical presence and calendar data for collaborative contextual presence.

Structure of the Paper: We identify the requirements for UPMS and emphasise on the motivation of our research by describing a general usecase scenario in Section 2. We present the conceptual design of our Ubiquitous Presence Management System in Section 3. Section 4, we demonstrate implementation and evaluation of a particular instance of the overall usecase scenario of Section 2. We position and advocate the importance of our work by comparing it with the existing work in Section 6 and discuss further steps and open challenges in Section 7.

2 System Requirements and Scenario

In order to achieve the general idea of semantically enhanced Ubiquitous Presence Management System, we identify a list of requirements, classified into three broader categories: user experience, interoperability and knowledge discovery. These requirements provide a guideline for the evaluation of performance, functionalities and potential impact of semantically enhanced UPMS.

User Experience: Mobile devices are now being widely used to provide and update the presence over the IM clients. Despite the rapid increase in performance of the mobile devices, user experience may suffer from applications designed to offer poor personalisation or unacceptable latency. User experience is thus strictly related to context awareness (or personalisation) and performances.

- **Context Awareness:** Presence-based services for mobile devices require regular updates because of the frequent change in the user's context. It is very problematic and cumbersome for the IM client users to regularly update the presence status after each and every change in their context. UPMS should have the ability to automatically detect user's context from diverse presence sources, and automatically update the presence status accordingly.
- **Performance:** Performance is a critical feature for UPMS relying on instant messaging clients because instant messaging applications are expected to perform within minimal time latency of realtime or near realtime, and the management of rich presence sources can add to this by making latency unacceptable for a positive user experience.

Interoperability: Interoperability is among one of the essential features for UPMS to be able to deal with incremental presence sources and to be applied to different scenarios. Different collaborative tools are designed independently to serve a specific purpose, however providing interoperability among these heterogeneous and autonomous tools is a challenging task.

- **Heterogeneity:** Presence management systems should not only have to deal with heterogeneous communication protocols e.g. XMPP, SIP etc., but should also be capable to provide an integrated view over heterogeneous data formats.
- **Extensilbility:** Rapid advancement in mobile technologies is resulting in greater interest for the design and development of new mobile applications for presence management. An ideal presence management system should be extensible to easily incorporate any new feature or support new types of collaborative tool.
- **Adaptability:** Adaptability refers to the ability of the system to adapt the changes in its environment without any or with minimal external involvement, which is an important non-functional requirement of UPMS.
- **Domain Independence:** Most of the existing presence management systems are aimed to address the problems related to any specific domain. Domain independence of the semantic presence management system will facilitate its reusability and utilisation in any domain.

Knowledge Discovery, Sharing and Linking: Semantic technologies provide a systematic way to interrelate heterogeneous information from diverse sources for better discovery. Different ontologies are defined to semantically annotate and interlink the data from various domains and ease data sharing and reuse.

- **Event Management and Notification:** Event management allows the system to identify relevant pieces of knowledge (events) and related actions to be preformed when such events occur. UPMS should facilitate its users in the registration of any event and specification of a set of actions to be performed on the occurrence of that particular event.
- **Recommendation:** *Mobile Recommender Systems* offer personalised, context-aware recommendation for the mobile device user by considering various parameters associated with the mobile device users. UPMS should be able to provide better (presence-aware) recommendations by combining text analysis and information extractions with semantics, e.g. for expert finding).

Ubiquitous Presence Management Scenario. Our company SmithCom is a very dynamic multinational corporation that has to deal with a many customers across the globe developing IT solutions and providing targeted customer support. The employees of the company are involved in a many meetings involving technical documentation, contracts, sales, IT support, financial planning and more. To optimise communication, collaboration and resource management, SmithCom had put in place an infrastructure that supports Ubiquitous Presence Management.

For a final review of the products beta release Alice organises a virtual meeting with Sean and the German lead developer Hans. Alice uses her presence-enhanced meeting scheduler to create the meeting, specify the list of attendants, set meeting priority, preferred time and meeting duration along with resources needed for the meeting, while the scheduler suggests a meeting slot according to current participants calendars. The UPMS checks for available resources and sends an email to book them for the meeting. If the required resources are not available, alternative meeting times are provided to Alice according to her availability and Sean and Hans calendars.

Ten minutes before the meeting starts, a reminder is generated for the attendees. The UPMS detects that Hans is not at his laptop, so he receives an SMS on his smartphone on his way to the office. In a similar way, Sean presence indicates he is in a priority telephone conference for another half an hour, so he receives the same reminder by email. Alice, as a meeting host, receives a summarisation of relevant documents, reports and recent communications associated with the upcoming meeting. Based on her preferences and semantic presence, a list of pointers to relevant documents is provided through the IM client.

When the meeting starts, the UPMS detects that Hans is on his way to the meeting room, and informs Sean and Alice that he will arrive in five minutes. During the meeting, one of Seans team members sends him an IM. According to Seans policy, based on Sean's users groups, the profiles of the people Sean is meeting with and his preferences, the IM system shows him as busy for his team members and the message is blocked until the end of the meeting.

During the meeting, Alice's secretary Claire receives a request by the CEO for a meeting. UPMS determines that Alice doesn't have her phone with her, since her physical location is different from her phones location, but she is close to her laptop, so the UPMS produces a non-intrusive alert on it and the CEO is provided with an estimated time for a call and a time slot is booked into his calendar.

Later, Claire needs Alice to sign some papers but the UPMS shows her busy in her office despite no meeting scheduled. The reason for this is that a sales representative together with a new customer dropped by Alices office, and the UPMS can detect the two employees in the room via their tags and that a conversation is going on among a set of people (sensors can measure the level of noise without transmitting the conversation), therefore she instructs the presence service to notify her when Alice is available.

3 Ubiquitous Presence Management System

In this section, we give an overview of the conceptual design of our proposed ubiquitous presence management system (UPMS). UPMS is capable of managing and maintaining unified presence after gathering information from various

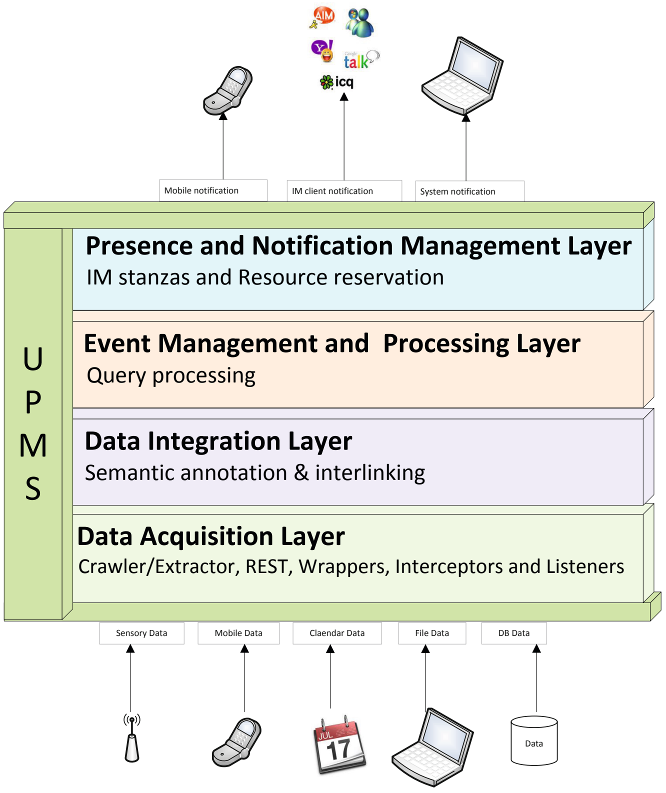

AIM
talk
icq
Mobile notification
IM client notification
System notification
UPMS
Presence and Notification Management Layer
IM stanzas and Resource reservation
Event Management and Processing Layer
Query processing
Data Integration Layer
Semantic annotation & interlinking
Data Acquisition Layer
Crawler/Extractor, REST, Wrappers, Interceptors and Listeners
Sensory Data
Mobile Data
Claendar Data
File Data
DB Data
JUL
17
Data

Fig. 1. Ubiquitous Presence Management System

data sources and sensors. It can also generate near realtime notifications which are triggered on the occurrence of any specific event. Figure 1 depicts a simplified architecture of the UPMS, demonstrating the inputs and outputs of the system. UPMS has ability to gather information from various collaborative tools, sensors, instant messaging clients, email clients or other collaborative tools which expose an interface to access their data. Once the information is acquired, it is semantically interlinked using various ontologies and live queries can be executed over the integrated data to monitor the occurrence of any event, and on the occurrence of that particular event, it can trigger various actions and notifications. UPMS consists of four processing layers, namely; (i) *Data Acquisition Layer*, (ii) *Data Integration Layer*, (iii) *Event Management and Processing Layer*, and (iv) *Presence and Notification Management Layer*. We discuss each of these layers and their underlying functionalities in detail below:

3.1 Data Acquisition Layer

The *Data Acquisition Layer* receives data as input to the UPMS. Data can be structured, unstructured and even raw data from the sensors. Data acquisition layer has to cope with heterogeneous data formats and data transmission protocols. UPMS in its current state provides four different mechanisms to receive data from multiple data sources,

(i) **Data Listeners:** Data Listeners continuously listen, intercept and filter the data from a communication channel and pick out the required information.
(ii) **RESTful Client:** Most of the collaborative tools including Google Calendar expose RESTful interfaces as a mean of communication with their clients. UPMS provides a RESTful client to collect data from the collaborative softwares and tools which support RESTful services.
(iii) **Crawlers & Extractors:** UPMS can acquire date from various sources including desktop integration applications and email services using crawlers and extractors.
(iv) **Data Mappers:** Data received from sensors is usually in raw format, UPMS contains data mappers to transform the sensors data into the desired format. In our usecase scenario, UPMS contains a mapper to map presence and geo-location into semantically interlinked data.

3.2 Data Integration Layer

UPMS relies on Semantic technologies to provide integrated access over heterogeneous data acquired from multiple sources. In our usecase, we semantically annotate the data from each source using predefined ontologies which provide a mechanism to semantically enrich the data of any specific source. SIOC (`http://rdfs.org/sioc/ns#`) and SIOCC (`http://rdfs.org/sioc/chat#`) provide semantic enrichment of the data related to the online presence, user accounts, instant messages, chat sessions and chat topics, which makes them ideal to semantically annotate XMPP messages received from the IM client. NCAL (`http://www.semanticdesktop.org/ontologies/ncal`) semantically annotates and interlinks meetings data described in calendars. SPITFIRE (`http://spitfire-project.eu/ontology/ns/`) allows to interlink and map various sensors data e.g geo-location. Once data is semantically enriched, the resulted RDF data is stored into an internal RDF data store within the UPMS. The data from this store can be queried using any specific query language defined to access linked data.

3.3 Event Management and Data Processing Layer

An event, which is often defined as "significant change in the state", can be registered at the event management component of the UPMS. Once an event is registered at UPMS, a continuous query is performed over the integrated data to check the occurrence of the event and as soon as the event is triggered, the event management component sends request to the presence and notification management component to perform the actions as defined for that particular event.

At the heart of the UPMS is a continuous query processor namely CQELS query engine [14], which queries live stream of data and integrated linked data. An event can be registered at UPMS by providing a CQELS query, which continuously monitors the occurrence of the event. The CQELS query filters live stream of data coming from various sources and combines it with the static interlinked data to find the desired query patterns. Upon the occurrence of the event, results of the query are generated and passed to *Presence and Notification Management Layer*, which is responsible to perform the actions defined in the event.

3.4 Presence and Notification Management Layer

Presence and Notification Management Layer is responsible to perform all the actions after the occurrence of the event. Results of the CQELS queries are passed by the *Event Management and Processing Layer* to the *Presence and Notification Management Layer* in the form of RDF triples. The presence and notification manager transforms the RDF triples into the required data format to disseminate the notifications. In our usecase scenario, we use RDF to XMPP mapping to generate notification messages (XMPP stanzas) for IM clients. Presence and notification manager is also capable of interacting with various tools and their underlying network protocols for the communication.

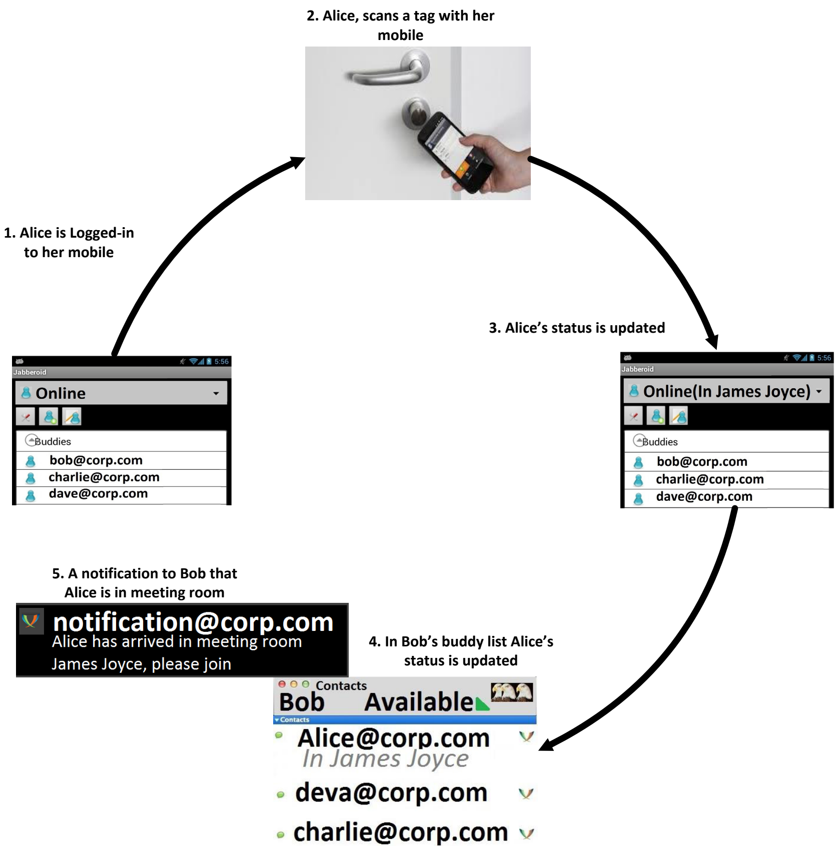
4 Implementation

As a proof of concept, we have implemented a usecase scenario of Section 2 using Cisco's Unified Presence System (CUP). CUP server is an XMPP based communication system to provide better means of communication within an enterprise [16]. In this section, we describe the NFC based usecase scenario for SmitCom and give a detailed description of UPMS implementation for this scenario.

4.1 Usecase Scenario

In order to facilitate localisation and presence management, employees of SmithCom are equipped with NFC-enabled smartphones running an NFC-enabled IM client, and meeting rooms are identified by a unique NFC tag. Resources are managed using IM messages for booking them in particular time slots for particular events.

The sales manager Alice schedules a meeting using *Gcal* in one week time, in room *James Joyce*. The meeting involves Bob, Charlie and Dave and needs a projector and a white board, which are available and reserved by the UPMS.


2. Alice, scans a tag with her mobile
1. Alice is Logged-in to her mobile
3. Alice's status is updated
Jabberoid
Online
Buddies
bob@corp.com
charlie@corp.com
dave@corp.com
Online(In James Joyce)
5. A notification to Bob that Alice is in meeting room
4. In Bob's buddy list Alice's status is updated
Contacts
Bob Available
Alice@corp.com
In James Joyce
deva@corp.com
charlie@corp.com

notification@corp.com
Alice has arrived in meeting room James Joyce, please join

Fig. 2. Usecase Scenario

Alice enters the meeting room *James Joyce*, fifteen minutes before the meeting, and she scans her mobile device against the wall-mounted NFC tag in the room. Her status on the IM client is automatically updated and made visible to her buddy list according to her privacy preferences. Being Alice the host of the meeting, she gets an IM message with the agenda for the meeting and a list of links to IM chats and emails related to the meeting, while the list of attendees (Bob, Charlie and Dave) receive an IM message advising that the meeting host has arrived and they should join the meeting in *James Joyce* room. The meeting can start perfectly on time with very little communication effort between the employees. Figure 2 depicts the proof of concept scenario described above.

4.2 System Architecture

Figure 3 represents the system architecture of the implemented usecase scenario, which depicts the residing components encapsulated by UPMS layer architecture

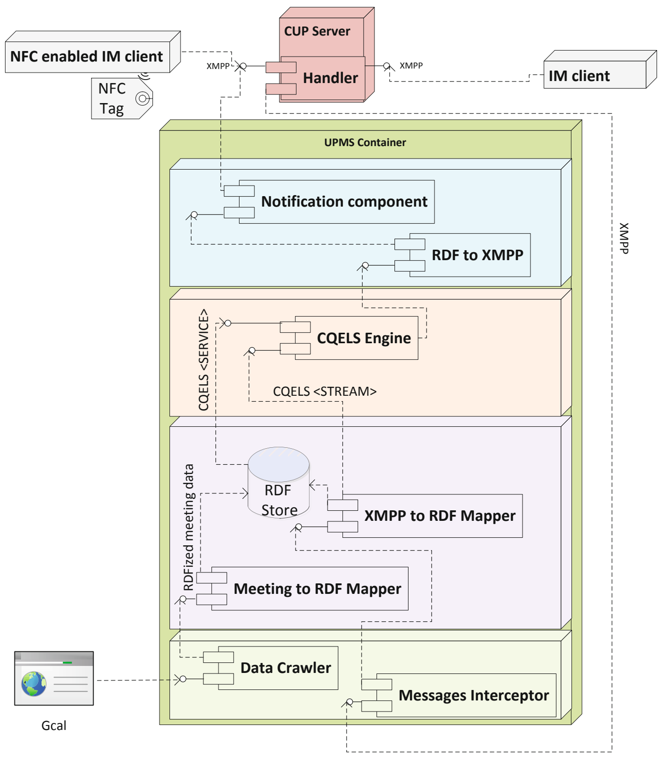


Fig. 3. System Architecture (Cisco's Unified Presence System for SmithCom)

as shown previously in Figure 1. UPMS has the flexibility to interact with multiple collaborative tools for data acquisition as well as generate the notifications to the respective targeted systems. In this prototype, we have demonstrated message interceptor for the XMPP stanzas sent from/to the CUP server, a meeting data crawler/extractor from Google Calendar (*Gcal*) for data acquisition, and a data mapper and notification dispatcher to disseminate notifications.

4.3 Process Flow

In this section the application process flow is explained describing all the steps involved to successfully execute the SmithCom usecase scenario as described previously. Each step is presented with a detailed description of the technologies involved.

Calendar Data Acquisition: Alice, a user is using a *Gcal* to schedule meetings. She is a registered user with <*alice@gmail.com*> account for *Gcal.* When

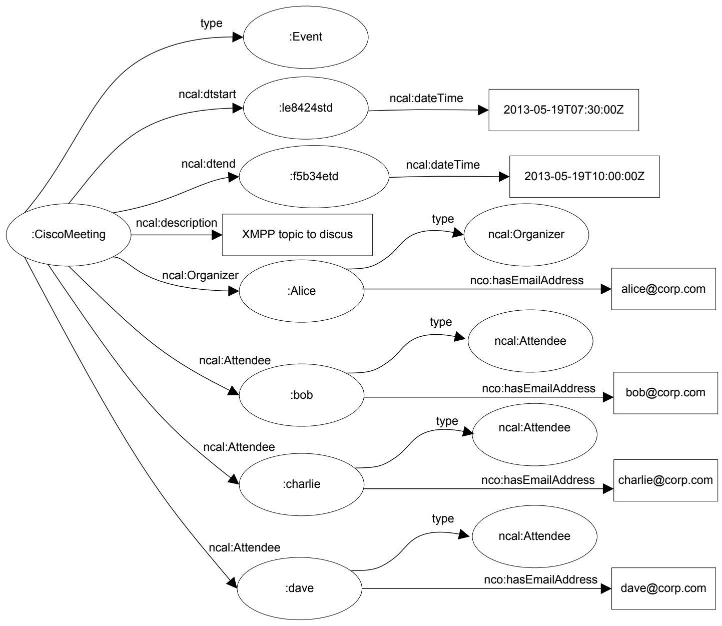


Fig. 4. Calendar Data Mapping

Alice, as a host, creates a meeting, the RESTFul client crawls the meeting data from *Gcal*.

Calendar Data Mapping: *Data Acquisition Layer* transfers the extracted data to the *Data Integration Layer*, where a *Meeting to RDF Mapper* transforms the meeting data into linked data by employing two ontologies defined for semantic desktop integration: (i) NCAL (`http://www.semanticdesktop.org/ontologies/ncal/`) to represent event extracted from the calendar, their start/end date and time, and participants in the event, (ii) NCO (`http://www.semanticdesktop.org/ontologies/nco/`), to represent persons and their e-mails addresses. Semantically interlinked data is stored in an RDF data store, which resides in *Data Integration Layer* of the UPMS. Figure 4 shows a semantic representation of a *Gcal* data for a meeting organised by Alice.

NFC-enabled IM Client: Our custom built NFC-enabled IM client has the ability to detect NFC tags. When a tag containing data in a supported format is tapped, the IM client acquires the data from NFC tag. The IM client generates two XMPP stanzas: (i) a *presence* stanza to update the user's status as acquired from NFC tag, and (ii) an *iq* stanza containing geo-location information acquired from the NFC tag. These stanzas fully comply with the existing standards defined in XMPP core and its XEP extension [19,10].

```
prefix ncal: <http://www.semanticdesktop.org/ontologies/
    2007/04/02/ncal#>
prefix nco: <http://www.semanticdesktop.org/ontologies/
    2007/03/22/nco#>

SELECT  ?attEmail ?starttime
WHERE {
         STREAM <http://deri.org/streams/lsm> [NOW]
         {
    <http://smithcominternal.com/example#aliceSmack>
    <http://www.semanticdesktop.org/ontologies/2007/03/22/nco#imID>
    ?host .        }

         SERVICE <http://lsm.deri.ie/sparql>
          {
         GRAPH <http://lsm.deri.ie/cisco/eventdata#>
          {
              ?E ncal:organizer ?organizer .
              ?organizer nco:hasEmailAddress ?hasemail .
              ?hasemail    nco:emailAddress ?host .
              ?E ncal:attendee ?attendee .
              ?attendee nco:hasEmailAddress ?email .
              ?email   nco:emailAddress ?attEmail .
              ?E ncal:dtstart ?start .
              ?start  ncal:dateTime ?starttime .
              Filter(str(?starttime)="2013-05-19T07:30:00Z")
         }
         }
    }
```

Listing 1. A Sample CQELS Query

XMPP Interceptor: In our usecase scenario, UPMS is implemented as an extension of the Cisco's CUP server, which is responsible for the information exchange between IM clients. *Message Interceptor* is responsible for intercepting and filtering the XMPP stanzas sent to/from the CUP server. The filtered XMPP stanzas are then passed on to *Data Integration Layer* of the UPMS for further processing.

XMPP to RDF Mapping: *XMPP to RDF Mapper* performs the mapping of the received XMPP stanzas using various ontologies: (i) OPO (`http://online-presence.net/opo/spec/`) to represent online presence information, (ii) SIOC (`http://rdfs.org/sioc/spec/`) and SIOCC (`http://rdfs.org/sioc/chat#`) to represent instant messages and chat messages, (iii) NCO (`http://www.semanticdesktop.org/ontologies/2007/03/22/nco#`) to represent user accounts, (iv) SPITFIRE (`http://spitfire-project.eu/ontology/ns#`) to represent information received from sensors, and (v) wgs84_pos (`www.w3.org/2003/01/geo/wgs84_pos#`) to represent geo-location information contained within XMPP messages. Figure 5 shows a pictorial representation of the mapping of *presence* stanza while Figure 6 shows mapping of *iq* stanza. XS-PARQL (a query language uses to transform the XML into RDF and vice versa) is used to transform the live stream of XMPP messages into RDF on the fly.

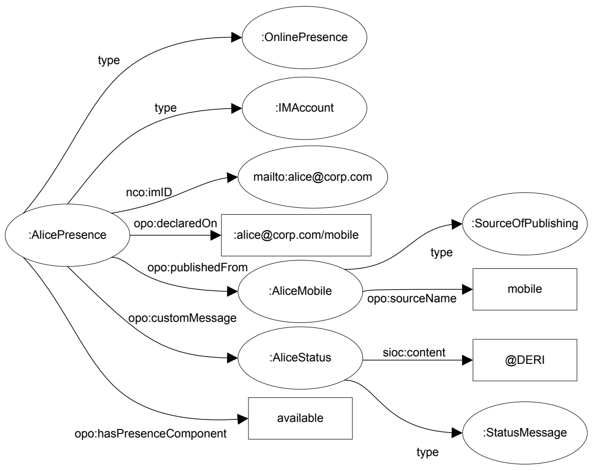


Fig. 5. Presence Stanza Mapping

Integrated Data Processing: CQELS queries perform integrated data processing by performing combined query over the live streams of mapped stanzas and integrated meeting data. UPMS in its current state expects from its users to provide a CQELS query to monitor the occurrence of any event. However, providing an interactive user interface for the event registration and generating CQELS queries automatically is part of our future agenda. Listing 1 shows a CQELS query which monitors the occurrence of our usecase scenario event. CQELS query contains two clauses, (i) STREAM clause filters XMPP messages to find out host of the meeting and (ii) SERVICE clause, queries the integrated data to look for a particular meeting organised by host, meeting start time and list of attendees.

Presence and Notifications Management: CQELS queries produce results on the occurrence of the event, which are processed at *Presence and Notification Management Layer* to achieved the desired outcomes as described in the event. In our usecase scenario, we used XSPARQL to transform the results of CQELS queries into XMPP messages, which are disseminated to all attendees of the meeting.

5 Evaluation

We evaluated the performance of our system using two approaches: (i) by comparing our system with the state of the art presence management systems, and (ii) by evaluating the performance of our system with the increasing number of users (meeting attendees).

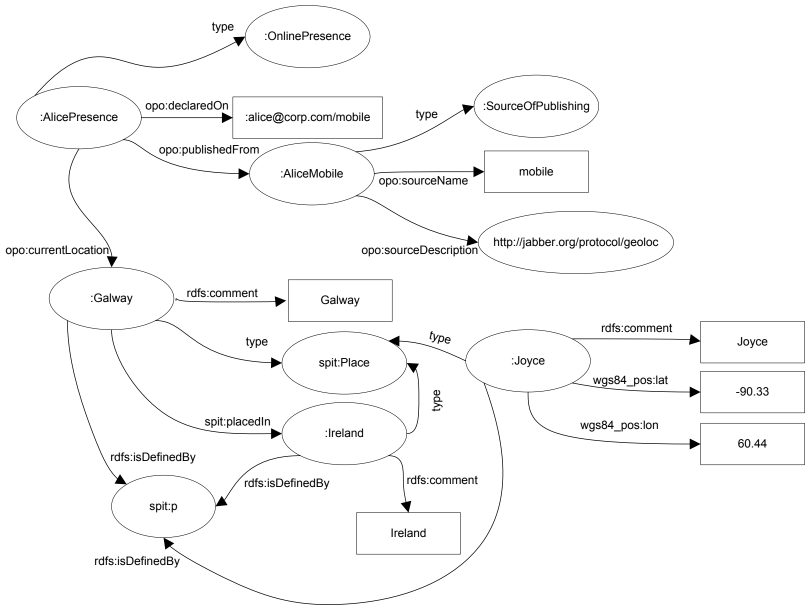


Fig. 6. IQ Stanza Mapping

5.1 Comparison

We used system requirements described in Section 2 as a guideline for the evaluation of functionalities provided by the UPMS. In Table 1, we compare our system with three state of the art presence management systems, namely (i) *Loca-Tag* (ii) *R-U-In?*, and (iii) *Follow Me!*. *Loca-Tag* is aimed to provide automatic update in the presence of the IM client's (Skype) user, reflecting the current location of the user observed using NFC tag [13]. *Loca-Tag* provides rich user experience but lacks interoperability and knowledge discovery features. *R-U-In?* leverages contextual information to provide better "social search" for heterogeneous social networking tools including IM [4]. However, it is domain dependent on social networks and also does not support event management and adoptability. *Follow me!* provides rich presence information by consolidating virtual and physical presence using contextual information [3]. Similar to the previous two presence management applications, *Follow me!* is also limited to provide rich user experience while falling short in providing interoperability and knowledge discovery. Contrary to the previously discussed systems, UPMS is not only aimed on providing rich user experience but it is also equipped with interoperability and knowledge discovery.

Table 1. System Requirements Comparison

Category	System Req.	UPMS	Loca-Tag	R-U-In?	Follow Me!
User Experience	Performance	✔	✔	✔	✔
	Context Awareness	✔	✔	✔	✔
Interoperability	Adaptability	✔	✘	✘	✘
	Extensibility	✔	✘	✘	✘
	Heterogeneity	✔	✘	✔	✘
	Domain Independence	✔	✘	✘	✘
Knowledge Discovery	Event Management	✔	✘	✘	✘
	Recommendations	✔	✘	✔	✘

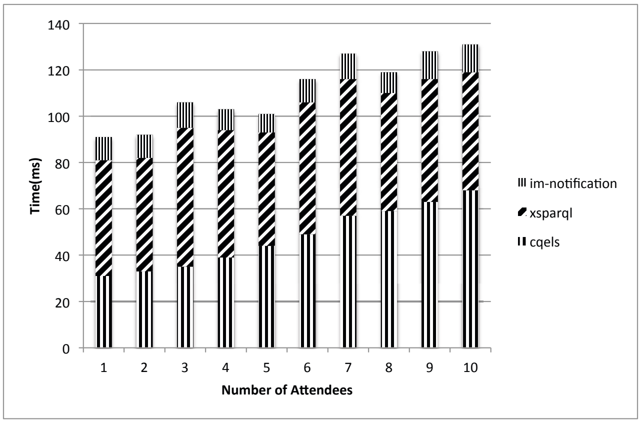


Fig. 7. Processing time in proportion to the number of attendees

5.2 Performance Evaluation

In order to evaluate the performance of our system we setup the testbed to implement the usecase scenario of Section 4. The processing time of the application is subdivided as (i) *XMPP to RDF Mapping Time* (XSPARQL), (ii) *CQELS Processing Time*, (iii) *RDF to XMPP Mapping Time* (XSPARQL) , and (iv) *Notification Dissemination Time*. We vary the number of attendees of the meeting from 1 to 10 and calculated the average processing time after 20 observations. Figure 7 shows the average processing time of the notification dissemination to all the attendees of a particular meeting generated after the arrival of the meeting host. We omit *XMPP to RDF Mapping Time* in Figure 7, because it was evident that *XMPP to RDF Mapping Time* is dependent on the size of the presence and iq stanza irrespective of the number of attendees. Average time of the 20 observations for *XMPP to RDF Mapping Time* for *presence* stanza and *iq* stanza was 136ms and 163ms respectively. As shown in Figure 7, *CQELS Processing Time* and *RDF to XMPP Mapping Time* increases with the increase in number of attendees. However, considering the existing time latency, UPMS is scalable to disseminate notification up to 100 attendees within a delay of less than a second. It is also worth mentioning that *Notification Dissemination Time* is also increasing with the number of attendees, however not very substantially due to better performance of the CUP server.

6 Related Work

In the last few years, NFC based applications are gaining popularity and many applications have been developed and implemented using NFC technology. A generalised approach for the design and development of NFC applications have been described in [6,5]. The state of the art and future directions for the NFC applications have been discussed in [18]. However, currently most of the existing applications provide domain specific and small scale solutions e.g mobile payment systems, e-ticketing systems or healthcare systems [22,12,20]. Loca-Tag enhances instant messaging tools to provide near real time location based presence sharing through the use of NFC enabled mobile devices [13]. A prototype developed in Loca-Tag sends the presence information for Skype[2] using a Web service. The trend of presence systems, which were developed earlier to serve presence sharing or instant messaging, have been shifted to intelligent presence management in a collaborative environment [11].

In [9], a holistic definition of presence, which they call "Consolidated Presence", has been presented. Consolidated presence is achieved by deploying richer semantic services to combine several presence information from various heterogeneous and dynamic sources. In order to receive richer semantic presence in Cisco's CUP server, a mapping of the core XMPP messages into RDF is formally defined in [7]. All XMPP messages are mapped into RDF using various ontologies e.g. (i) SIOC (`http://rdfs.org/sioc/ns#`) represents user account, IM messages and threads, topics and chat channels, (ii) OPO (`http://online-presence.net/opo/spec/`) represents dynamic aspects of user's presence, (iii) NCAL (`http://www.semanticdesktop.org/ontologies/ncal/`) semantically annotates calendar events, their start and end date, topic, host name, and list of participants, (iv) NCO (`http://www.semanticdesktop.org/ontologies/nco/`) represents known persons and their email addresses, and (v) SPITFIRE (`http://spitfire-project.eu/ontology/ns/`) semantically represents sensor network and its surrounding environment.

However, to the best of our knowledge, there is no existing system which can utilise NFC-enabled mobile devices to update location based presence information for the IM clients and can also integrate the information from various heterogeneous collaborative tools to provide a complete semantically interlinked notification system for ubiquitous presence management.

7 Conclusion and Future Work

In this paper, we have proposed, deployed and evaluated a semantically enriched solution for Ubiquitous Presence Management. This work is pioneering

[2] `www.skype.com`

next generation communication platforms for the Smart Enterprise, and it is based on i) semantic enrichment of XMPP stanzas to provide a holistic view of heterogeneous presence sources (including sensor data) and knowledge from other collaborative tools, ii) continuous event detection based on CQELS semantic query processing, iii) query-driven data integration and transformation from XML to RDF and back via XSPARQL, and iv) query-driven generation of notifications triggered by appropriate events. We showcase our prototype using NFC-enabled devices and calendar events to dynamically detect meeting status and progress, and notify attendees accordingly, and we present an evaluation based on the implemented proof of concept.

Next steps for the realisation of a more comprehensive Ubiquitous Presence Management System will include additional collaborative tools and learn user profiles that can be used to suggest interesting events and to configure a personal Ubiquitous Presence Management Dashboard. Leveraging Linked Data, in particular form Social Networks and DBPedia is also an interesting direction for automatic context characterisation. These aspects together with the characterisation of the interplay between physical and collaborative contexts will be investigated in future steps.

References

1. Akhtar, W., Kopecký, J., Krennwallner, T., Polleres, A.: XSPARQL: Traveling between the XML and RDF Worlds - and Avoiding the XSLT Pilgrimage. In: Bechhofer, S., Hauswirth, M., Hoffmann, J., Koubarakis, M. (eds.) ESWC 2008. LNCS, vol. 5021, pp. 432–447. Springer, Heidelberg (2008)
2. Ali, M.I., Lopes, N., Friel, O., Mileo, A.: Update Semantics for Interoperability Among XML, RDF and RDB. In: Ishikawa, Y., Li, J., Wang, W., Zhang, R., Zhang, W. (eds.) APWeb 2013. LNCS, vol. 7808, pp. 43–50. Springer, Heidelberg (2013)
3. Antonic, A., Slivar, I., Zarko, I.P.: Follow me – A rich presence application for smartphones. In: 2012 20th International Conference on Software, Telecommunications and Computer Networks, SoftCOM, pp. 1–5. IEEE (2012)
4. Banerjee, N., Chakraborty, D., Dasgupta, K., Mittal, S., Nagar, S., et al.: Ru-in?-exploiting rich presence and converged communications for next-generation activity-oriented social networking. In: Tenth International Conference on Mobile Data Management: Systems, Services and Middleware, MDM 2009, pp. 222–231. IEEE (2009)
5. Benyó, B., Sódor, B., Fördos, G., Kovácps, L., Vilmos, A.: A generalized approach for NFC application development. In: Proc. of the Second International Workshop on Near Field Communication, NFC, pp. 45–50. IEEE (2010)
6. Benyo, B., Vilmos, A., Kovacs, K., Kutor, L.: The Design of NFC Based Applications. In: Proc. of 11th International Conference on Intelligent Engineering Systems, pp. 277–280 (2007)
7. Dabrowski, M., Scerri, S., Rivera, I., Leggieri, M.: Dx–Initial Mappings for the Semantic Presence Based Ontology Definition (November 2012), http://www.deri.ie/publications/technical-reports/

8. Dumitrache, A., Mileo, A., Zimmermann, A., Polleres, A., Obermeier, P., Friel, O.: Enabling privacy-preserving semantic presence in instant messaging systems. In: Beigl, M., Christiansen, H., Roth-Berghofer, T.R., Kofod-Petersen, A., Coventry, K.R., Schmidtke, H.R. (eds.) CONTEXT 2011. LNCS, vol. 6967, pp. 82–96. Springer, Heidelberg (2011)
9. Hauswirth, M., Euzenat, J., Friel, O., Griffin, K., Hession, P., Jennings, B., Groza, T., Handschuh, S., Zarko, I.P., Polleres, A., Zimmermann, A.: Towards Consolidated Presence. In: Proc. of CollaborateCom 2010, pp. 1–10 (2010)
10. Hildebrand, J., Saint-Andre, P.: Xep-0080: User location. XMPP Standards Foundation (2007)
11. IJsselsteijn, W., van Baren, J., van Lanen, F.: Staying in touch: Social presence and connectedness through synchronous and asynchronous communication media. In: Human-Computer Interaction: Theory and Practice (Part II), vol. 2, pp. 924–928 (2003)
12. Juntunen, A., Luukkainen, S., Tuunainen, V.K.: Deploying NFC Technology for Mobile Ticketing Services–Identification of Critical Business Model Issues. In: Proc. of Ninth International Conference on Mobile Business, pp. 82–90. IEEE (2010)
13. Köbler, F., Koene, P., Krcmar, H., Altmann, M., Leimeister, J.: LocaTag - An NFC-Based System Enhancing Instant Messaging Tools with Real-Time User Location. In: Second International Workshop on Near Field Communication, NFC, pp. 57–61. IEEE (2010)
14. Le-Phuoc, D., Parreira, J.X., Hausenblas, M., Hauswirth, M.: Continuous query optimization and evaluation over unified linked stream data and linked open data. Technical Report DERI-TR-2010-09-27, DERI, IDA Business Park, Lower Dangan, Galway, Ireland (2010)
15. Le-Phuoc, D., Quoc, H.N.M., Parreira, J.X., Hauswirth, M.: The linked sensor middleware–connecting the real world and the semantic web. In: Proceedings of the Semantic Web Challenge (2011)
16. Morgan, B., Lisenbea, S., Popovich, M.: Cisco Unified Presence Fundamentals. Cisco Systems (2010)
17. Nardi, B.A., Whittaker, S., Bradner, E.: Interaction and outeraction: instant messaging in action. In: Proc. of CSCW, pp. 79–88. ACM (2000)
18. Ok, K., Coskun, V., Aydin, M.N., Ozdenizci, B.: Current benefits and future directions of NFC services. In: 2010 International Conference on Education and Management Technology (ICEMT), pp. 334–338. IEEE (2010)
19. Saint-Andre, P.: Extensible messaging and presence protocol (xmpp): Core (2011)
20. Vergara, M., Díaz-Hellín, P., Fontecha, J., Hervás, R., Sánchez-Barba, C., Fuentes, C., Bravo, J.: Mobile prescription: An NFC-based proposal for AAL. In: Proc. of Second International Workshop on Near Field Communication, NFC, pp. 27–32 (2010)
21. Voida, S., Mynatt, E.D., MacIntyre, B., Corso, G.M.: Integrating virtual and physical context to support knowledge workers. IEEE Pervasive Computing 99(3), 73–79 (2002)
22. Zou, J., Zhang, C., Dong, C., Fan, C., Wen, Z.: Mobile payment based on RFID-SIM card. In: Proc. of IEEE 10th International Conference on Computer and Information Technology, CIT, pp. 2052–2054. IEEE (2010)

Semantic Measures Based on RDF Projections: Application to Content-Based Recommendation Systems

(Short Paper)

Sébastien Harispe, Sylvie Ranwez, Stefan Janaqi, and Jacky Montmain

LGI2P, Ecole des mines d'Alès, Parc Scientifique G. Besse, F-30035 Nîmes Cedex 1
`firstname.name@mines-ales.fr`

Abstract. Many applications take advantage of both ontologies and the Linked Data paradigm to characterize various kinds of resources. To fully exploit this knowledge, measures are used to estimate the relatedness of resources regarding their semantic characterization. Such semantic measures mainly focus on specific aspects of the semantic characterization (e.g. types) or only partially exploit the semantics expressed in the knowledge base. This article presents a framework for defining semantic measures to compare instances defined within an RDF knowledge base. A special type of measure, based on the representation of an instance through projections, is detailed and evaluated through its use in a music band recommender system.

Keywords: Semantic Measures, Semantic similarity/relatedness, RDF Projection, Content-based Recommendation Systems, Similarity between instances.

1 Introduction

"Which music bands are similar to the Rolling Stones?" It would be quite natural to ask such a question to a friend with some knowledge of music. Most classical search engines however will fail to provide an answer, since it refers to resources defined as music bands (notion of *type*) and must be related to the 'Rolling Stones' (notion of semantic relatedness). Answering such questions extensively relies on similarity evaluations to formulate recommendations such as: "If you like the Rolling Stones, you might also like…". So, how is it possible to define whether or not two music bands are related by studying their properties and more generally, how can the degree of relatedness of two instances be assessed? *Data Retrieval* techniques based on an exact search cannot be used herein; the inaccuracy expressed by the query entails considering imprecise results and therefore requires the use of *Information Retrieval* (IR) techniques.

Taking advantage of Semantic Web technologies and the Linked Data paradigm to assess the semantic relatedness of entities is not new. Measures that are used in this context often compare pairs of (groups of) classes and only a few can be used to compare instances. Moreover, an instance is often represented by a reductive canonical form (bag of classes or concepts) [1]. Two main approaches have been proposed to estimate the degree of relatedness of instances defined in an RDF knowledge base: a direct one that controls the semantic model associated with the knowledge base [2], and an indirect one that does not consider or just slightly considers these semantics [3]. However, especially in *Recommendation System* (RS), such measures must exploit semantics and enable justifying why a strong/weak semantic relatedness between instances is being assessed.

Ehrig et al. [4] first propose a framework for defining ontology-based semantic measures to compare instances through their direct properties. It has been extended to integrate imprecise evaluations of direct properties into SPARQL [5]. Based on the concept of a *similarity aggregation operator*, they defined a strategy from which a complex element defined in an RDF graph may be compared on the basis of multiple similarity measures and an aggregation scheme [5]. In some cases however, instances can only be compared by incorporating their indirect properties, e.g. information relative to properties characterizing the instances they are related to. To bridge this gap, Albertoni and De Martino proposed to include an evaluation of indirect properties as a means of better estimating the relatedness of instances [6].

The present contribution extends existing frameworks by defining a canonical form based on the notion of *RDF projection*. This approach enables a fine-tuned characterization of the representation of instances according to specific use contexts. Moreover, our approach enables to express complex indirect properties not taken into account in existing frameworks. This representation of an instance is ultimately used to define a parameterized semantic measure for RS definitions.

2 Semantic Measures for IR and Recommendation Systems

This section presents the formal notations used to represent an RDF graph and introduces the reader to semantic measures that better fit the needs of IR and RSs.

2.1 RDF Knowledge Base

Inspired by Semantic Web technology developments, a growing number of industrial, governmental and academic organizations adopt RDF graphs to store their knowledge. This representation offers multiple advantages, in particular due to the explicit definition of the relationships established among the resources expressed in this graph. However, RDF graph expressivity and exploitability are maximal if the associated semantic has been defined through ontologies (RDF-Schema and OWL are used therefore). The semantic graph can therefore be queried by a query language (e.g. SPARQL), often based on sub-graph matching. However, in the context of IR or RSs the user is seeking to interact with the system under fewer constraints, e.g. by conducting inexact/imprecise searches, e.g. for music bands *similar* to one another.

To simplify the technical presentation of our proposal, we consider an RDF(S) knowledge base as graph $G = (V, R)$, in which $V = C \cup I \cup D$ is the set of vertices composed of classes C, instances I, vertices D associated with various types of data (e.g. strings), and set of relationships $R \subseteq C \times C \cup I \times I \cup I \times D \cup I \times C \cup C \times D$. In the example illustrated in Figure 1, classes represent the concepts defined in an ontology related to music: *Music Band*, *Music Genre*, etc., while the instances are music bands: *The Rolling Stones*, music genres: *rock*, etc. Moreover, a given instance can also establish specific relationships with other instances or data (e.g. a literal corresponding to the name of the band). An RDF knowledge base can therefore be decomposed according to: i) the intensional layer (ontologies, classes), ii) the extensional layer (instances), and iii) the data layer. This paper has adopted the RDF terminology with a preference for the term *class* over the term *concept* commonly used in IR.

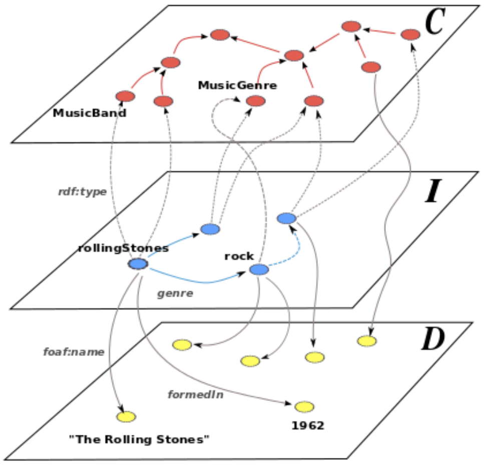


Fig. 1. Representation of an RDF knowledge base according to three layers: intensional (I), conceptual (C), and data (D)

A semantic measure enables estimating the similarity or proximity of semantic elements (e.g. words, terms, classes) or instances semantically characterized (e.g. documents annotated by classes) while taking into account the semantic space in which they have been defined (text corpus, ontologies). We focus here on semantic measures relying on knowledge defined within a semantic graph. Two types of graph-based semantic measures can be distinguished depending they aim to compare classes or instances. We here focus on semantic measures dedicated to instance comparisons.

2.2 Semantic Measures between Instances

Semantic measures between instances have been widely studied to perform instance matching in various knowledge bases, e.g. RDF and databases [7]. The aim is to detect duplicated instances in one or more knowledge bases. In addition, semantic measures have also been used to discover relationships between instances [8].

Evaluating the proximity between instances requires defining a representation (or canonical form) to characterize an instance. Four approaches can be distinguished:

- **Representing an instance as a graph vertex.** The instance is represented through the vertex of the graph making reference to it. The proximity between two instances is therefore evaluated using measures exploiting graph structure analysis and does not explicitly rely on the semantic carried by the graph: the more the compared instances are interconnected the more related they will be assumed [3].
- **Representing an instance using a set of classes.** The instance is associated with its set of affiliated and possibly weighted classes. The measures defined to compare sets of classes can then be used for this canonical representation. Such a canonical form remains too restrictive for representing instances defined in an RDF knowledge base since only the types of instances will be considered. In Figure 1, the instance *rollingStones* would therefore be reduced to its set of affiliated classes (e.g. *MusicBand*).
- **Representing an instance through a list of properties.** An instance can be evaluated by studying its direct properties. Two types of properties may be distinguished: non-taxonomical (*object* and *datatype properties* in OWL); and taxonomical, i.e. those involving classes. Datatype properties can be compared using measures adapted to the type of properties considered, e.g. in using a measure to compare dates of music band formations. Scores produced by the various measures are thus aggregated to obtain a global score of similarity [7]. Such a representation is commonly adopted in ontology alignment, instance matching or link discovery between instances, e.g. SemMF [2] and SILK [8].
- **Representing an instance through an extended list of properties:** This representation is an extension of the canonical form previously presented. It can be implemented to take into account indirect properties of instances, i.e. properties induced by the resources associated with the represented instances. In reflecting on our music-related example, such a representation might be used to consider the characteristics (properties) of the music genres for the purpose of comparing two music bands.

Several frameworks have been proposed to compare instances. Comparison based on direct properties were first characterized by Ehrig et al. [4]. This framework has next been extended to capture some of the indirect properties [6] through a path in the graph. From a different perspective, Andrejko and Bieliková [9] suggested an unsupervised approach for comparing a pair of instances by considering their indirect properties. Each direct property shared between the compared instances plays a role in computing the global similarity. When the property corresponds to an object property, the approach combines a taxonomical measure with a recursive treatment processing the properties of instances associated with the instance being processed. Lastly, in estimating the similarity between two instances, the measure aggregates the scores obtained during the recursive process.

2.3 Semantic Measure Specificities for Recommendation Systems (RS)

The purpose of a RS is to propose relevant resources to users in accordance with a context and their specific interests. A RS relies on three components: i) the knowledge base (resources and knowledge model), ii) information characterizing system users, and iii) an algorithm for exploiting components i) and ii) in order to produce recommendations [10]. The Linked Data paradigm and ontologies have both been recently proven particularly well suited for defining such systems [11].

Despite the existence of numerous approaches for defining RSs [10], this paper focuses on the *content-based* approach that relies on resource properties. In most cases, RSs are fine-tuned by experts possessing an in-depth understanding of the knowledge model and who are capable of distinguishing the properties to be taken into account to parameterize the RS. The representation of an instance as an extended list of properties seems to be the most appropriate in this context.

The framework proposed by Albertoni and De Martino [6] enables use of indirect properties of instances to define semantic relatedness measures but does not take complex indirect properties into account, i.e. properties that rely in combining various other properties. Moreover, it cannot be used to exploit characterizations of various types of instances. To address this limitation, Andrejko and Bieliková [9] proposed a recursive process on those instances but that cannot be used to define the direct and indirect properties to take into consideration.

The direct and indirect properties to be considered when comparing two instances depend to the usage context. The expressivity of existing frameworks merely enables partially characterizing an instance defined in an RDF knowledge-base. The difficulty lies in expressing indirect properties and the impossibility of evaluating complex (in)direct properties limits the definition of semantic measures. To remedy this shortcoming, the next section introduces a new framework for defining semantic measures.

3 A Framework for Defining Semantic Measures That Compare Instances of an RDF Knowledge Base

3.1 Characterizing an Instance through Projections

A direct or indirect property of instance i corresponds to a partial representation of i. In Figure 2, the *rollingStones* instance can be represented by its name or music genres. A *simple property* of an instance is therefore expressed through resources linked to it. Representing an instance through its labels is therefore the same as considering all the l labels for which a path links i to l through the relationship $rdf{:}\,label$; in other words, a triplet $< i, rdf{:}\,label, l >$ exists. In a general manner, the path linking two resources is characterized by an ordered list of relationships $r_0/r_1/\ldots/r_n$, with $r_i \in R$, (the *property path* in SPARQL 1.1). Like for a property, a path is also associated with a range defined according to the range of r_n, its last relationship. Let's distinguish three types of paths: i) *Data*: the range is a set of data, e.g. Strings, Dates (Fig. 2, case 2); ii) *Instances*: the range is a set of instances (Fig 2, case 1); and iii) *Classes*: the range is a set of classes (Fig 2, case 3).

Complex properties require several paths in order to be expressed: e.g. to compare two music bands through the Euclidian distance between their places of origin requires two paths $\{hometown/geo{:}lat,\ hometown/geo{:}long\}$ (Fig. 2, case 4). In order to characterize all properties of an instance, the notion of path can thus be generalized by introducing the notion of projection.

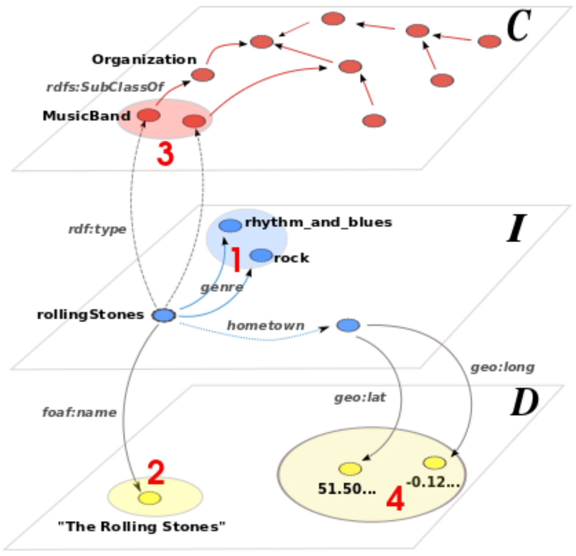


Fig. 2. Examples of properties associated with the *MusicBand* class

A projection refers to projecting a mathematical structure from one space to another. A projection P is composed of a set of paths and defined by $P{:}\,I \rightarrow K$, with K being the set defining the types of projection $k \in K$, onto which an instance can be *projected*. For simple projections, composed of a single path, the range corresponds to the path range. For complex projections, the range depends on the complex objects representing the complex properties of an instance. Let's note that complex objects are used to represent properties not explicitly expressed in the knowledge base. Four broad types of projections can therefore be distinguished: the three capable of being associated with a single path (*Data, Instances, Classes*), and the *Complex* type used to represent an instance by means of a set of complex objects. Let's denote P^k the projection of range $k \in K$ and $P^k(i)$ the type k projection of instance i.

A set of projections called the *context of projection* CP^c can be associated with class c. This context defines the representation of an instance of this class. It thus enables distinguishing the various properties of interest for characterizing an instance.

3.2 Semantic Measures That Take Advantage of Projections

The proximity of two instances is evaluated based on the context of projection of the class of affiliated instances, by taking into account all projections composing this context. Each projection is associated with a measure σ^k that enables comparing a pair of instance projections of type k, where $\sigma^k: k \times k \to [0,1]$.

Two *Classes* type projections can be compared using a semantic measure adapted to a comparison of classes. A comparison of *Data* type projection requires defining a measure adapted to the type of values, e.g. the Levenshtein distance for Strings. *Instances* type projections can be compared using set-based measures and *Complex* projections require defining a measure to enable comparing two complex objects.

Once a measure has been chosen to compare each projection, a general semantic measure σ_c can be defined between two instances u and v of type c:

$$\sigma_c(u,v) = \sum_{P_i^k \in CP^c,\ \exists P_i^k(u) \wedge \exists P_i^k(v)} w_i \times \sigma^k\left(P_i^k(u), P_i^k(v)\right) \tag{1}$$

where w_i is the projection weight associated to P_i^k and the sum of weights equals 1. This measure exploits each projection shared between the compared instances.

As previously observed, a projection defines a set of resources that characterize a specific property of an instance. To estimate the similarity of two instances relative to a specific projection, a measure σ^k must be specified to compare two sets of resources. When an indirect method is used to compare two projections, a measure enabling the comparison of two sets of resources needs to be defined. The relevance of a measure is defined by both the use context and the semantics of the similarity scores.

Two groups of instances can be compared by using a direct or indirect approach; an example is provided in the next section. When an indirect approach is selected, it is possible to use the context of projection defined for the class of the two instances under comparison. This context defines the properties that must be taken into account when comparing two instances of this specific type. Applying such a strategy potentially corresponds to a recursive treatment, for which a stop condition is required. In all cases, computing the proximity of two projections should not imply use of the context of projection containing both projections. A proximity measure can thus be represented through an execution graph highlighting the dependencies occurring between contexts of projection. Consequently, this execution graph must be analysed to detect cycles for the purpose of ensuring computational feasibility. If a cycle is detected, the measure will not be computable.

3.3 Framework Extensions

The partial ordering of classes can be exploited to enhance the characterization of an instance according to the projections associated with its inferred classes. It might be worthwhile therefore to provide projection overloading mechanisms depending on the partial ordering (note the drawback of multiple inheritances), or to define contexts of projection that characterize subsets of instances not framed in specific classes (e.g. a set of instances returned by a SPARQL query). The proposed framework thus enables easily comparing instances of a class based on the fine-tuned characterization of their properties. The instances of different classes can be compared according to projections shared by the least common ancestors of their classes, i.e. projections characterizing the more concrete and similar affiliated classes. Such a strategy however features certain drawbacks in the context of a relatedness evaluation since only instances of similar classes will tend to obtain high relatedness scores. This is because the global measure is solely driven by the property (feature) comparison of the targeted instances. In some use contexts, instances of various types are in fact expected to show high relatedness. These specific dimensions of relatedness can only be captured by measures evaluating the structural properties of the graph, i.e. the interlinking of instances, and must therefore be framed in a graph-theoretic model, e.g. [3]. The definition of a context of projection can therefore be relaxed in order to include interlinking metrics. Another approach would be to extend the notion of projection to represent an instance through abstract properties processed using measures evaluating interlinking, e.g. instances could be represented through their induced graph (weighted according to distance) so as to take greater advantage of measures based on graph diffusion distances and interlinking analysis.

4 Application to the Definition of Recommendation Systems

The proposed framework enables expressing semantic measures to compare instances defined in an RDF knowledge base. This approach is particularly well suited to defining semantic measures for the design of content-based RSs.

This section will present an example of how to use the framework to define a music band RS based on an RDF knowledge base. The specific RDF base employed has been built from DBpedia [12] and Yago2 [13]. Other examples of Linked Data use for the purpose of deriving music RSs can be found in [11, 14, 15]. The aim of the system proposed herein is to recommend music bands.

This RS relies on a relatedness measure between two instances of the class *MusicBand*. In considering the *target band*, e.g. "*The Rolling Stones*", the RS proposes related music bands to the user. The higher the relatedness score of a music band with the target band, the more relevant this band becomes for the recommendation. This relatedness measure is defined using two contexts of projection associated with the classes *MusicBand* and *MusicGenre*.

$CP^{MusicBand}$ is composed of three projections dealing with: i) their names, ii) their types (e.g. Yago2 affiliated classes), iii) the distance of their place of formation (complex property built from latitude and longitude), and iv) the proximity of their related music genres. Projection (i) corresponds to the maximum similarity obtained using a Levenshtein distance. Projection (ii) is evaluated using a measure that enables comparing groups of classes using the taxonomical structure of ontologies. Projection (iii) relies on a basic distance of points in a sphere. Projection (iv), related to the music genres associated with the music bands, is based on an average type of aggregation strategy; the measure used to compare two music genres relies on the context of projection defined for the class *MusicGenre*.

$CP^{MusicGenre}$ is composed of two simple projections based respectively on the labels associated with music genres and the structuration defined by the *subgenre* relationship that establish a partial ordering among the various music genres. The measures adopted here are similar to projections (i) and (ii) above.

To distinguish the relevant music bands when considering the target band, the four projections composing $CP^{MusicBand}$ are evaluated. To proceed, a vector containing the relatedness of the target band with other music bands is computed for each projection. Computing all vectors in order to provide recommendations for a single group takes 1 second using our implementation based on the Semantic Measures Library (http://www.semantic-measures-library.org) using a 2Go RAM personal computer. A demonstrator is available at http://www.lgi2p.ema.fr:8090/kid/tools/bandrec.

To evaluate the relevance of a RS based our proposal, let's compare the results obtained by our demonstrator to those recommended by Last.fm. For each music band, Last.fm proposes a set of bands and artists denoted as similar. This recommendation relies both on a large database dedicated to the music and on an analysis of their user preferences. Our demonstrator makes use of a less curated knowledge base (built from DBpedia), although it still relies on a structured representation of knowledge and also add the notion of music band popularity. This evaluation step relies on 11 queries, whose results obtained by our approach were compared to those proposed by Last.fm. Among the 40 bands proposed by Last.fm for these 11 queries, 19 were also recommended by our system. The differences between these set of recommendations mainly rely on the quality of annotations associated with the bands as well as on the importance assigned to group popularity. This result is promising since many of the recommendations proposed by our system are relevant according to the semantic characterization associated with the targeted band. This first evaluation demonstrates not only the added value of the proposed framework for defining semantic measures that serve to compare instances defined in an RDF knowledge base, but also how it can be used to design content-based RSs. Extended evaluations have to be performed to validate those results. In addition, a thorough study on the choice of projections and measures is needed to facilitate the use of the framework by a wider audience.

5 Conclusions

A new framework has been proposed for defining semantic measures between pairs of instances defined in an RDF knowledge base. Based on the notion of *RDF projection*, this framework allows for an improved characterization of instances properties and thus paves the way for the design of highly specific semantic measures compatible with a wide array of application contexts. Based on a software prototype which implements this framework, we demonstrated the suitability of our proposal for Information Retrieval, and more particularly, for content-based recommendation system design. Moreover, this framework enables domain experts to explicitly define the aspects of instances that must be taken into account to ensure the relevance of results and to characterize the semantics associated to a recommendation.

References

1. Sy, M.-F., Ranwez, S., Montmain, J., Regnault, A., Crampes, M., Ranwez, V.: User centered and ontology based information retrieval system for life sciences. BMC Bioinformatics 13(suppl. 1), S4 (2012)
2. Oldakowski, R., Bizer, C.: SemMF: A Framework for Calculating Semantic Similarity of Objects Represented as RDF Graphs. Poster at the 4th International Semantic Web Conference (2005)
3. Jeh, G., Widom, J.: SimRank. In: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, p. 538. ACM Press, New York (2002)
4. Ehrig, M., Haase, P., Hefke, M., Stojanovic, N.: Similarity for Ontologies - a Comprehensive Framework. In: Workshop Enterprise Modelling and Ontology: Ingredients for Interoperability, at PAKM (2004)
5. Kiefer, C., Bernstein, A., Stocker, M.: The Fundamentals of iSPARQL - A Virtual Triple Approach for Similarity-Based Semantic Web Tasks. In: Aberer, K., et al. (eds.) ISWC/ASWC 2007. LNCS, vol. 4825, pp. 295–309. Springer, Heidelberg (2007)
6. Albertoni, R., De Martino, M.: Semantic similarity of ontology instances tailored on the application context. In: Meersman, R., Tari, Z. (eds.) OTM 2006. LNCS, vol. 4275, pp. 1020–1038. Springer, Heidelberg (2006)
7. Euzenat, J., Shvaiko, P.: Ontology matching. Springer (2007)
8. Volz, J., Bizer, C., Gaedke, M., Kobilarov, G.: Silk – A Link Discovery Framework for the Web of Data. In: Proceedings of the 2nd Linked Data on the Web Workshop, pp. 1–6 (2009)
9. Andrejko, A., Bieliková, M.: Comparing Instances of Ontological Concepts for Personalized Recommendation in Large Information Spaces. Computing and Informatics 28, 429–452 (2013)
10. Burke, R.: Hybrid Recommender Systems: Survey and Experiments. User Modeling and User-Adapted Interaction 12, 331–370 (2002)
11. Celma, O., Serra, X.: FOAFing the music: Bridging the semantic gap in music recommendation. Web Semantics Science Services and Agents on the World Wide Web 6, 250–256 (2008)
12. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.G.: DBpedia: A nucleus for a web of open data. In: Aberer, K., et al. (eds.) ISWC/ASWC 2007. LNCS, vol. 4825, pp. 722–735. Springer, Heidelberg (2007)
13. Hoffart, J., Suchanek, F.M., Berberich, K., Weikum, G.: YAGO2: a spatially and temporally enhanced knowledge base from Wikipedia. Artificial Intelligence 194, 28–61 (2013)
14. Passant, A.: Dbrec—music recommendations using DBpedia. In: Patel-Schneider, P.F., Pan, Y., Hitzler, P., Mika, P., Zhang, L., Pan, J.Z., Horrocks, I., Glimm, B. (eds.) ISWC 2010, Part II. LNCS, vol. 6497, pp. 209–224. Springer, Heidelberg (2010)
15. Baumann, S., Schirru, R.: Using Linked Open Data for Novel Artist Recommendations. In: 13th Internal Society for Music Information Retrival Conference, Porto (2012)

Analyzing Dimension Mappings and Properties in Data Warehouse Integration

Domenico Beneventano, Marius Octavian Olaru, and Maurizio Vincini

Department of Engineering "Enzo Ferrari"
University of Modena and Reggio Emilia, Italy
firstname.lastname@unimore.it

Abstract. Data quality is one of the main issues both when building a Data Warehouse (DW) as when integrating two or more heterogeneous DWs. In the current paper, we perform extensive analysis of a mapping-based DW integration methodology and of its properties. In particular, the method is *coherent*, meanwhile under certain hypothesis it is able to preserve *soundness* and *consistency*. Furthermore, the necessary conditions for *homogeneity* will be discussed.

1 Introduction

Data Warehouse integration is the process of combining *multidimensional* information contained in two or more heterogeneous DWs by correctly identifying and integrating similar facts, measures and dimension categories. The process is required when two or more collaborating organizations need to be able to jointly take decisions based on strategic information obtained from all the participants of the collaboration. In such cases, however, *data quality* plays a crucial role, as decisions taken on incorrect information may be useless or may have a potentially negative impact on the organization. That is why any DW integration methodology must ensure basic quality properties, like coherency, soundness and consistency.

The problems when integrating two or more DWs are numerous. The information may be represented differently (e.g., a fact in one DW may be a dimension category in the other DW), with different values (e.g., a *month* may be represented by its name or by a number) or simply the same information is structured differently (e.g., different dimension categories).

In our previous work [1,2] we have proposed a mapping-based integration methodology that is able to generate mappings between dimension categories and to allow category and members import from similar dimensions. In the current paper we extend the previous work by formally reasoning about the properties of the mapping discovery and integration methodology. Interestingly, the mapping discovery step of the methodology is able to generate only mappings that are *coherent*, meanwhile the hypotheses under which *soundness* and *consistency* are preserved will be discussed.

The paper is structured as follows: Sect. 2 provides an overview of related work; Sect. 3 provides the preliminary discussion of dimension mappings and the properties that a mapping can have, meanwhile Sect. 4 provides the analytic discussion of the properties that are guaranteed and/or maintained while performing mapping discovery and dimension integration. Finally, Sect. 5 presents the conclusions of the current work.

2 Related Work

Data Warehouse integration has received little attention until recent years. Although most researchers proposed design methodology to facilitate the exchange of multidimensional information, recently a number of approaches tackled the DW integration problem in a systematic way.

For example, [3] proposes a mapping among dimensions defined as a partial function on the sets of dimension categories of the two mapped DWs. A mapping may have three properties, *coherency*, *soundness* and *consistency* that guarantee the correctness of the mapping among dimensions. The paper also introduces the *dimension algebra* as a way of manipulating DW dimensions, and two approaches to DW integration: a loosely coupled architecture and a tightly coupled architecture.

In [4], the authors propose a mapping methodology for DW elements (dimension levels and facts) by using *class similarity.* The approach presents some similarities with the integration methodology we previously developed [1,2]; however, meanwhile the authors in [4] rely on *class similarity* for the mapping process, we on the other hand use semantic similarity only as a validation step. Moreover, although the authors do not explicitly define and manage mapping properties, the methodology discard mappings that are not *coherent.*

In [5] the authors propose a semi-automatic definition of inter-attribute semantic mappings and transformation functions that can be used when performing low-level integration of different DWs.

3 Preliminaries

Although many models for representing DW dimensions have been proposed, for the purposes of the current paper the model presented in [6] will be used.

A *dimension schema* is a *directed acyclical graph* (dag) $H = (C, \nearrow)$, where C is a finite set of *categories* and $\nearrow$ is a partial order relation over the set C. The partial order relation $\nearrow$ expresses the conceptual *roll-up* relations among categories. A *bottom category* c_{bottom} is a category reachable from no other category of the schema: $\nexists c \in C$ such that $c \nearrow c_{bottom}$. A *hierarchy domain* is a dag $h = (M, <)$, where M is the set of *members* of the hierarchy and $<$ a partial order relations over the set M. The reflexive and transitive closures of $\nearrow$ and $<$ will be denoted by $\nearrow^*$ and $<^*$ respectively. For example, in Fig. 1 *month* is a category, meanwhile "*Jan 2011*", "*Feb 2011*" are members of that category. Also, *month* $\nearrow$ *season* and "Jan 2011" $<$ "spring 2011", etc.

A *dimension* d over a schema $(C, \nearrow)$ is a graph morphism $d : (M, <) \longrightarrow (C, \nearrow)$ such that $\forall x <^* y$ and $x <^* z$ such that $y \neq z$, then $d(y) \neq d(z)$.

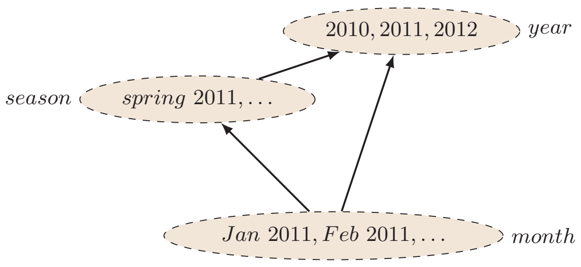


Fig. 1. A *time* dimension

Moreover, let $\mathfrak{m} : C \rightarrow \mathcal{P}(M)$ be a function that assigns each category the set of members mapped to the category by the graph morphism; in other words $\mathfrak{m}(c) = \{m \in M | d(m) = c\}$ is the set of members of c.

3.1 Dimension Mappings and Properties

A dimension mapping is a function that maps categories of one dimension to categories of another dimension [3]. Given two dimensions $d_1 : (M_1, <_1) \rightarrow (C_1, \nearrow_1)$ and $d_2 : (M_2, <_2) \rightarrow (C_2, \nearrow_2)$, a mapping is a *partial* function $\mu : C_1 \rightarrow C_2$.

The work in [3] defines properties that a mapping may have: *coherency*, *soundness* and *consistency.*

Coherency guarantees that mapped dimension categories roll-up to similar categories. Formally, μ is coherent if $\forall c_i, c_j \in C$, $c_i \nearrow_1^* c_j \Leftrightarrow \mu(c_i) \nearrow_2^* \mu(c_j)$.

Soundness is guaranteed whenever mapped categories contain the same members, meaning that $\mathfrak{m}(c) = \mathfrak{m}(\mu(c))$ for all $c \in C_1$.

Consistency states that members of mapped categories roll-up to members of categories that are also mapped by μ. In other words, $\forall m_{i_1}, m_{j_1} \in M_1$ such that $m_{i_1} <_1^* m_{j_1}$ then $\forall m_{i_2} \in \mathfrak{m}(\mu(d(m_{i_1})))$ such that $m_{i_1} = m_{i_2}$, then $\exists m_{j_2} \in \mathfrak{m}(\mu(d(m_{j_1})))$ such that $m_{i_2} <_2^* m_{j_2}$.

Furthermore, a mapping that is coherent, sound and consistent is called a *perfect mapping.*

Schema Homogeneity
In [6] a schema is defined *homogeneous* if for all categories $c_i \nearrow c_j$, if a member of c_i rolls-up to a member of c_j, than all members of c_i roll-up to a member of c_j. Otherwise, it is *heterogeneous.*

3.2 A Dimension Integration Methodology

In our previous work [1,2] we have presented a mapping and integration methodology that is able to: (a) *map* dimension categories and (b) *import* dimension members from one dimension to another.

The mapping generation step uses the graph-like structure of dimension and cardinality-related properties to map categories. Subsequently, semantic similarity is used to discard *unreliable* mappings by computing a similarity score with the *MELIS* [7] methodology.

The second step of the methodology imports categories and member from one dimension to another by using the following approach. Let d_1 and d_2, be two dimensions, and $c_i, c_j \in C_1$ such that $c_i \nearrow_1 c_j$ and $c'_i \in C_2$. Let ξ be a mapping generated by the first step of the methodology. If $\xi(c_i) = c'_i$ and $\xi(c_j) \notin C_2$, then d_2 is augmented with the category c_j and with the roll-up relations derived from the mappings. A new dimension $d_2^\sharp : (M_2^\sharp, <_2^\sharp) \rightarrow (C_2^\sharp, \nearrow_2^\sharp)$ is derived, where $C_2^\sharp = C_2 \cup \{c_j\}$; $\nearrow_2^\sharp$ is extended to include the relations between c_j and the categories of C_2. Also, $M_2^\sharp = M_2 \cup \{\mathfrak{m}(c_j)\}$ (see Fig. 2) and relation $<_2$ is extended to $<_2^\sharp$ in order to include the relations between the newly imported members and the initial members of the categories in d_2. Formally, for every $c_i, c_j \in C_1$ and $c'_i, c'_j \in C_2^\sharp$ such that $c_i \nearrow_1 c_j$ and $c'_i \nearrow_2^\sharp c'_j$, if $\xi(c_i) = c'_i$, then for all $m_i \in \mathfrak{m}(c_i)$ and $m_j \in \mathfrak{m}(c_j)$ such that $m_i <_1 m_j$, and for all $m'_i \in \mathfrak{m}(c'_i)$ such that $m_i = m'_i$, then it must be that $m_j \in M_2^\sharp$ and $m'_i <_2^\sharp m_j$ (see Fig. 2). The mapping ξ is extended to include the newly imported category. A new mapping $\xi^\sharp : C_1 \rightarrow C_2^\sharp$ is generated as follows:

$$\xi^\sharp(c) = \begin{cases} \xi(c), & \text{if } c \neq c_j \\ c_j, & \text{if } c = c_j \end{cases}$$

The members of c'_j are imported using an approach based on the *RELEVANT* [8] clustering methodology. Note that, for the sake of simplicity, the example in Fig. 2 does not highlight the advantages of using *RELEVANT*, which was chosen for its ability to combine information from more than one dimension and to discriminate between incorrect members by using clustering techniques rather than direct equality of the members. In our methodology, *RELEVANT*

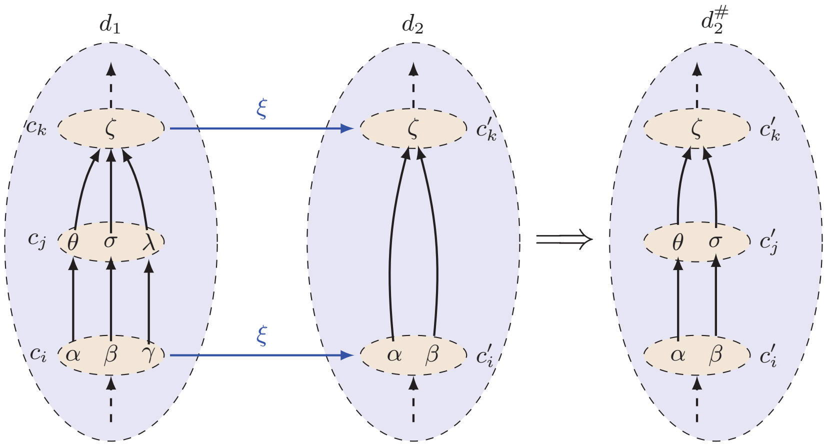


Fig. 2. The *category* and *member* importation rule

is used by enforcing that clustered members from mapped dimension categories roll-up to members of the same cluster (as computed by *RELEVANT*) of members of mapped categories.

4 Property Analysis

In this section we will analyze the properties that are guaranteed and preserved by the proposed dimension integration methodology. To this end, we will denote by ξ a mapping generated by the first step of such methodology (see previous section). Also, the proofs of all Theorems and Corollaries are reported in [9].

4.1 Coherency, Soundness and Consistency Check

The first important result is that the mapping generated by the presented methodology is always *coherent.*

Theorem 1. *The mapping ξ is coherent.*

The proof of the theorem is presented in [9] and it relies on graph-theory to demonstrate that the mappings are coherent by construction.

Although the first step of the integration methodology produces a coherent mapping, soundness and consistency are guaranteed only in certain cases. In order to verify whether soundness is verified, two steps must be performed. First, the initial mapping must be checked. Formally, for all categories $c \in C_1$, it must be that $\mathfrak{m}(c) = \mathfrak{m}(\xi(c))$. This is a simple inclusion test that will be analyzed no further. Secondly, the soundness and consistency property must be verified when performing the category and member importation.

The following theorem provides a sufficient condition for guaranteeing soundness and consistency when importing categories and members.

Theorem 2. *If ξ is sound and consistent, then $\xi^{\sharp}$ is also sound and consistent.*

Theoreme 2 provides a *sufficient*, but not *necessary* condition for soundness and consistency. In fact, there may be cases when mapping two dimensions where the initial mapping ξ is neither sound nor consistent, but the final mapping $\xi^{\sharp}$ becomes sound and/or consistent. For example, Fig. 3 provides two dimensions and a mapping ξ that is neither sound nor consistent, as $\mathfrak{m}(c_j) \neq \mathfrak{m}(c'_j)$ (the member σ belongs to $\mathfrak{m}(c_j)$ but not to $\mathfrak{m}(c'_j)$) and member β rolls-up to member α in dimension d_1, but rolls-up to no member of c'_j in dimension d_2. Note that dimension d_2 is also heterogeneous. Assuming the methodology generated the mapping ξ (see Fig. 3), step 2 of the methodology renders the mapping both sound and consistent.

The reason for which soundness and consistency are considered together is that they are closely related. In some cases (not all) soundness follows from consistency. The following corollary states a relationship between consistency and soundness of a mapping relation.

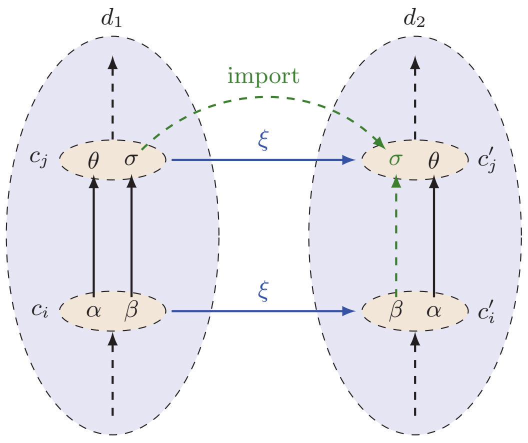


Fig. 3. No sound → sound mapping

Corollary 1. *If ξ maps only pairs of categories c_i and c_j such that $c_i \nearrow c_j$ and ξ is consistent, than ξ is also sound.*

Moreover, a mapping ξ yielding all three properties remains a *perfect* mapping after the integration methodology.

Corollary 2. *If ξ is a perfect mapping, then $\xi^\sharp$ is also a perfect mapping.*

4.2 Checking Homogeneity

Unfortunately, homogeneity is not preserved when integrating different DW dimensions. Not even the case when integrating two homogeneous dimensions can ensure that the derived dimension is homogeneous.

Interestingly, heterogeneity is also not preserved. There may be the case when a homogeneous dimension is obtained when integrating two heterogeneous dimensions. For example, in Fig. 4 when integrating members from dimension d_1 to dimension d_2 (both heterogeneous), the newly derived dimension $d_2^{\#}$ is homogeneous. In this later case, the instance of dimension d_2 is completed with information from d_1. A situation like the one described in Fig. 4 may be encountered in real life cases when analysts decide to model the same information differently, or when information is partially missing by choice or by error. For example, categories c_j and c'_j may represent the region of a city (categories c_i and c'_i), that was omitted for some cities in d_1 (member β) or for other members in d_2 (member α). The missing information is thus derived from the other dimension.

In some circumstances, homogeneity may be maintained when integrating two different homogeneous dimensions. The following theorem provides a sufficient condition to guarantee the preservation of homogeneity.

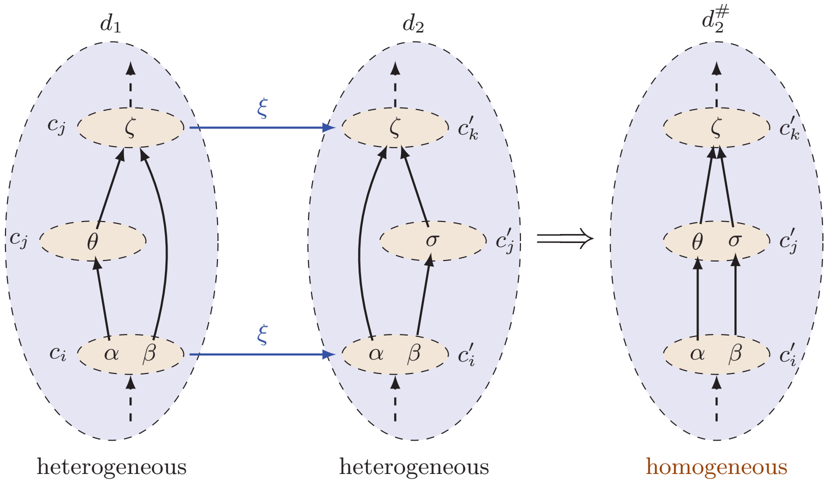


Fig. 4. Heterogeneous $\Longrightarrow$ Homogeneous

Theorem 3. *If d_1 and d_2 are homogeneous and $\mathfrak{m}(c'_i) \subseteq \mathfrak{m}(c_i)$, then $d_2^{\sharp}$ is also homogeneous.*

An interesting observation may be drawn from Theorem 3. It turns out that when integrating two homogeneous dimensions d_1 and d_2 with bottom categories c_{bottom_1} and c_{bottom_2}, if $\mathfrak{m}(c_{\text{bottom}_1}) = \mathfrak{m}(c_{\text{bottom}_2})$, then by importing categories and members from one dimension to another, the newly obtained dimensions $d_1^{\#}$ and $d_2^{\#}$ are identical, a part from the names of the categories. In other words, there will be a total mapping $\varkappa : C_1^{\#} \to C_2^{\#}$ that is *perfect*. Furthermore, $\varkappa^{-1}$ is also a *perfect* mapping. The newly derived dimensions $d_1^{\#}$ and $d_2^{\#}$ are identical to the one derived in [3] using the *tightly coupled* approach.

Corollary 3. *If $\mathfrak{m}(c'_i) \nsubseteq \mathfrak{m}(c_i)$ and $c_j \notin C_2$, then $d_2^{\#}$ is heterogeneous.*

5 Conclusions

In the current paper, a formal analysis of a mapping based DW integration methodology and its properties have been performed. The presented methodology is able to generate mappings that are *coherent*, which in turn allow correct aggregation of information from the different DWs. Moreover, under specific constraints, after performing the integration steps, the mapping may also be rendered *sound* and *consistent*.

Coherency and consistency are properties related to roll-up relations, thus a mapping satisfying both properties ensures that the integrated information can be correctly aggregated (or disaggregated). This observation is relevant as the multidimensional data is usually explored along aggregation patterns, drilling-down or rolling-up from a starting analysis point.

On the other hand, soundness states that the mapped dimension categories contain the same members, which in turn give analysts the possibility of executing meaningful drill-across queries that would otherwise be impossible if the related categories contained distinct members.

Finally, although some researchers allow the design of heterogeneous dimensions, we on the other hand consider that homogeneity allows a more clear representation of multidimensional data, both for designers and analysts as for business people that may have a simpler perception of the underlying DW model. The current paper analyzed schema heterogeneity and provided a sufficient condition for maintaining homogeneity that is a necessary condition for summarizability[10] and for materializing views as a mean of optimizing response time when executing dependent queryes[11].

References

1. Bergamaschi, S., Olaru, M.O., Sorrentino, S., Vincini, M.: Dimension matching in Peer-to-Peer Data Warehousing. In: 16th IFIP WG 8.3 International Conference on Decision Support Systems, Anávissos, Greece, pp. 149–160 (2012)
2. Guerra, F., Olaru, M.-O., Vincini, M.: Mapping and Integration of Dimensional Attributes Using Clustering Techniques. In: Huemer, C., Lops, P. (eds.) EC-Web 2012. LNBIP, vol. 123, pp. 38–49. Springer, Heidelberg (2012)
3. Torlone, R.: Two approaches to the integration of heterogeneous data warehouses. Distributed and Parallel Databases 23(1), 69–97 (2008)
4. Banek, M., Vrdoljak, B., Tjoa, A.M., Skocir, Z.: Automated Integration of Heterogeneous Data Warehouse Schemas. IJDWM 4(4), 1–21 (2008)
5. Bergamaschi, S., Guerra, F., Orsini, M., Sartori, C., Vincini, M.: A semantic approach to ETL technologies. Data & Knowledge Engineering 70(8), 717–731 (2011)
6. Hurtado, C.A., Gutierrez, C., Mendelzon, A.O.: Capturing summarizability with integrity constraints in OLAP. ACM Transactions on Database Systems 30(3), 854–886 (2005)
7. Bergamaschi, S., Bouquet, P., Giacomuzzi, D., Guerra, F., Po, L., Vincini, M.: An Incremental Method for the Lexical Annotation of Domain Ontologies. Int. J. Semantic Web Inf. Syst. 3(3), 57–80 (2007)
8. Bergamaschi, S., Sartori, C., Guerra, F., Orsini, M.: Extracting Relevant Attribute Values for Improved Search. IEEE Internet Computing 11(5), 26–35 (2007)
9. Beneventano, D., Olaru, M.O., Vincini, M.: Dimension Mapping and Properties (Technical Report) (2013), `http://www.dbgroup.unimo.it/files/ODBASE.pdf`
10. Lenz, H.J., Shoshani, A.: Summarizability in OLAP and Statistical Data Bases. In: SSDBM, pp. 132–143 (1997)
11. Harinarayan, V., Rajaraman, A., Ullman, J.D.: Implementing data cubes efficiently. In: Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data - SIGMOD 1996, pp. 205–216. ACM Press, New York (1996)

Semantic Enrichment of OLAP Cubes: Multi-dimensional Ontologies and Their Representation in SQL and OWL

Bernd Neumayr, Christoph Schütz, and Michael Schrefl

Johannes Kepler University Linz, Austria
{neumayr,schrefl,schuetz}@dke.uni-linz.ac.at

Abstract. A multi-dimensional ontology (MDO) enriches an OLAP cube with concepts that represent business terms in the context of data analysis. The formal representation of the meaning of business terms facilitates the unambiguous interpretation of query results as well as the sharing of knowledge among business analysts. In contrast to traditional ontologies, an MDO captures the multi-dimensional, hierarchical world view of business analysts. In this paper, we introduce a translation of MDO concepts to SQL in order to allow for the querying of a closed-world OLAP cube. We introduce a representation in OWL in order to determine subsumption hierarchies of MDO concepts using off-the-shelf reasoners.

Keywords: Business Intelligence, OLAP, Data Warehouse, Knowledge Representation and Reasoning.

1 Introduction

An OLAP cube is typically deployed on top of the asserted data in a data warehouse and provides a coherent, multi-dimensional and multi-granular view of (a fragment of) the data in the data warehouse. Analysts query OLAP cubes in order to gain insights into their businesses. When many business analysts work together understandability, reuse, and maintainability of multi-dimensional queries become key challenges. These problems can be tackled by defining a shared vocabulary of business terms.

The web ontology language OWL is a first candidate for a formal definition language for shared vocabularies or ontologies. OWL comes with automated reasoning support for subsumption and satisfiability checking over ontologies. Such automated reasoning support has proven vital for maintaining the consistency of large, collaboratively-engineered ontologies. Using traditional ontology languages such as OWL it is, however, difficult to represent the multi-granular and multi-dimensional conceptualizations of business analysts where abstract points at various granularities in a multi-dimensional space represent and quantify parts of the domain of interest.

The introduction of the multi-dimensional ontology language overcomes the shortcomings of traditional ontology languages in the context of data warehousing. A multi-dimensional ontology (MDO) captures the multi-granular and multi-dimensional world-view of business analysts; it unambiguously defines business terms and explains the calculation of measures. Thus, an MDO facilitates the selection of facts as well as the interpretation of query results.

For closed-world querying, an OLAP cube is enriched with SQL views representing the concepts of the ontology. First, the definition of relational views over the OLAP cube allows for querying using SQL. This approach is akin to cube and dimensional views provided by Oracle OLAP [1]. Second, every MDO concept is translated into a relational view over the cube views and the dimensional views. This translation of MDO concepts into relational views is platform-independent. For increased performance, platform-specific logical and physical optimizations may be applied, which is out of the scope of this paper.

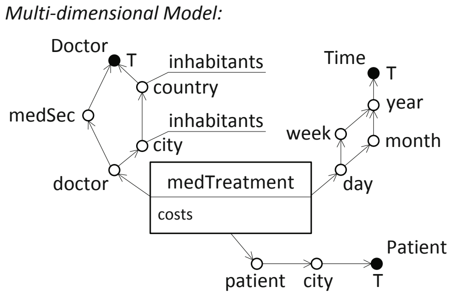
A representation of MDO concepts in OWL allows for automated subsumption and satisfiability checking, using off-the-shelf reasoners. Disjointness of transitive roll-up relationships is a fundamental property of roll-up hierarchies and difficult to represent in OWL 2 DL since functionality cannot be defined upon transitive object properties. In this paper we present a solution for this problem.

The MDO has been developed as part of the *Semantic Cockpit* (semCockpit) project [13,12], a joint effort between academia, BI industry, and public health insurers. In addition to introducing the representation of MDO concepts in SQL and OWL, in this paper, we extend the MDO language with disjunction and complement of concepts. Due to these extensions, the Datalog-based approach [12] is insufficient.

The remainder of the paper is organized as follows. In Sect. 2 we state the problem and motivate our approach. In Sect. 3 we define the abstract data model for OLAP cubes. In Sect. 4 we present the definition of MDO concepts over OLAP Cubes and introduce a representation in OWL as well as transformations into relational views for querying. In Sect. 5 we present a proof-of-concept prototype and discuss implementation issues. In Sect. 6 we review related work.

2 Motivation

An OLAP cube is a multi-dimensional model representing real-world facts at various levels of aggregation. Facts are quantified by measures and identified by entities from multiple dimensions. The dimensions have hierarchically organized levels which allow for the analysis of measures at different granularities. Figure 1 illustrates a multi-dimensional model for the analysis of medical treatments. In this model, the recorded measure are the costs which are available by the day for a particular acting doctor and insurant. Along the dimensions Time, Doctor, and Patient a business analyst may roll up the data in order to obtain the costs for coarser time periods and groups of acting doctors and insurants. As non-dimensional attribute, the number of inhabitants provides additional information about the city and country of a doctor.



Relational Views:
medTreatment(costs, <u>actDoc, time, insurant</u>)
costs(costs, <u>actDoc, time, insurant</u>)
costsPerInsurant(costsPerInsurant, <u>actDoc, time</u>)
Doctor(<u>Doctor</u>, Doctor_lvl)
Doctor_rollup(<u>Doctor, Doctor_sup</u>)
Time(<u>Time</u>, Time_lvl)
Time_rollup(<u>Time, Time_sup</u>)
Patient(<u>Patient</u>, Patient_lvl)
Patient_rollup(<u>Patient, Patient_sup</u>)
CityE(<u>CityE</u>, inhabitants)
CountryE(<u>CountryE</u>, inhabitants)

Fig. 1. A multi-dimensional model and its relational views for querying

Many data warehouse systems manage OLAP cubes in multi-dimensional data structures that are optimized for query performance. Some of these systems, for example, Oracle OLAP [1], provide relational views as a query interface for the business analyst. For example, the relational view representation of the multi-dimensional model in Fig. 1 comprises a fact view (medTreatment) which stores in column costs the costs at the most granular level. Measure view costs contains the costs at every level of granularity. The other columns (actDoc, time, insurant) of views medTreatment and costs reference dimension views (Doctor, Time, Patient) which associate entities of a dimension with a particular level. Another measure view (costsPerInsurant) abstracts from the Patient dimension and defines the costs per insurant as a derived measure for dimensions Doctor and Time at every level of granularity. Roll-up views (Doctor_rollup, Time_rollup, Patient_rollup) determine for each entity its superordinate entities, that is, the transitive and reflexive closure of the immediate parent entities. Entity views (CityE, CountryE) represent the non-dimensional attributes (inhabitants).

Working with the relational views over an OLAP cube can be cumbersome. Business analysts encode business terms directly into SQL queries. For example, the query in Fig. 2 retrieves the costs per insurant in the year 2010 by city of acting doctor where the acting doctor is in a big city of a small country. The analyst defines in the where-clause of the SQL query a big city as a city with

```
SELECT * FROM costsPerInsurant
WHERE actDoc IN (
  SELECT Doctor FROM Doctor_rollup
  WHERE Doctor_sup IN (
    SELECT CityE FROM CityE WHERE inhabitants>1000000 ) )
AND actDoc IN (
  SELECT Doctor FROM Doctor_rollup
  WHERE Doctor_sup IN (
    SELECT e_country FROM e_country WHERE inhabitants<20000000 ) )
AND actDoc IN (
  SELECT Doctor FROM Doctor WHERE Doctor_lvl = 'city' )
AND time = '2010'
```

Fig. 2. An SQL query over the views on the multi-dimensional model in Fig. 1

more than a million inhabitants and a small country as a country with less than twenty million inhabitants. For each query, the analyst must define the business terms from scratch.

The explicit definition of business terms in concept views allows for the reuse of business term definitions in several queries and facilitates the formulation of these queries. For example, the query in Fig. 3 references a concept view (actDocInBigCityAndSmallCountry) that already defines an acting doctor in a big city of a small country. In the reformulated query, a natural join with the concept view replaces the subqueries from the original query (Fig. 2). Consequently, when formulating the query, the analyst is freed of defining these business terms.

```
SELECT * FROM costsPerInsurant
  NATURAL JOIN actDocInBigCityAndSmallCountry
WHERE actDoc IN (
  SELECT Doctor FROM Doctor WHERE Doctor_lvl = 'city' )
AND time = '2010'
```

Fig. 3. A reformulation with concept views of the SQL query from Fig. 2

Besides their use in the formulation of queries, concept views also facilitate the definition of derived measures. A derived measure, as opposed to a base measure with asserted measure values, applies some measurement instruction to the available measure values. For example, a business analyst may derive the costs per rural insurant (Fig. 4) from the costs using the concept view that defines rural patients (ruralPatient).

```
CREATE VIEW costsPerRuralInsurant AS
SELECT AVG(mt.costs) AS costsPerRuralInsurant, d.sup AS Doctor, t.sup AS Time
FROM medTreatment mt NATURAL JOIN Doctor_rollup d
  NATURAL JOIN Time_rollup t NATURAL JOIN ruralPatient rp
GROUP BY d.sup, t.sup
```

Fig. 4. Definition of a derived measure using concept views

Rather than defining concept views in SQL, we propose a multi-dimensional ontology (MDO) language. The MDO language allows for an organization-wide coherent definition of business terms. In order to share definitions of business terms among business analysts there must be some sort of classification which facilitates the retrieval of concepts. Automated reasoners may organize the concepts in subsumption hierarchies. As a prerequisite for the use of existing, off-the-shelf automated reasoners, the MDO concepts must be formalized in a standard ontology language, for example, the web ontology language OWL. Thus, in this paper, we also provide a translation of MDO concepts into OWL which allows for their automated classification.

3 OLAP Cubes

In this section we describe the abstract data model of OLAP cubes together with a concrete notation and representations in OWL and SQL. The notion of *cube* is used differently: (a) for a set of facts at some granularity, (b) for a set of base and derived facts at different granularities (data cube), and (c) a set of related data cubes. We use the term OLAP cube in the third sense.

3.1 Abstract Data Model

The MDO language allows for the specification of schema and instance of OLAP cubes. Table 1 defines, in the left column, the concrete MDO syntax for the specification of an OLAP cube schema. Throughout the remainder of this paper, variables are *emphasized* and terminal symbols are written in sans-serif typeface. The schema of an OLAP cube consists of entity classes, dimensions, dimension roles, dimension spaces, and measures. An entity class (Row 1) defines a set of attributes $attr_1, \ldots, attr_n$ with data types $dt_1, \ldots, dt_n$. Attributes are also referred to as descriptors or non-dimensional attributes. A dimension (Row 2) consists of a set of levels organized in a level roll-up hierarchy with a single top level and a single bottom level. A level refers to an entity class with an entity class being referred to by at most one level per dimension. A level-range restricted dimension (Row 3) is defined over a dimension d and restricts the set of levels of d to those levels that are between or equal to a given from-level l' and a given to-level l''.

A dimension space is the multi-dimensional space that corresponds to the cartesian product of a list of dimensions. Within a dimension space, a dimension may play several dimension roles (Row 4). A dimension space (Row 5) is defined over a set of dimension roles $dr_1, \ldots, dr_n$ for level-range restricted dimensions $lrd_1, \ldots, lrd_n$. The level is the role that an entity class plays within a dimension. The overall set of dimension roles of an OLAP cube constitutes a universal dimension space. A measure (Row 6) is defined for a dimension space. The measure values depend functionally on the dimension roles in the dimension space.

Table 2 illustrates, in the left column, the definition of example fragments of an OLAP cube schema. Entity class CityE (Row 1) has an attribute inhabitants of data type integer. This entity class CityE is associated with the city level of the Doctor dimension (Row 2). In the hierarchy of the Doctor dimension, the city level is the superordinate level of doctor. The level-range restricted dimension Doctor_CityCountry (Row 3), the Doctor dimension to the levels from city up to country. The acting doctor (actDoc, Row 4) is a role of the doctor dimension. The actDoc role is used in dimension space ds1. In this dimension space, the actDoc role is defined for the restricted level range from city to country. The dimension space ds1 has a measure costs (Row 6).

An instance of an OLAP cube (Table 3) comprises entities and nodes. These individuals are described by their connections and properties. An entity (Row 1) is member of an entity class *ecl* and assigns data values to the attributes of the

Table 1. Representation of an OLAP cube schema in OWL and as relational views

	MDO	OWL	Relational
(1)	CREATE ENTITY CLASS ecl ($attr_1$ dt_1, ..., $attr_n$ dt_n);	$ecl \sqsubseteq$ Entity $\exists attr_1.\top \sqsubseteq ecl \ldots$ $\exists attr_n.\top \sqsubseteq ecl$ $\top \sqsubset \forall attr_1.dt \ldots$ $\top \sqsubset \forall attr_n.dt$ $\top \sqsubseteq \leqslant 1 attr_1 \ldots$ $\top \sqsubseteq \leqslant 1 attr_n$	$ecl(\underline{ecl}, attr_1, \ldots, attr_n)$
(2)	CREATE DIMENSION d WITH LEVELS l_1 ecl_1, ..., l_n ecl_n AND HIERARCHY l'_1 UNDER l''_1, ..., l'_m UNDER l''_m;	$d \sqsubseteq$ Node $\exists$atLevel.$\{l_1\} \equiv$ $d \sqcap \exists$roleOf.$ecl_1 \ldots$ $\exists$atLevel.$\{l_n\} \equiv$ $d \sqcap \exists$roleOf.ecl_n directlyRollsUpTo(l'_1, l''_1) ... directlyRollsUpTo(l'_m, l''_m)	d_lvl($\underline{d\text{\_lvl}}$,entityclass) d_lvlparent($\underline{d\text{\_lvl}, d\text{\_lvl\_sup}}$) d_lvlrollup($\underline{d\text{\_lvl}, d\text{\_lvl\_sup}}$) $d(\underline{d}, d\text{\_lvl})$ d_parent($\underline{d, d\text{\_sup}}$) d_rollup($\underline{d, d\text{\_sup}}$) insert into d_lvl values (l_1,ecl_1); ... insert into d_lvl values (l_n,ecl_n); insert into d_lvlparent values (l'_1,l''_1); ... insert into d_lvlparent values (l'_m,l''_m);
(3)	CREATE LEVELRAN-GERESTRICTED DIMENSION lrd AS $d[l'..l'']$;	$lrd \equiv d \sqcap \exists$atLevel.($\exists$rollsUpTo$^-$.$\{l'\} \sqcap$ $\exists$rollsUpTo.$\{l''\}$)	create view lrd as select * from d where d_lvl in ((select d_lvl_sup as d_lvl from d_lvlrollup where d_lvl = 'l'') intersect (select d_lvl from d_lvlrollup where d_lvl_sup = 'l'''));
(4)	CREATE DIMENSION ROLE dr OF d;	$\top \sqsubseteq \leqslant 1 dr$ $\exists dr.\top \sqsubseteq$ Point $\top \sqsubseteq \forall dr.d$	... dr varchar not null references d ...
(5)	CREATE DIMENSION SPACE ds AS $(dr_1 lrd_1, \ldots, dr_n lrd_n)$;	$ds \equiv$ Point$\sqcap \exists dr_1.lrd_1 \sqcap$ $\cdots \sqcap \exists dr_n.lrd_n$	create view ds as select * from (select d_1 as dr_1 from lrd_1) natural join ... natural join (select d_n as dr_n from lrd_n); — d_i is the dimension referred to by dr_i
(6)	CREATE MEASURE msr FOR ds;	$\exists msr.\top \sqsubseteq ds$	$msr(\underline{dr_1, \ldots, dr_n}, msr)$ — $dr_1, \ldots, dr_n$ are the dimension roles of ds

entity class *ecl*. A node (Row 2) is member of a dimension and defined at a particular level of the dimension. The nodes of a dimension are organized in a node roll-up hierarchy (Row 3) which corresponds to the level roll-up hierarchy. A node refers to a member of the entity class that the node's level refers to. The node is the role that the entity plays within the dimension. An entity may be referred to by at most one node per dimension.

In addition to entities and nodes, an OLAP cube comprises points. A point is member of a dimension space ds. The extension of a dimension space ds, that is, the set of member points, corresponds to the cartesian product of the dimension extensions of the dimension roles $dr_1, \ldots, dr_n$ of dimension space ds. A point is described by measure values. A point is identified by its coordinates $nd_1, \ldots, nd_n$, one for each of the dimension roles $dr_1, \ldots, dr_n$ of dimension space ds. Each

Table 2. Example fragments of an OLAP cube schema

	MDO	OWL	Relational
(1)	CREATE ENTITY CLASS CityE (inhabitants integer)	$\text{CityE} \sqsubseteq \text{Entity}$ $\exists\text{CityE\_inhabitants}.\top \sqsubseteq \text{CityE}$ $\top \sqsubseteq\ \leqslant 1\text{CityE\_inhabitants}$	CityE($\underline{\text{CityE}}$, inhabitants)

(2)	CREATE DIMENSION Doctor WITH LEVELS doctor DoctorE, city CityE AND HIERARCHY doctor UNDER city;	$\text{Doctor} \sqsubseteq \text{Node}$
		$\exists\text{atLevel}.\{\text{doctor}\} \equiv \text{Doctor} \sqcap \exists\text{roleOf}.\text{DoctorE}$
		$\exists\text{atLevel}.\{\text{city}\} \equiv \text{Doctor} \sqcap \exists\text{roleOf}.\text{DoctorE}$
		$\top \sqsubseteq\ \leqslant 1\text{rollsUpTo\_doctor}$
		$\top \sqsubseteq\ \leqslant 1\text{rollsUpTo\_city}$
		directlyRollsUpTo(doctor,city)
			Doctor_lvl($\underline{\text{Doctor\_lvl}}$,entityclass)
			Doctor_lvlparent($\underline{\text{Doctor\_lvl, Doctor\_lvl\_sup}}$)
			Doctor_lvlrollup($\underline{\text{Doctor\_lvl, Doctor\_lvl\_sup}}$)
			Doctor($\underline{\text{Doctor}}$, Doctor_lvl)
			Doctor_parent($\underline{\text{Doctor, Doctor\_sup}}$)
			Doctor_rollup($\underline{\text{Doctor, Doctor\_sup}}$)
			insert into "Doctor_lvl" values ('doctor','DoctorE');
			insert into "Doctor_lvl" values ('city','CityE');
			insert into "Doctor_lvlparent" values ('doctor','city');

(3)	CREATE LEVELRANGERE-STRICTED DIMENSION Doctor_CityCountry AS Doctor[city..country];	$\text{Doctor\_CityCountry} \equiv \text{Doctor} \sqcap \exists\text{atLevel}.(\exists\text{rollsUpTo}^{-}.\{\text{city}\} \sqcap \exists\text{rollsUpTo}.\{\text{country}\})$	create view "Doctor_CityCountry" as select * from "Doctor" where "Doctor_lvl" in ((select "Doctor_lvl_sup" as "Doctor_lvl" from "Doctor_lvlrollup" where "Doctor_lvl" = 'city') intersect (select "Doctor_lvl" from "Doctor_lvlrollup" where "Doctor_lvl_sup = 'country'));
(4)	CREATE DIMENSION ROLE actDoc OF Doctor;	$\top \sqsubseteq\ \leqslant 1\text{actDoc}$ $\exists\text{actDoc}.\top \sqsubseteq \text{Point}$ $\top \sqsubseteq \forall\text{actDoc}.Doctor$	... "actDoc" varchar not null references "Doctor" ...
(5)	CREATE DIMENSION SPACE ds1 AS (actDoc Doctor_CityCountry, time Time_MonthYear);	$\text{ds1} \equiv \text{Point} \sqcap \exists\text{actDoc}.\text{Doctor\_CityCountry} \sqcap \exists\text{time}.\text{Time\_MonthYear}$	create view "ds1" as select * from (select "Doctor" as "actDoc" from "Doctor_CityCountry") natural join (select "Time" as "time" from "Time_MonthYear");
(6)	CREATE MEASURE costs FOR ds1;	$\exists\text{costs}.\top \sqsubseteq \text{ds1}$	costs($\underline{\text{actDoc,time}}$,costs)

coordinate refers to one of nodes $nd_1, \ldots, nd_n$ of the dimensions $d_1, \ldots, d_n$ that correspond to the dimension roles $dr_1, \ldots, dr_n$. A link between point, measure, and value is also referred to as fact. A fact assigns a value val to a measure msr of a point p.

3.2 OWL Representation and Relational Views

For the purpose of automated subsumption checking over MDO concepts using off-the-shelf reasoners, we introduce a representation of MDO in OWL 2 DL[1]. This OWL representation does not represent the full semantics of OLAP cubes. For the purpose of management of shared business terms, an incomplete representation of OLAP cubes is sufficient.

[1] `http://www.w3.org/TR/owl2-overview/`

Table 3. Representation of an OLAP cube instance in OWL and as relational views

	MDO (open world)	**OWL** (open world)	**Relational** (closed world)
(1)	CREATE ENTITY e OF ecl ($attr_1 = val_1, \ldots,$ $attr_n = val_n$);	$ecl(e)$ $attr_1(e, val_1)$ ... $attr_n(e, val_n)$	insert into ecl (ecl, $attr_1$, ..., $attr_n$) values (e, val_1, ..., val_n);
(2)	CREATE NODE nd OF d AT l AS ROLEOF e;	$d(nd)$ atLevel(nd, l) roleOf(nd, e)	insert into d(d,d_lvl) values (e,l);
(3)	CREATE DIRECT ROLLUP FROM nd TO nd';	directlyRollsUpTo(nd, nd')	insert into d_parent (d, d_sup) values (nd,nd'); — d is the dimension of nd

The OLAP cube representation in OWL builds on a set of generic OWL classes and properties as described in Fig. 5. Object property directlyRollsUpTo represents roll-up hierarchies of nodes and levels alike. No distinction is required between the different kinds of hierarchies. Object property rollsUpTo represents the transitive and reflexive closure of directlyRollsUpTo. Since OWL does not provide a means for defining an object property as transitive-reflexive closure of another property, an incomplete work-around is provided as follows. The rollsUpTo property is characterized as transitive and directlyRollsUpTo defined as a sub-property; each node and level is specified to roll-up to itself.

Table 1, in the middle column, shows the OWL equivalents, in Description Logics notation, for the MDO OLAP cube schema definitions. Table 3, in the middle column, shows the OWL equivalents for the MDO OLAP cube instance definitions. Typically, only a small subset of the entities and nodes of an OLAP

Class: **Entity**	(1)
Class: **Node** SubClassOf: directlyRollsUpTo Self and roleOf some Entity and atLevel some Level and rollsUpTo only Node	(2)
Class: **Level** SubClassOf: directlyRollsUpTo Self and rollsUpTo only Level	(3)
Class: **Point**	(4)
DisjointClasses: Entity, Node, Level, Point	(5)
ObjectProperty: **directlyRollsUpTo** SubPropertyOf: rollsUpTo	(6)
ObjectProperty: **rollsUpTo** Characteristics: Transitive	(7)
ObjectProperty: **roleOf** Characteristics: Functional Domain: Node Range: Entity	(8)
ObjectProperty: **atLevel** Characteristics: Functional Domain: Node Range: Level	(9)
for the set of all entity classes $\{ecl_1, \ldots, ecl_n\}$: DisjointClasses: ecl_1, ..., ecl_n	(10)
for the set of all dimensions $\{d_1, \ldots, d_n\}$: DisjointClasses: d_1, ..., d_n	(11)
for the set of all entities $\{e_1, \ldots, e_n\}$: DifferentIndividuals: e_1, ..., e_n	(12)

Fig. 5. OWL classes and properties for the representation of OLAP cubes (in Manchester syntax)

cube are represented as individuals in the MDO. An OLAP cube further consists of points which are, however, not explicitly represented at the individual level of an MDO.

We do not make any assumptions about the physical storage of OLAP cubes or the algorithms for calculating and materializing derived measures. We regard OLAP cubes as data structures at the logical level which allow to directly query derived measure values and to abstract from the calculation of measure values. We employ SQL as language for querying OLAP cubes via relational views.

Table 1, in the right column, shows the relational views for the OLAP cube schema definitions. The primary keys of a view are underlined. The primary key values of the entity views, also referred to as entity IDs, are assumed to be globally unique. Some of the relational views are derived from other views. The lvlrollup views (Row 2) are the transitive and reflexive closure of the lvlparent views, the rollup views are the transitive and reflexive closure of the parent views. The definition of these views is omitted due to space considerations. Table 3, in the right column, shows the relational views for the OLAP cube instance definitions.

4 Defining MDO Concepts over OLAP Cubes

In this section we introduce a language for describing business terms as used in data analysis in the form of concepts in a multi-dimensional ontology. We specify the syntax of entity concept, dimensional concept, and multi-dimensional concept descriptions together with their representation in SQL and in OWL.

There are two main tasks concerning MDO concepts: First, MDO concepts should be directly usable in querying the OLAP cube. Second, MDO concepts should be organized in subsumption hierarchies to simplify their collaborative management and use.

For querying the OLAP cube we make the closed world assumption and consider the OLAP cube as the single interpretation (in a model-theoretic sense) of the MDO and assume that the OLAP cube is consistent with the MDO (that is, a model of the MDO). The translation of MDO concepts to SQL *concept views* does not only allow to query the OLAP cube but also defines their interpretation. That is, the semantics of the MDO concept description language is informally specified in terms of SQL. To facilitate translation to SQL views we only allow acyclic concept definitions.

For deriving subsumption hierarchies we make the open world assumption, because subsumption hierarchies should not change when the OLAP cube instance changes. To automate subsumption checking, MDO concepts are translated to OWL 2 DL and subsumption checking is delegated to OWL reasoners.

The signature of a concept is given by its name and indicates the sort of individuals over which it is defined. The extension or interpretation of a concept may be specified in one the following ways: The extension of a *primitive concept* is asserted or derived outside the MDO and is treated as black box for reasoning. The extension of a *defined concept* is derived according to a concept expression which are the basis for automated subsumption checking. The extension of an *SQL-defined concept* is specified by an SQL query over the enriched OLAP cube. For subsumption checking, SQL-defined concepts are treated as primitive concepts with their associated SQL query being ignored.

4.1 Entity Concepts

An entity concept ec is defined over an entity class ecl, its domain, and is interpreted by a subset of the members of this entity class. An entity concept is either primitive (see Table 4 Row 1), SQL-defined (Row 2), or defined (3). Primitive concepts are defined or asserted outside the MDO. SQL-defined concepts are defined within the MDO by an SQL query which is ignored for reasoning and thus not part of the transformation to OWL. Defined concepts are defined by an entity concept expression in one of the following forms: (3a) a singleton concept consisting of a single entity, (3b) an attribute-value restriction, for example, concept bigCity in Table 5, (3c) the disjunction of entity concepts, for example, concept bigOrSmallCity, (3d) the conjunction of entity concepts, (3e) or the complement of an entity concept. Attributes, entities, and entity concepts referred to in an entity concept definition all belong to the entity class which serves as domain of the defined concept.

Table 4. MDO entity concepts and their representation in OWL and SQL

	MDO	in **OWL**	in **SQL**
	CREATE ENTITY CONCEPT ec FOR ecl	$ec \sqsubseteq ecl$	$ec(\underline{ecl})$
(1) AS PRIMITIVE;			
(2) AS SQL: $sqlquery$;			create view ec AS $sqlquery$;
(3) AS		$ec \equiv$	create view ec as
(3a)	e;	$\{e\}$	select ecl from ecl where ecl = 'e';
(3b)	$attr\ \theta\ value$; — θ is some comparison operator	$\exists attr.dt[\theta\ value]$ — dt is the datatype of $attr$	select ecl from ecl where $attr\ \theta\ value$;
(3c)	UNION OF (ec_1, ..., ec_n);	$ec_1 \sqcup \cdots \sqcup ec_n$	(select ecl from ec_1) union ... union (select ecl from ec_n);
(3d)	INTERSECTION OF (ec_1, ..., ec_n);	$ec_1 \sqcap \cdots \sqcap ec_n$	(select ecl from ec_1) intersect ... intersect (select ecl from ec_n);
(3e)	NOT ec';	$ecl \sqcap \neg ec'$	(select ecl from ecl) except (select ecl from ec');

Entity concepts can easily be represented in traditional ontology languages with the translation of MDO entity concepts to OWL being straightforward (see middle column of Table 4). Representing entity concepts as SQL views is also straightforward (see third column of Table 4). Every concept is represented by a

Table 5. Examples of MDO entity concepts

MDO	in OWL	in SQL
CREATE ENTITY CONCEPT bigCity FOR CityE AS inhabitants > 100000;	bigCity $\sqsubseteq$ CityE bigCity $\equiv$ $\exists$CityE_inhabitants.integer[> 100000]	create view "bigCity" as select "CityE" from "CityE" where "inhabitants" > 100000
CREATE ENTITY CONCEPT bigOrSmallCity FOR CityE AS UNION OF (bigCity, smallCity);	bigOrSmallCity $\sqsubseteq$ CityE bigOrSmallCity $\equiv$ bigCity $\sqcup$ smallCity	create view "bigOrSmallCity" as (select "CityE" from "bigCity") union (select "CityE" from "smallCity");

concept view with the concept name as view name and a single attribute named after the entity class.

4.2 Dimensional Concepts

A dimensional concept dc is defined over a level-range-restricted dimension lrd, its domain, and is interpreted by a subset of the nodes in this domain. Interpretations of dimensional concepts have the *hierarchy property*: if a node is in the interpretation of a dimensional concept, then all its sub- and superordinate nodes within the domain of the concept are also in the interpretation of the concept. First experience shows that enforcing the hierarchy property simplifies working with and understanding dimensional concepts.

Table 6. MDO dimensional concepts and their representation in OWL and SQL

	MDO	OWL	Relational/SQL
	CREATE DIMENSIONAL CONCEPT dc FOR lrd	$dc \sqsubseteq lrd$	$dc(\underline{d})$ — d is the dimension of lrd
(1)	AS PRIMITIVE;		
(2)	AS SQL: $sqlquery$;		create view dc as $sqlquery$;
(3)	AS	$dc \equiv$	create view dc as
(3a)	nd;	$\{nd\}$	select d from d where d = 'nd';
(3b)	$d : ec$;	$d \sqcap \exists$roleOf.ec	select d from d where d in (select * from ec);
(3c)	dc'*;	$\exists$rollsUpTo_$l.dc'$ — l is a level of dc'	select d from d_rollup where d_sup in (select d from dc');
(3d)	$dc'[lrd]$	$dc' \sqcap lrd$	select d from dc' natural join lrd
(3e)	UNION OF (dc_1, ..., dc_n);	$dc_1 \sqcup \cdots \sqcup dc_n$	(select d from dc_1) union ... union (select d from dc_n);
(3f)	INTERSECTION OF (dc_1, ..., dc_n);	$dc_1 \sqcap \cdots \sqcap dc_n$	select d from dc_1 natural join ... natural join dc_n;
(3g)	NOT dc';	$lrd \sqcap \neg dc'$	(select d from lrd) except (select d from dc')

A dimensional concept is described in one of the following ways: (1) as primitive, (2) by an SQL-view over the OLAP cube, or (3) by a concept expression in one of the following forms, (3a) a single node such that only this node is in the interpretation of the concept, (3b) by a reference to an entity concept such that each node that refers to an entity satisfying the entity concept is in the interpretation of the defined concept, for example, concept doc_bigCity in Table 8, (3c) by hierarchy expansion of some concept such that each node of the dimension that is in the interpretation of the concept or some direct or indirect successor node thereof is in the interpretation, concept DocInBigCity, (3d) by level range restriction of some concept, (3e) by disjunction of concepts defined for the same level-range or (3f) by conjunction of concepts, for example, concept DocInBgCtySmCntry, (3g) the complement of a concept with regard to its domain (with the open world interpretation of complement).

Table 7. Representing disjointness of sub-dimensions

for each level l:	for example, for level country:
$\text{rollsUpTo\_}l \sqsubseteq \text{rollsUpTo}$	$\text{rollsUpTo\_country} \sqsubseteq \text{rollsUpTo}$
$\exists\text{atLevel}.\exists\text{rollsUpTo}.\{l\} \equiv \exists\text{rollsUpTo\_}l.\top$	$\exists\text{atLevel}.\exists\text{rollsUpTo}.\{\text{country}\} \equiv \exists\text{rollsUpTo\_country}.\top$
$\top \sqsubseteq \forall\text{rollsUpTo\_}l.\exists atLevel.\{l\}$	$\top \sqsubseteq \forall\text{rollsUpTo\_country}.\exists\text{atLevel}.\{\text{country}\}$
$\top \sqsubseteq \leqslant 1\text{rollsUpTo\_}l$	$\top \sqsubseteq \leqslant 1\text{rollsUpTo\_country}$
for each node nd at level l:	for example, for node austria at level country:
$\exists\text{rollsUpTo}.\{nd\} \equiv \exists\text{rollsUpTo\_}l.\{nd\}$	$\exists\text{rollsUpTo}.\{\text{austria}\} \equiv \exists\text{rollsUpTo\_country}.\{\text{austria}\}$

The domain of a defined concept may be derived from its concept expression. The domain of a concept conjunction is the intersection of domains of the constituent concepts. The domain of the complement of a concept is that of the concept. Concepts that are constructed by a single node or by reference to an entity concept are flat, that means that their domain is restricted to a single level namely those of the node(s). The domain of a concept defined by hierarchical expansion is that of the concept but expanded to the bottom level of the dimension.

The representation of dimensional concepts in OWL (see the middle column in Table 6) comes with one key challenge. OWL 2 DL does not allow to express that a transitive property (such as rollsUpTo) maps to exactly one object in a given range (for example, to one node of one level). But, the essential characteristics of roll-up hierarchies of data warehouse dimensions are that the rollsUpTo-relationship between nodes is transitive, and each node of a level rolls up to exactly one node of each higher level (to which the former level rolls up). Without these semantics of roll-up hierarchies in data warehousing being captured, the subsumption hierarchy determined by an OWL reasoner will be sound, but incomplete. More specific, it will not recognize that if two concepts dc and dc' are disjoint, their hierarchical expansions, dc^* and dc'^* will be disjoint, too.

To cope with this limitation of OWL, we include redundant, derived information in the OWL representation of levels and nodes. For each level l we introduce a functional roll-up property and assert for every named node nd at level l that all its descendant nodes roll up to nd via this functional roll-up property (see Table 7). Using such derived functional roll-up properties for defining hierarchical expansion of dimensional concepts (see Table 6 Row *3c* and the second row in Table 8) allows the reasoner to derive disjointness of concepts which are defined by hierarchical expansion.

Table 8. Examples of MDO dimensional concepts

MDO	in **OWL**	in **SQL**
CREATE DIMENSIONAL CONCEPT doc_bigCity FOR Doctor_city AS Doctor:bigCity;	doc_bigCity $\sqsubseteq$ Doctor_city doc_bigCity $\equiv$ Doctor $\sqcap$ $\exists$roleOf.bigCity	create view "doc_bigCity" as select "Doctor" from "Doctor" where "Doctor" in (select * from "bigCity")
CREATE DIMENSIONAL CONCEPT DocInBigCity FOR Doctor_doctorCity AS doc_bigCity*;	DocInBigCity $\sqsubseteq$ Doctor_doctorCity DocInBigCity $\equiv$ $\exists$rollsUpTo_city.doc_bigCity	create view "DocInBigCity" as select "Doctor" from "Doctor_rollup" where "Doctor_sup" in (select * from "bigCity")
CREATE DIMENSIONAL CONCEPT DocInBgCtySmCntry FOR Doctor_doctorCity AS INTERSECTION OF(DocInBigCity, DocInSmallCountry);	DocInBgCtySmCntry $\sqsubseteq$ Doctor_doctorCity DocInBgCtySmCntry $\equiv$ DocInBigCity $\sqcap$ DocInSmallCountry	create view "DocInBgCtySmCntry" as (select "Doctor" from "DocInBigCity") intersect (select "Doctor" from "DocInSmallCountry")

The translation of dimensional concepts to SQL views is presented in Table 6. A dimensional concept is represented by a single-column concept view which holds a subset of the extension of the node view of the level-range restricted dimension which represents its domain. Since entity IDs (primary key values in entity views) are also used as node IDs, entity concept views can easily be joined with node views (see 3b).

4.3 Multi-dimensional Concepts

A multi-dimensional concept mc is defined over a dimension space ds, its domain. Multi-dimensional concepts have an *inner* interpretation and an *outer* interpretation. The inner interpretation contains all points that have *exactly* the dimension roles of the dimension space and fulfill the restrictions on these dimension roles. The outer interpretation contains all points that have *at least* the dimension roles of the dimension space and fulfill the restrictions on these dimension roles.

The concept expression of a multi-dimensional concept (md-concept) is given in one of the following ways: (1) as primitive, (2) by an SQL query over the relational representation of the underlying OLAP cube (3) by a concept expression in one of the following forms, (3a) a tuple of references to dimensional concepts for some dimension roles in which case all points that satisfy the dimensional concepts in the indicated dimension roles are in the interpretation,

Table 9. MDO multi-dimensional concepts and their representation in OWL and SQL

	MDO	in **OWL**	in **SQL** (inner interpretation)
	CREATE MULTIDIMENSIONAL CONCEPT mc FOR ds ...	$mc \sqsubseteq ds$ — $dr_1, \ldots, dr_n$ are the dimension roles of ds. d_i is the dimension of dr_i.	$mc(\underline{dr_1, \ldots, dr_n})$
(1)	AS PRIMITIVE;		
(2)	AS SQL: *sqlquery*;		create view mc as *sqlquery* ;
(3)	AS	$mc \equiv$	create view mc as
(3a)	$(dr_1{:}dc_1, \ldots, dr_n{:}dc_n)$;	$\exists dr_1.\{nd_1\} \sqcap \cdots \sqcap \exists dr_n.\{nd_n\}$	select * from ds where dr_1 in (select * from dc_1) and ... and dr_n in (select * from dc_n);
(3b)	mc'^{*};	see text	select * from ds s, mc' c where (s.dr_1, c.dr_1) in (select * from d_1_rollup) and ... and (s.dr_n,c.dr_n) in (select * from d_n_rollup) ;
(3c)	$mc'[ds]$	$mc' \sqcap ds$	select * from mc' natural join ds;
(3d)	UNION OF (mc_1, ..., mc_m);	$mc_1 \sqcup \cdots \sqcup mc_m$	(select * from mc_1) union ... union (select * from mc_m);
(3e)	INTERSECTION OF (mc_1, ..., mc_m);	$mc_1 \sqcap \cdots \sqcap mc_m$	select * from mc_1 natural join ... natural join mc_m;
(3f)	NOT mc';	$ds \sqcap \neg mc'$	(select * from ds) except (select * from mc')
(3g)	$msr\ \theta\ value$	$\exists msr.[\theta\ value]$	select s.* from ds s natural join msr m where $msr\ op\ value$;

for example, leadDocInBgCtySmCntry in Table 10, (3b) by hierarchy expansion of an md-concept, such that each point that is in the interpretation of the md-concept or a descendent thereof is in the interpretation, for example, concept leadDocInBgCtySmCntryIn2010, (3c) by restriction to a dimension space which is restricted to a smaller granularity range, (3d) by disjunction of md-concepts with the same domain or (3e) by conjunction of md-concepts, (3f) by complement of an md-concept (open world interpretation), or (3g) by a boolean expression over measure-value comparisons of measures applied to a point, for example, expensiveDoctorCombos.

Md-concepts are divided into primitive and defined. Defined md-concepts are divided into dimension-based and fact-based. A dimension-based md-concept is an md-concept that is defined only by reference to dimensional concepts. The hierarchical expansion, disjunction, conjunction, and complement of dimension-based md-concepts is also a dimension-based md-concept. A fact-based md-concept is an md-concept that is defined by predicates on measure values.

Non hierarchically-expanded md-concepts are mapped to OWL according to Table 9 and subsumption checking is delegated to the OWL reasoner. In order for subsumption checking over hierarchically-expanded dimension-based md-concepts to be delegated to the OWL reasoner, the md-concepts must be transformed into disjunctive normal form (DNF). The DNF of a dimension-based md-concept *mc* is a disjunction of tuples of references to dimensional concepts (Equation 1). The conjunction of tuples of references to dimensional

Table 10. Examples of MDO multi-dimensional concepts

MDO	in **OWL**	in **SQL**
CREATE MULTIDIMENSIONAL CONCEPT leadDocInBgCtySmCntry FOR ds_leadDoc AS leadDoc:DocInBgCtySmCntry;	leadDocInBgCtySmCntry $\sqsubseteq$ ds_leadDoc leadDocInBgCtySmCntry $\equiv$ $\exists$ds_leadDoc .DocInBgCtySmCntry	create view "leadDocInBgCtySmCntry" as select * from leadDoc where leadDoc in (select * from "DocInBgCtySmCntry")
CREATE MULTIDIMENSIONAL CONCEPT leadDocInBgCtySmCntryIn2010 FOR ds_leadDoc_time AS INTERSECTION OF (leadDocInBgCtySmCntry, timeIn2010)) ;	leadDocInBgCtySmCntryIn2010 $\sqsubseteq$ ds_leadDoc_time leadDocInBgCtySmCntryIn2010 $\equiv$ leadDocInBgCtySmCntry $\sqcap$ timeIn2010	create view "leadDocInBgStySmCntryIn2010" as select * from "leadDocInBgCtySmCntry" natural join "timeIn2010"
CREATE MULTIDIMENSIONAL CONCEPT expensiveDoctorCombos FOR leadDoc_actDoc_year AS medianCostsPerIns $>$ 1000	expensiveDoctorCombos $\sqsubseteq$ leadDoc_actDoc_year expensiveDoctorCombos $\equiv$ medianCostsPerIns.double[$>$ 1000.0]	create view "expensiveDoctorCombos" as select * FROM "leadDoc_actDoc_year" natural join "medianCostsPerIns" where "medianCostsPerIns" $>$ 1000;

concepts is rewritten to a single tuple where dimensional concepts that are referred to by the same dimension role are conjoined (Equation 2). The DNF of the hierarchical expansion of mc corresponds to the DNF of mc with the referred dimensional concepts being hierarchically expanded (Equation 3). The DNF of mc restricted to a dimension space ds that is restricted to a smaller granularity range ($ds = dr_1[l_1..l'_1], \ldots, dr_n[l_n..l'_n]$) corresponds to the DNF of mc with the referred dimensional concepts being restricted accordingly (Equation 4). The DNF of the complement of mc corresponds to a disjunction of tuples of references to dimensional concepts. In this disjunction, every term is only restricted in one dimensional concept with the others left unrestricted (Equation 5). Remember, disjunction of md-concepts is only possible over md-concepts with the same dimension space.

$$DNF(mc) = \bigvee_{i=1..m} (dr_1{:}dc_{1_i}, \ldots, dr_n{:}dc_{n_i}) \tag{1}$$

$$\begin{aligned}&(dr_1{:}dc_1, \ldots, dr_k{:}dc_k, \ldots, dr_n{:}dc_n) \wedge (dr_1{:}dc'_1, \ldots, dr_k{:}dc'_k) = \\ &(dr_1{:}(dc_1 \wedge dc'_1), \ldots, dr_k{:}(dc_k \wedge dc'_k), \ldots, dr_n{:}dc_n)\end{aligned} \tag{2}$$

$$DNF(mc^*) = \bigvee_{i=1..m} (dr_1{:}dc^*_{1_i}, \ldots, dr_n{:}dc^*_{n_i}) \tag{3}$$

$$DNF(mc[ds]) = \bigvee_{i=1..m} (dr_1{:}dc_{1_i}[l_1..l'_1], \ldots, dr_n{:}dc_{n_i}[l_n..l'_n]) \tag{4}$$

$$\begin{aligned} DNF(\neg(dr_1{:}dc_1, \ldots, dr_k{:}dc_k, \ldots, dr_n{:}dc_n)) = \\ (dr_1{:}\neg dc_1, \ldots, dr_k{:}\top_k, \ldots, dr_n{:}\top_n) \vee \cdots \vee \\ (dr_1{:}\top_1, \ldots, dr_k{:}\neg dc_k, \ldots, dr_n{:}\top_n) \vee \cdots \vee \\ (dr_1{:}\top_1, \ldots, dr_k{:}\top_k, \ldots, dr_n{:}\neg dc_n) \end{aligned} \tag{5}$$

5 Prototype Implementation

In this section we describe the implementation of the approach as part of the semCockpit system prototype. The approach is implemented using off-the-shelf OWL reasoners (HermiT [16]) and database technology (Oracle DB 11.2g), the Java programming language and the OWL API [6]. The prototype architecture is geared towards functionality, rapid prototyping and experimentation. Performance considerations are widely neglected.

MDOs are persisted in a relational database called MDO DB. The schema of the MDO DB implements the abstract syntax (not discussed in this paper) of the MDO language. Every MDO language construct is represented by one or more tables in the MDO DB and every MDO object is represented by one or more tuples. By using transactional support of the relational database management system, MDOs can be maintained concurrently by different analysts.

The concrete MDO syntax (as presented in this paper) is implemented as ANTLR4 grammar. An MDO parser translates MDO statements to insert-statements against the MDO DB.

The semCockpit data warehouse (semDWH) holds the semantically enriched OLAP cube and provides relational views on dimensions, (derived) facts and MDO concepts as specified in Sections 3 and 4. In order to allow for rapid prototyping and experimentation with different language features, the OLAP cube is directly implemented as a set of SQL views over the asserted data. In future work, we will investigate platform-specific logical and physical optimizations.

The MDO DB also holds the OWL and SQL representations of MDO objects. The translations from MDO syntax to OWL and SQL (as specified in Tables 4, 6, and 9) are implemented by stored procedures within the MDO DB. When inserting or updating an MDO object, inserts or updates to the tables holding the OWL and SQL representations are triggered. This allows for the inspection and alteration of OWL and SQL representations of MDO concepts prior to invoking the reasoner and prior to creating or replacing views in the semDWH.

6 Related Work

The MASTRO system [5] for ontology-based data access uses ontologies for querying (relational) databases. Similarly, the semCockpit approach employs the MDO as a “high-level, conceptual view over data repositories”. In order to accommodate for the particularities of data warehousing and OLAP the semCockpit approach assumes complete data (closed world assumption) as a prerequisite for correct data aggregation. Furthermore, the MDO provides a set of primitives for the definition of the multi-dimensional and hierarchical space the multi-dimensional concepts are defined in.

Other work has investigated the difference between open-world ontologies and closed-world databases. Motik et al. [9] extend knowledge bases that are based on Description Logics with closed-world integrity constraints. Lim et al. [8] employ virtual views as a query interface for semantically-enriched relational data.

In the context of data warehousing and OLAP, the use of existing domain ontologies has been discussed from different perspectives. The AMDO method [15] allows for an automated derivation of conceptual multi-dimensional models from domain knowledge represented in ontologies. The DWOBS tool [7] supports the design of data warehouses from ontology-based operational databases. Pardillo and Mazón [14] propagate the use of ontologies to fix the shortcomings of traditional data warehouse design methods. In contrast, the semCockpit approach introduces MDOs specifically for knowledge sharing between business analysts.

The Multidimensional Integrated Ontology (MIO) [10] integrates various sources of knowledge and defines measures, dimensions, and hierarchies for OLAP over the different sources of knowledge. In contrast, MDOs do not integrate existing ontologies. Rather, MDOs define business terms, building on hierarchical and multi-dimensional primitives for querying traditional data warehouses.

The partitioning of the terminological box of an ontology [4] into a schema part and a view part, with distinct language constructs for either part, is of particular importance for MDOs. For an overview of the use of semantic web technologies in Business Intelligence we refer to Berlanga et al. [3].

7 Conclusion

A multi-dimensional ontology (MDO) is a formal representation of a conceptualization of a data analysis domain. This formal representation facilitates the unambiguous interpretation of query results and allows business analysts to share their knowledge in data analysis projects. By providing translations of MDO concepts to relational views, semantically-enriched OLAP cubes may be queried using SQL. By providing a representation of MDO in OWL, off-the-shelf reasoners may be used for the classification of business terms which is vital for the management of large vocabularies. Furthermore, the OWL representation of MDO concepts facilitates the integration of existing domain ontologies.

The MDO is the fundamental of ontology-driven comparative data analysis as investigated in the *Semantic Cockpit* (semCockpit) project. In the semCockpit system dimensions are extended by concept levels, the members of which are MDO concepts. In order to ensure summarizability the MDO concepts of a concept level must be pairwise disjoint. Checking disjointness of MDO concepts can be reduced to subsumption checking as provided by the approach presented in this paper. Semantic dimensions [2] allow for the use of external domain ontologies, for example, SNOMED CT, as dimensions of OLAP cubes. Having an OWL representation of MDO concepts serves as the basis for the seamless integration of semantic dimensions into the MDO. BI analysis graphs [11] extend the common set of OLAP operations, for example, slice and dice, with a set of operations for the navigation along subsumption hierarchies of MDO concepts.

References

1. Querying Dimensional Objects, Oracle OLAP User's Guide, 11g Release 2 (2010), `http://docs.oracle.com/cd/E11882_01/olap.112/e17123/query.htm`
2. Anderlik, S., Neumayr, B., Schrefl, M.: Using domain ontologies as semantic dimensions in data warehouses. In: Atzeni, P., Cheung, D., Ram, S. (eds.) ER 2012. LNCS, vol. 7532, pp. 88–101. Springer, Heidelberg (2012)
3. Berlanga, R., Romero, O., Simitsis, A., Nebot, V., Pedersen, T.B., Abello, A., Aramburu, M.J.: Semantic web technologies for business intelligence (2011)
4. Buchheit, M., Nutt, W., Donini, F.M., Schaerf, A.: Refining the structure of terminological systems: Terminology = schema + views. In: Hayes-Roth, B., Korf, R.E. (eds.) AAAI, pp. 199–204. AAAI Press/The MIT Press (1994)
5. Calvanese, D., Giacomo, G.D., Lembo, D., Lenzerini, M., Poggi, A., Rodriguez-Muro, M., Rosati, R., Ruzzi, M., Savo, D.F.: The MASTRO system for ontology-based data access. Semantic Web 2(1), 43–53 (2011)
6. Horridge, M., Bechhofer, S.: The OWL API: A Java API for OWL ontologies. Semantic Web 2(1), 11–21 (2011)
7. Khouri, S., Bellatreche, L.: DWOBS: Data warehouse design from ontology-based sources. In: Yu, J.X., Kim, M.H., Unland, R. (eds.) DASFAA 2011, Part II. LNCS, vol. 6588, pp. 438–441. Springer, Heidelberg (2011)
8. Lim, L., Wang, H., Wang, M.: Unifying data and domain knowledge using virtual views. In: VLDB, pp. 255–266 (2007)
9. Motik, B., Horrocks, I., Sattler, U.: Bridging the gap between OWL and relational databases. Journal of Web Semantics 7(2), 74–89 (2009)
10. Nebot, V., Llavori, R.B.: Building data warehouses with semantic web data. Decision Support Systems 52(4), 853–868 (2012)
11. Neuböck, T., Neumayr, B., Rossgatterer, T., Anderlik, S., Schrefl, M.: Multidimensional navigation modeling using BI Analysis Graphs. In: Castano, S., Vassiliadis, P., Lakshmanan, L.V., Lee, M.L. (eds.) ER Workshops 2012. LNCS, vol. 7518, pp. 162–171. Springer, Heidelberg (2012)
12. Neumayr, B., Anderlik, S., Schrefl, M.: Towards ontology-based olap: datalog-based reasoning over multidimensional ontologies. In: Song, I.Y., Golfarelli, M. (eds.) DOLAP, pp. 41–48. ACM (2012)
13. Neumayr, B., Schrefl, M., Linner, K.: Semantic Cockpit: An ontology-driven, interactive business intelligence tool for comparative data analysis. In: De Troyer, O., Bauzer Medeiros, C., Billen, R., Hallot, P., Simitsis, A., Van Mingroot, H. (eds.) ER 2011 Workshops. LNCS, vol. 6999, pp. 55–64. Springer, Heidelberg (2011)
14. Pardillo, J., Mazón, J.N.: Using ontologies for the design of data warehouses. International Journal of Database Management Systems 3(2), 73–87 (2011)
15. Romero, O., Abelló, A.: A framework for multidimensional design of data warehouses from ontologies. Data Knowledge Engineering 69(11), 1138–1157 (2010)
16. Shearer, R., Motik, B., Horrocks, I.: HermiT: A highly-efficient OWL reasoner. In: Dolbear, C., Ruttenberg, A., Sattler, U. (eds.) OWLED 2008. CEUR Workshop Proceedings, vol. 432. CEUR-WS.org (2008)

Ontology Driven Information Extraction from Tables Using Connectivity Analysis

Ashwin Bahulkar and Sreedhar Reddy

Tata Consultancy Services Ltd
54 B Hadapsar Industrial Estate
Pune, India
{ashwin.bahulkar,sreedhar.reddy}@tcs.com

Abstract. Table is one of the most common mechanisms used for presenting structured information on the web. A table presents information on a set of related concepts in a domain. A column typically represents a concept or an attribute of a concept that the column header identifies. A row contains corresponding instances and attribute values. However column headers are usually quite noisy and sometimes even missing. While a human reader can figure out the required domain mappings relatively easily by using domain knowledge and surrounding context, discovering them algorithmically poses challenges. In this paper we present an algorithm that exploits the idea that a table only presents information on connected entities of a domain ontology. The algorithm works in two phases. In the first phase it uses local optimization criteria such as lexical matching, instance matching, and so on to find an initial set of mappings. In the second phase it takes these mappings and constructs all possible connected sub graphs of the ontology that can be formed from these mappings. The largest of these sub graphs that has the highest local mapping score is then selected as the underlying domain mapping of the table. We present experimental results demonstrating the effectiveness of the algorithm.

Keywords: Ontology, information extraction, web tables.

1 Introduction

A table is a common means for presenting structured information in documents. Tables vary in complexity, from a simple two-column representation to a large multi-column representation with multi-row headers. The latter are especially common in scientific literature. A scientific domain such as bio-medical or materials science is characterized by a rich underlying ontology. In these domains a table may present information spanning multiple entities in the ontology. For example, in the materials domain, a table may present information on materials, their compositions, properties

Table 1. Table with unlabeled columns

Physical Property		
Density	0.84	[g/cm³]
Specific Volume	1.19	[cm³/g]

Table 2. Table with misleading column headers

Quantity	**Value**	**Unit**
Thermal Expansion	10-13	e-6/K
Thermal Conductivity	50-50	W/m.K
Specific heat	460-540	J/kg.K

Table 3. Table from Transport domain

No	**Name**	**Type**	**Zone**	**From**	**Dep**	**To**	**Arr**	**Duration**	**Halts**
19019	Mumbai Exp	Exp	WR	BDTS	00:05	NZM	05:25	29h 20 min	74
02455	Swaraj Express	SF	NR	BDTS	06:25	NDL	07:40	25h 15m	7
19023	Delhi Exp	Exp	WR	BDTS	07:25	NDL	12:45	29h 20min	67
12471	Rajdhani Exp	SF	WR	BDTS	07:55	NZM	04:30	20h 35min	15

of the materials and the conditions under which those properties are exhibited. This maps to several entities, their attributes and relationships in the materials ontology. These mappings are typically indicated by column labels. However, there is usually a large variation in the kinds of labels used. Short hand notations, truncated strings, etc are quite common. Sometimes labels may even be completely missing. These variations exist not only because there is no standard terminology for labels, but also to satisfy requirements of presentation such as layout constraints, etc. Despite this noise, a human reader can usually figure out the mappings easily, as he/she does not go by the labels alone but uses his/her domain knowledge and the knowledge of the context in which the table is presented. Hence an automated extraction algorithm that relies solely on labels and other local cues will not do well in the presence of such noise, especially when complex tables are involved. A table usually presents information on related entities. Many ambiguous situations can be easily resolved if we make the hypothesis that all information in a table is related. We call this "connectivity hypothesis". For example, in Table 2 we have a column named "Quantity". This does not map to any term in the materials ontology, a fragment of which is shown in Fig 1. However a table level connectivity analysis tells us that this most probably maps to "Property" as the other two columns are mapped to "Property Value" and "Unit". In Table 1, column 1 is labeled "Physical Property" while columns 2 and 3 are unlabeled. With local analysis (instance analysis) we can figure out that column 3 has units and thus label column 3 as "Unit". At this stage table connectivity analysis with respect to the ontology tells us that the unlabeled column can only map to "Property Value".

Table 3 shows a table from transportation domain. Column 2 is labeled “Name” which could potentially refer to name of a place, train or some other means of transportation. Columns after it are labeled “arr time”, “dep time”, “from” and “to”. Since these are only related to transportation mode, global analysis rules out place as a possible match for column 2. Looking further at the column labeled “zone” global analysis can infer that column 2 can only map to Train as zone is an attribute of Train.

As illustrated in the previous examples, connectivity hypothesis and global analysis not only help us figure out correct mappings for columns, but also figure out the relationships between the columns.

We present an algorithm that exploits the connectivity hypothesis to figure out correct mappings. The algorithm proceeds in two steps:

Step 1
In the first step, it uses local optimization criteria such as lexical matching, instance matching, and so on to find an initial set of mappings. Columns are mapped to classes and attributes in the ontology based on local cues such as the header and contents of a column. Lexical matching consists of matching the column header against the names of all classes and attributes in the ontology and selecting those that match above a threshold. Instance matching consists of matching cells against known instances in the dictionary. Cell contents are also matched against regular expressions and data types defined in the ontology. A weighted score is computed from these individual scores.

Step 2
In the second step, we resolve ambiguous mappings and find mappings for unmapped columns by doing global connectivity analysis. This is done by trying to find the largest connected sub graph of the ontology that covers the mappings discovered in the local matching step. There may be several candidate graphs of same size. We break the tie by taking the graph that has the highest aggregate local matching score.

Our experimental results show that this global approach does significantly better than approaches that rely purely on local cues. The improvement is more pronounced in the case of complex tables. We obtained an accuracy value close to 80% with the global approach as against an accuracy of around 60% with local approaches.

1.1 Application Context

The approach presented here is developed as part of an information management system that extracts and integrates information of a scientific domain from a number of heterogeneous sources. Queries in this domain can be complex spanning multiple entities with multiple conditions. For example, Table 5 shows information about 5 different concepts – an alloy, its composition, its properties, property values and the conditions under which those values are obtained. A typical query could ask for the name of an alloy with a particular composition, exhibiting a certain property at a certain temperature. Often information to satisfy such queries does not come from a single source. We might have to integrate information extracted from multiple websites.

Many of these websites are form based which return documents containing tables. The approach presented here is used for discovering mappings between such tables and ontology elements. These mappings, once discovered, can be reused for all future extractions from the site as pages are typically generated from the same underlying template.

2 Related Work

Limaye *et al.* [1] discuss a machine learning approach that uses a probabilistic graphical model for simultaneously choosing entities for cells, types for columns and relations for column pairs. As with any machine learning approach, the success of this depends on the quality of the seed catalog used for training. We propose a heuristic approach that exploits the connectivity hypothesis which seems to work quite well for complex tables.

Cafarella *et al.* [2] discuss an approach for querying the web for relevant web tables. Their approach collects corpus-wide statistics on co-occurrence of schema attributes, and uses these statistics to improve search relevance, synonym finding and join processing. The approach is meant primarily for returning right set of web tables to the user and not so much for automatically extracting information from these tables.

Embley *et al.* [3] present a set of heuristics for filtering relevant tables in HTML documents. We use a set of heuristics that are similar for filtering out irrelevant tables. The paper also presents an approach for extracting information from the tables. But they assume that each table presents information about just one fundamental concept and its attributes. Information on related concepts is expected to be found on separate linked pages. This is a limitation for processing complex tables such as the ones discussed in this paper where information on several related concepts is presented together.

Wang et al. [4] discuss an ontology based approach for extracting information from complex tables. They also present a grammar for complex table layouts and rules for transforming them into a standard form that is amenable for querying. They use ontology to identify concepts and concept instances represented by table cells. Concepts are recognized by their names and possible synonyms and instances are recognized by names or patterns. Where ambiguity arises, they use ontological closeness, i.e. how closely related the concepts of two adjacent cells are in the ontology hierarchy, to resolve the ambiguity. This is different from our approach where we exploit ontology to resolve ambiguities globally (i.e. not just adjacent cells) using the connectivity hypothesis.

A.Pivk et al.[8] discuss an approach to convert web tables into F-logic frames. While they exploit the information recorded in the WordNet ontology such as hyponym, hypernym, etc, the approach does not work with domain specific ontologies and so does not exploit the rich relational information present in them.

Web wrappers have traditionally been used to extract information from semi structured data sources, particularly web pages [5]. Wrappers evolved from being manually written to now being discovered through machine learning approaches. DIADEM [6] discusses an approach to extract data from semi structured web pages using an ontology centric approach. However this does not cover extraction from web tables.

We have not come across any work that uses connectivity analysis at a global level to figure out mappings for information extraction the way we do.

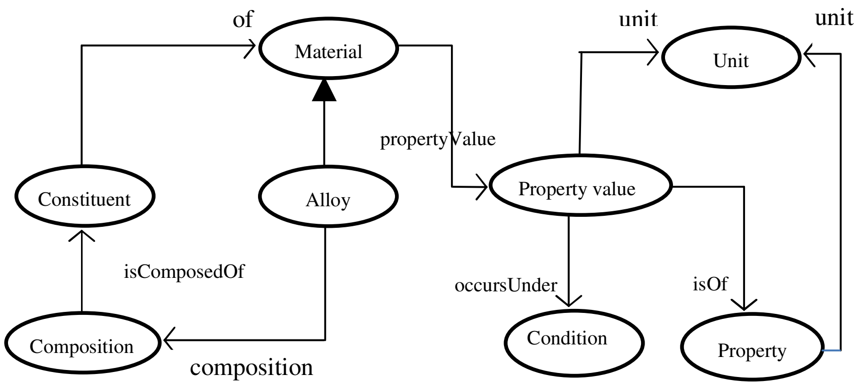


Fig. 1. The materials ontology

3 Ontology Enrichment

An ontology provides a conceptual model of a domain and defines the terminology for concepts, properties and relationships that exist in the domain. Fig 1 shows a fragment of the materials ontology we refer to in our examples. We enrich the standard ontology with the following annotations:

1. Synonyms of classes and attributes.
2. Regular expressions: to specify patterns for attribute values and instance names. We have extended the regular expressions language a little to allow expressions to refer to Ontology terms. For example, instances of the Property Value class might contain property values (numbers) followed by units, e.g. a weight value could be written as “5 kg”. In this case, a regular expression could be written as “[0-9]+ Inst(Unit)”. Here “Inst” specifies that the string should match with an instance of the class Unit. The string “5 kg” would match the regular expression if “kg” is known to be an instance of the class Unit.

4 Mappings Model

Fig 2 shows the model used for capturing mappings between a table and the domain ontology. A table may be embedded in a context that identifies an ontology object on which the table provides additional information. A column may be mapped to a class, attribute or object. A cell may be mapped to an object or an attribute value. When a column is mapped to a class, corresponding cells are mapped to objects of that class. When a column is mapped to an attribute, corresponding cells map to values of the attribute for objects contained in a related column. When no such related column exists, the attribute is assumed to be of the context object the table maps to. When a column maps to an object, corresponding cells map to objects that are related to the column object; the relationship is identified by the corresponding association object. The association object identifies the class of the column object, class of the cell objects and the association between the two classes. Two columns may be related to each other either by an association relationship or an attribute relationship; in the former case the two columns must map to the corresponding end classes of the identified association; in the latter case one column must map to a class and the other to an attribute of that class. A column may also be related to the context object of the table. Again this may be an association or attribute relationship with the same semantics as discussed above. The model in Fig 2 is intended for the case where columns are mapped to ontology elements. A similar model exists for the case where rows are mapped to ontology elements. We don't discuss this case in the paper as the treatment is similar.

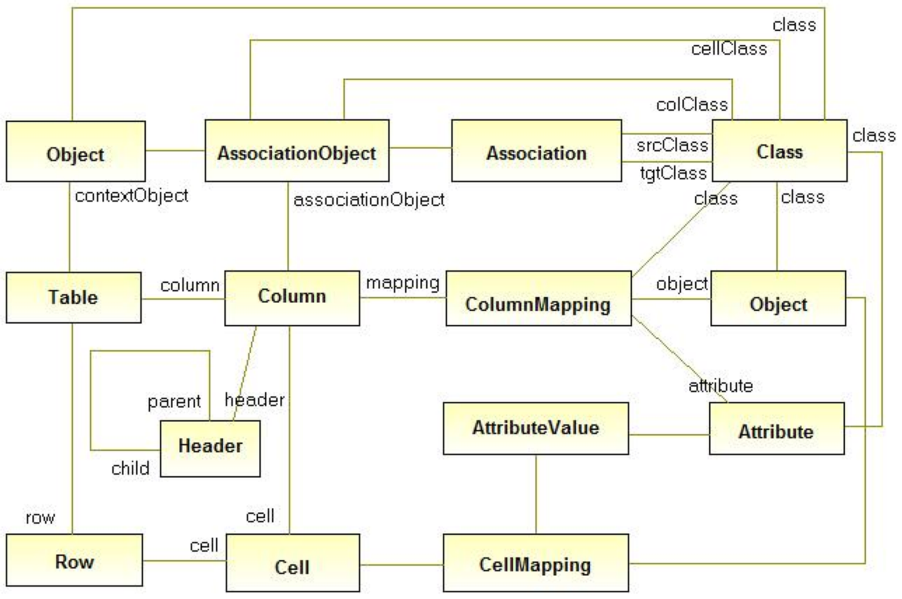


Fig. 2. Mappings Model

The objective of the mapping discovery algorithm is to discover an instance of this mappings model that is as close to the intended mapping as possible.

Before running the algorithm we preprocess the table to bring it into a normalized form. For instance we use the columnspan attribute to detect multi-row headers. A cell with a columnspan greater than 1 indicates that it has sub headers below it. We split such a cell into as many copies as the columnspan. Thus we only need to look at the cells in a single column to retrieve the header hierarchy of that column. There are a few other preprocessing steps such as removing visual attribute tags, etc. which we are not discussing further in the paper.

5 Algorithm for Discovering Mappings

A table generally presents information on related entities. We present an algorithm that exploits this notion of connectedness of information to figure out mappings. The algorithm proceeds in two phases:

Phase 1
In the first phase it uses local optimization criteria such as lexical matching, instance matching, and so on to find an initial set of mappings. Columns are mapped to classes and attributes in the ontology based on a score computed from local cues such as header matching and content matching consisting of instance matching, type matching, regular expression matching, etc.

Phase 2
In the second phase, we resolve ambiguous mappings and find mappings for unmapped columns by doing global connectivity analysis. This is done by trying to find the largest connected sub graph of the ontology that covers the mappings discovered in the local matching step. There may be several candidate graphs of same size. We break the tie by taking the graph that has the highest aggregate local matching score.

5.1 Phase 1 - Finding Local Cues

The following procedure computes a weighted sum of scores from various local cues. We try to find matching classes and attributes for each column. Class matching includes lexical matching as well as instance matching. Only classes whose lexical matching score is above a given threshold are considered as likely candidates. Attribute matching includes lexical matching, type matching and regex pattern matching. Only attributes whose lexical matching is above a given threshold are considered as likely candidates. For lexical matching we use a standard edit-distance based algorithm [7]. We also match column header against instance objects recorded in domain dictionary. When a column header maps to an instance object, we expect the cells to map to instance objects of a related class. This is done by taking the class of the header object and looking for classes related to it in the ontology. The classes whose instance matching and regex matching scores are above a threshold are considered as probable candidates for the cells.

The procedure finds all probable matches that satisfy threshold constraints. These are then pruned in the next phase.

The algorithm is presented in more detail below. In the algorithm we treat the model given in Fig 2 as a data structure, where associations and attributes are treated as members of the corresponding structure element.

```
procedure ComputeLocalMatchingScore
Input: Table tbl, Ontology ont
Output: A set of ColumnMapping elements
Begin
   For each column col in tbl.column do
      // Step 1: find potential matching classes
      For each class cls in ont do
           lms = LexMatch(col.header, cls.name)
           if (lms > lexicalMatchingThreshold) then
               // Get the percentage of column cells that
               // match instances of the class in the do
               // main dictionary
               ims = InstMatch(col, cls)
               // Get the percentage of column cells that
               // match the class name patterns, if
               // any specified for the class.
               rms = RegExMatch(col, cls)
               ws = lexMatchWeight * lms
                       + instMatchWeight * ims
                       + regexMatchWeight * rms;
               cm = new ColumnMapping
               cm.mappingScore = ws
               cm.class = cls
               col.mappimg = cm
           end if
      end for
      // Step 2: find potential matching attributes
      For each attribute attr of each class in ont
      do
           lms = LexMatch(col.header, attr.name)
           if (lms > lexicalMatchingThreshold) then
               rms = RegExMatch(col, attr)
              ws = LexMatchWeight * lms +
                    RegexMatchWeight * rms
               cm = new ColumnMapping
               cm.mappingScore = ws
               cm.attribute = attr
```

```
                col.mappimg = cm
            end if
        end for

        // Step 3: Next try to match column header with
        // objects (of any class) recorded in the domain
        Object objList[] = FindMatchingObject(col)
        For each obj in objList do
            cm = new ColumnMapping
            cm.object = obj
            col.mapping = cm

            // When a column header maps to an object, its
            // cells are expected to match with objects of
            // a related class. We try to find such a
            // matching related class. Again there could be
            // more than one probable match.
            For each relCls in
                   obj.class.association.tgtClass
            do
               // Consider classes whose combined instance
               // and regex matching scores are above a
               // threshold. Instance and regex matching
               // steps are as discussed previously for
               // class matching.
               instScore = getInstanceMatchScore(col,
                                                relCls);
               if (instScore > InstMatchThreshold) then
                   AssociationObject ascObj =
                                 new AssociationObject()
                   ascObj.association = assoc with relCls;
                   ascObj.columnClass = obj.class;
                   ascObj.cellClass = relCls;
                   col.associationObject = ascObj;
               end if
            end for
        end for
End

Procedure for lexical matching in a column:
procedure LexMatch
Input:  Header hdr, Class cls
Output:  matchScore : float.
Begin
    // Where we have multi-level headers, we start by
    // looking for a match for the most concrete class at
    // the leaf level and go up the hierarchy until we
    // find a match
    Header leafHdr = getLeafHeader(hdr);
    List clsNameList = getAllSynonyms(cls.name)
    **while** leafHdr is not NULL **do**
       // Get max matching score from the given list
       score = getEditDistance(leafHdr.label,clsNameList)
       **if** (score > lexicalMatchThreshold) **then**
          return score;
       **end if;**
       leafHdr = leafHdr->parent;
    **End while**
    return 0;
**End**
```

Illustration for Local Analysis

We illustrate how the algorithm works for the Table . 3. The table belongs to the Transport Ontology, which contains the classes Train, Bus, Flight journeys, each having attributes like arrival time, departure time, origin and destination place etc. The table gives information about different trains. Lexical matching recognizes the column headers “No”,”Name”,”Zone”, “From”, “To”, “Arr”, “Dep”, and it maps these columns to ontology attributes. Each mapping has a score which is based on regular expressions and instance matching for the cells in the column, for example the attribute “Arrival time” is associated with a regular expression, [0-2][0-3]:[0-5][0-9] which specifies clock time, and all the cells under the “Arr” column confirm to this regular expression. However, ambiguity exists because all of these attributes do not belong only to the Trains class in the Transportation ontology, they could belong to the Bus as well as Flight class. Only the “Zone” attribute is unique to the Train class. This ambiguity is resolved in the further stages of the algorithm.

In Table. 1, column labeled “Physical Property” is mapped to the Property Class, in the Material Science ontology, given in Fig 1. The 2nd and 3rd columns do not have a header, so they cannot be mapped to any ontology class or attribute in the local analysis phase. Mappings for these columns can be discovered only in the global analysis phase.

5.2 Phase 2: Analyzing Global Cues

This phase has two steps. In the first step we try to find mappings for columns that were not mapped in phase 1. This is done by trying out ontology elements that are in the vicinity of elements that are already mapped. This follows the intuition that a table presents information only on related ontology elements. Here again we try to find all probable matches for an unmapped column subject to threshold constraints.

In the second step, we try to discover the largest connected graph among the mappings discovered in the previous steps. This is done by taking each combination of possible column mappings and constructing connected graphs among them. A connected graph is constructed by taking the set of mapped classes and trying out all possible edges (corresponding to associations in the ontology) among them and taking the set of edges that results in the largest graph. After enumerating all such connected graphs among all possible combinations, we take the largest such connected graph as the final mapping. The largest connected graph may not be unique as there can be several graphs of the same size. We break the tie by taking the graph that has the highest aggregate local matching score.

Vicinity Analysis to Map Unmapped Columns

```
Procedure mapUnmappedColumns()
Input:
    Partially mapped table tbl, Ontology ont
Output:
    Table tbl with more refined mapping
Begin
    List<Column> colList = getUnmappedColumns(tbl)
    // Get the list of classes to which columns have
    // already been mapped.
    List<Class> clsList = getMappedClasses(tbl)

    // First check if the column matches with an
    // attribute of an already mapped class; record all
    // such matches as potential mappings.
    For each col in colList do
        For each cls in clsList do
            // First check if the column matches with an
            // attribute of an already mapped class;
            // record all such matches as potential
            // mappings.
            For each attr in cls.attribute do
               // Attribute matching score is computed as
               // discussed in function
               // ComputeLocalMatchingScore, except that
               // here there is no lexical matching
               score = GetAttributeMatchingScore(col,
                                                  attr)
               if (score > columnMatchThreshold) then
                  cm = new ColumnMapping
                  cm.mappingScore = score
                  cm.attribute = attr
```

```
                    col.mappimg = cm
                 end if
              end For

              // Next check if the column matches with an
              // associated class of an already mapped
              // class; record all such matches as
              // potential mappings.
              For each aCls in cls.association.class do
                // skip if an already mapped class
                if aCls is in clsList then
                    skip; // go to the next iteration
                end if
                // Class matching score is computed as
                // discussed in function
                // ComputeLocalMatchingScore, except that
                // here there is no lexical matching
                score = GetClassMatchingScore(col, cls)
                if (score > columnMatchThreshold) then
                    cm = new ColumnMapping
                    cm.mappingScore = score
                    cm.class = aCls
                    col.mapping = cm
                    skip;
                end if
                // Next check if the column matches with an
                // attribute of the associated class of an
                // already mapped class; record all such
                // matches as potential mappings.
                For each attr in aCls.attribute do
                  score = GetAttributeMatchingScore(col,
                                                    attr)
                  if (score > columnMatchThreshold) then
                     cm = new ColumnMapping
                     cm.mappingScore = score
                     cm.attribute = attr
                     col.mappimg = cm
                  end if
                end For
             End For
         End For
     End For
  End
```

Illustration for Vicinity Analysis

The Table 1, has 2 unmapped columns, and the 1st column has been mapped to the Property class. The Vicinity analysis phase then finds mappings for the 2nd and 3rd column in the vicinity of the Property class. Thus the classes Property Value and Unit are potential mappings for the 2nd and 3rd columns. For the 2nd column, the class Property value has a high local matching score, because most cells conform to the regular expressions specified for the Property Value class. The 3rd column maps to the Unit class because it contains several known instances of the Unit class.

Connectivity Analysis to Discover the Largest Connected Graph

The previous stages of the algorithm have yielded mappings for each column of the table along with the confidence levels for each mapping. A column may be mapped to multiple classes and attributes. These classes form the nodes of the potential mapping graph.

Our objective is to obtain the largest of these potential mapping graphs with the best local matching score. I.e. we want to discover a graph that maps the largest number of columns to ontology elements that form a connected graph. There can potentially be more than one connected graph of the same size. Among them we choose a graph whose nodes have the highest aggregate local matching scores.

```
Procedure FindMappingGraph
Input: Table tbl
Output: mapping graph
Begin
   Struct ColumnMapping {Column col, Class cls}
   // Generate all possible combinations of column-class
   // mappings. In the data structure below each
   // combination is a list of column-class mappings;
   // each such combination will have exactly one element
   // for each column.
   Struct ColumnMappingCombination List<ColumnMapping>
   List<ColumnMappingCombination> listOfColMapCmb =
           generateMappingCombinations(tbl)
   List<Graph> potentialGraphsList
   For each colMapCmb in listOfColMapCmb do
      Graph connectedGraph =
         getLargestConnectedGraph(colMapCmb)
      potentialGraphsList.add(connectedGraph)
   End For


   // By largest connected graph we mean a connected
   // graph with the largest number of nodes.
   List<Graph> largestGraphsList =
      findLargestGraphs(potentialGraphsList)
   float largestScore = 0;
   Graph selectedGraph;
   For each grph in largestGraphList do
     float curScore = getAggregateLocalScore(grph)
     if (curScore > largestScore) then
       selectedGraph = grph
     end if
   End For
   return selectedGraph
End
```

Illustration for Global Analysis

The global approach resolves the ambiguity introduced in Phase 1. Since there are multiple number of mappings for many columns, several potential graphs can be generated. Out of all of these, the graph which maps the columns labeled "Arr", "Dep", "From","To" to attributes of the Train class has the largest size because the column labeled "Zone" maps to only the attribute Train.zone, whereas the graphs which map the columns to attributes of the Bus or Flight class have smaller graph sizes, they map lesser number of columns. In addition to this, the column labeled "Name" contains the names of a few trains, which increases it's local matching score.

6 Experimental Results

We summarize the results of our experiments in this section. We have experimented with tables of varying complexity drawn from a number of pages of different websites corresponding to three different domains, namely materials science (mat-web.com, copper.org, matbase.com, keytometals.com, polymer.nims.go.jp), transport (Eurostar.com, indiarailinfo.com, Neetabus.in, seat61.com) and meteorology (imd.gov.in, climatemps.com, accuweather.com). We classified the tables into three categories: simple, moderately complex and complex. A simple table represented a single class and its attributes. A moderately complex table had two to three classes, their attributes and relationships. A complex table had four or more classes, their attributes and relationships. To give an example of a complex table, Table 5 has 6 classes and 6 associations with nested column headers. We have collected results for three different configurations of the algorithm:

- **Local-only mode:** only local cues, namely lexical matching, regular expressions and instance matching are used. I.e. only phase 1 of the algorithm is run.
- **Local+column-neighborhood analysis.** Here in addition to the above we also try to find mappings for unmapped columns by searching in the immediate vicinity of classes to which left and right columns are mapped. This is similar to the approach used in [4].
- **Local+global connectivity analysis.** This is the full-fledged algorithm where we try out various connected sub graphs to find the best matching subgraph.

Table 4 summarizes the results. Overall, across all tables, local-only approach gave us an accuracy score (f1-score) of 58.6%, local+column-neighborhood analysis an accuracy of 66.1% and local+global analysis an accuracy of 81.8%. For simple tables these figures were 80, 85.7 and 87.6 respectively, whereas for complex tables corresponding figures were 45.3, 48.9 and 72.2. The figures show that while the overall accuracy is much higher for simpler tables, the algorithm variants do not make that much of a difference. This is to be expected as connectivity analysis does not play a big role in the single class case, as mappings are limited to a single class. In the case of complex tables the global connectivity algorithm makes a big difference as can be seen from the score jumping from 45.3 to 72.2. This is because complex tables present a lot of ambiguous scenarios stemming from inaccurate and missing labels. Global connectivity analysis helps pin down such ambiguous cases by placing them in the connected context of the rest of the graph. The results also show that in the case of complex tables, adding column-neighborhood analysis does not improve the situation as substantially as the global connectivity analysis. To see why, let's look at columns "Dep" and "Arr" in the example of Table 3. By looking at their left and right columns we won't be able to resolve them to Departure and Arrival attributes of class Train. They could be the attributes of any other mode of transport. It is only through global connectivity analysis that we will be able to identify them as the attributes of Train which in turn we were able to identify as the only mode of transport in the table because 'zone' is an attribute only of the class Train.

In our experiments we tried out the following weightages for different local matching criteria:

Case 1: lexMatchWeight = 0.5; instMatchWeight = 0.25; regexMatchWeight = 0.25

Case 2: lexMatchWeight = 0.75; instMatchWeight = 0.125; regexMatchWeight = 0.125

Case 3: lexMatchWeight = 0.25; instMatchWeight = 0.375; regexMatchWeight = 0.375

Case 1 had an accuracy of 81.88%, case 2 had 80% and case 3 had 78%. These weightage differences did not have any substantial impact on the results (within an accuracy variation of ~4%). This reaffirms that graph connectivity is the dominant factor by far in resolving the ambiguous cases.

We have also investigated how the accuracy varies with the parameter *lexicalMatchingThreshold*. This value determines how strictly lexical matching is carried out. Lower values will result in more potential matches per column and hence increased ambiguity. In our experiments we observed for a *lexicalmatchingThreshold* value of 0.3 we got on an overall accuracy of 74%, whereas with a *lexicalmatchingThreshold* of 0.8 we got an overall accuracy of 81%. This shows the global connectivity algorithm is quite robust for large variations in ambiguity.

Results given in table 4 were obtained with the configuration <*legicalMatchingThreshold*=0.8, *lexMatchWeight*=0.5, *instMatchWeight*=0.25, *regexMatchWeight*=0.25>.

Table 4. F1 scores for various kinds of tables

	No. of tables	Local + global connectivity analysis	Local + Column-neighborhood analysis	Local-only
All tables	28	81.88	66.1	58.5
Simple tables	6	87.6	85.7	80
Moderately comlex Tables	13	82	72	58.6
Complex tables	9	72.2	48.9	45.3

Running Time. The complexity of the algorithm can increase significantly as we go from the local-only approach to global connectivity analysis. For an ontology of size N nodes and a table of M columns, the worst case complexity can be N**M. However, in practice a column does not have more than 3-4 ambiguous matches, so the average complexity is within acceptable limits. For example, in the case of the table in Table 5 from the materials domain, the running time for global connectivity analysis was 0.326 seconds as compared to 0.269 seconds for the local-only mode. This table has 14 columns and the corresponding ontology has 15 classes, 20 attributes and 21 associations.

Table 5. Complex Table

Copper and Copper Alloy		Condition	Composition, %						Test Temperature, K	Plastic Properties			
No	Name		Pb	Fe	Sn	Zn	Ni	P		Tensile Strength, psi	Yield Strength, psi	Elongation, % in 4D	Reduction of Area, %
102	Oxygen free	Cold Drawn 60%	4 ppm	4 ppm	1 ppm		4 ppm	1 ppm	295	48400	46800	17	77
									195	52900	49800	20	74
									76	66400	54400	29	78
									20	74500	58500	42	76

7 Conclusions and Future Work

We have presented an ontology based algorithm for information extraction from tables. The algorithm discovers mappings between table columns and ontology elements. It exploits the notion of connectedness of information in a table to resolve ambiguous mappings. The algorithm uses several strategies for local matching such as lexical matching, instance matching, regular expression pattern matching and so on. Despite the sophistication of local matching strategies complex tables usually throw up many ambiguous cases. Global connectivity analysis makes a substantial difference in resolving such cases. Our experimental results bear this out.

Going forward we would like to benchmark the effectiveness of our approach against machine learning based approaches such as the one discussed in [1]. We also want to test the algorithm on tables found in scientific publications. We want to integrate the table extraction algorithm with text based information extraction algorithms. We believe this integrated approach will increase the accuracy further as we can obtain additional context from the surrounding text.

References

1. Limaye, G., Sarawagi, S., Chakrabarti, S.: Annotating and Searching Web Tables Using Entities, Types and Relationships. Proceedings of the Very Large Data Bases Endowment 3(1) (2010)
2. Cafarella, M.J., Halevy, A., Wang, Z.D., Wu, E., Zhang, Y.: WebTables: Exploring the Power of Tables on the Web. In: Very Large Data Bases, Auckland, New Zealand (2008)
3. Embley, D.W., Tao, C., Liddle, S.W.: Automating the Extraction of Data from HTML Tables with Unknown Structure. Data & Knowledge Engineering - Special Issue 54(1) (July 2005)
4. Wang, H.L., Wu, S.H., Wang, K.K., Sung, C.L., Hsu, W.L., Shih, W.K.: Semantic Search on Internet Tabular Information Extraction for Answering Queries. In: Proceedings of the ACM CIKM International Conference on Information and Knowledge Management (2000)
5. Chang, C.-H., Kayed, M., Girgis, M.R., Shaalan, K.: Survey of Web Information Extraction Systems. IEEE Transactions on Knowledge and Data Engineering (2004)
6. Furche, T., Gottlob, G., et al.: DIADEM: Domain-centric, Intelligent, Automated Data Extraction Methodology. In: World Wide Web Conference – European Projects Track (2012)
7. Levenshtein distance. In: Black, P.E. (ed.) Dictionary of Algorithms and Data Structures, August 14, U.S. National Institute of Standards and Technology, Algorithms and Theory of Computation Handbook. CRC Press LLC (2008) (accessed October 31, 2011)
8. Pivk, A., Cimiano, P., Sure, Y.: From Tables to Frames. In: McIlraith, S.A., Plexousakis, D., van Harmelen, F. (eds.) ISWC 2004. LNCS, vol. 3298, pp. 166–181. Springer, Heidelberg (2004)

Efficient Projection of Ontologies

Julius Köpke, Johann Eder, and Michaela Schicho

Department of Informatics-Systems, Alpen-Adria Universität Klagenfurt, Austria
`firstname.lastname@aau.at`
`http://www.aau.at/isys`

Abstract. In collaborative and interoperable information systems it is frequently necessary to export a subset of an ontology, in particular to export all instances and axioms subsumed by a shared domain ontology. Technically this can be realized by projecting the Abox axioms of the private instances to the vocabulary of the shared ontology using a reasoner. We show that the direct application of this method results in a very redundant export. Therefore, we propose methods to generate a much smaller equivalent export in an efficient way. These methods are based on a deep analysis of properties of axiom sets that can be used to efficiently filter redundancies during the export process and provide methods to further minimize redundancies in the result. We evaluate the performance and the degree of minimality that can be reached with our approach with standard benchmarking ontologies with large Aboxes and a use case of the bio-medical domain.

Keywords: OWL, Ontology, Abox export, Abox minimizing, instance export, instance migration.

1 Introduction

For cooperative information systems it is frequently necessary to export parts of an ontology from one information system and import it in another (e.g. shared) information system. Here we focus on a particular situation where a local (or private) ontology is an extension of a shared (domain) ontology. The reasons for this situation may be manifold: the local information system might include other (upper) domain ontologies as well, the local ontology might be more fine grained than the shared ontology, the information shared should not be too detailed e.g. to protect trade secrets or privacy.

We motivate the projection of ontologies with a concrete use case: the sharing of data in federated biobanks. Biobanks are collections of biological material (Tissue, body fluids, cell cultures, etc.) and data describing this material and their donors (e.g health records, lab data, -omics data) [2,26]. Biobanks are very valuables resources for life science research providing material and data for biomedical research projects and studies. Currently, great efforts are undertaken to link biobanks creating an interconnected federated resource for biomedical research [25,7].

Creating a network of federated biobanks [21,7] requires the integration of heterogenous, autonomous, individually developed biobank databases [9,10]. Experiments based on data of (unknowingly) poor quality yield incorrect results [24]. One of the greatest challenges is to assure that data is correctly interpreted, data is correctly transformed between the different sites and it is possible to decide whether data stemming from different sites is comparable and can be used together for performing a biomedical study. This can be achieved by sharing not only data, but also the provenance [22] of the data and of the material in a provenance ontology. The provenance of a data item is the process which lead to this data item [4]. Thus, provenance is based on the question: *Who* has done something; *how, where*, and *when* was it done and based on *what*? As a consequence, provenance information is conceptualized using five interconnected elements including where, who, how, when and what [19] and is frequently represented in form of an ontology. However, since the local provenance ontology might contain more fine grained information, and not all information might be allowed to be shared for privacy reasons and ethical, and legal concerns [8], it is necessary to project the provenance information to the shared upper ontology of the federation.

For an example: the local biobank might store information that the diagnosis was made by John Smith, and we might infer that John Smith is pathologist, he has 10 years of experience, is 40 years old, etc. For the exported provenance ontology we would like to restrict the information to the fact that the diagnosis was made by a pathologist with more than 4 years of experience. This is the relevant information for the federation, as it allows to assign a credibility measure to the diagnosis. The other information is necessary for the local information system for various reasons, but it should not be passed on to the federation.
Data should only be exported if it is mapped to the federation (or domain) ontology and can be expressed in terms of this domain ontology. If we simply remove all explicit references to elements of the private ontology we will loose too much information. When the local instance data only addresses concepts and properties of the private ontology we would in fact loose all information (for example consider a medical case that only points to a local diagnosis ontology that is mapped to a global ontology. Simply striping away local assertions will result in instances without any assertions). Instead we need to rewrite the data that is only implicitly instance-data of the domain ontology to explicit instance data of the domain ontology that does not have any references to the private ontology.

In principal we need to export all Abox axioms regarding a set of instances of a private OWL [17] (DL direct semantics) ontology that only refer to the shared ontology. Since the number of such axioms is potentially infinite we restrict the exported axioms to Abox axioms that refer only to named entities. This is directly inline with our scenario and guarantees finite sets. This also implies that we only export axioms that refer to sub- or equivalent- roles / classes of the domain ontology which is sufficient for a large class of ontologies including our use-case. In order to also export axioms that refer to super-classes /-roles of the shared ontology the shared ontology can potentially be extended with such named concepts before the export is executed. Creating an export in a naive way results in very large and highly redundant sets of axioms. Straightforward redundancy removals on the other hand are highly inefficient. So the goal for the research reported here was to develop a method which is both efficient and which generates export sets with low redundancy. This new method is based on an in-depth analysis which properties of ontologies lead to redundancies in the export set. Exhaustive experiments with several ontologies show that this method is much more efficient that a straightforward approach and the remaining redundancy is marginal.

2 Abox Projection

2.1 Definition

We assume that the reader is familiar with the basic definitions of OWL (DL) [17] and description logics (see [14]) and only provide the essential definitions we need for defining the export problem formally here.

Definition 1. *The signature of an ontology O* is a set of named entities. This set can be further divided into the set of named classes denoted $O.sig.classes$, the set of roles $O.sig.Roles$ and the set of named individuals $O.sig.Individuals$.

Definition 2. *We restrict Abox axioms to the following axiom types*

- Class membership: $C(a)$ the named individual a is an instance of the named class C.
- Role assertions[1]: $R(a, b)$ defines that the named individual a is connected to the named individual b with the named role R. OWL allows to define object-properties as roles between individuals and data-type property roles as roles that connect individuals and typed literals.
- Equivalent individuals: $(a \approx b)$ defines that the named individuals a and b are equivalent.
- Different individuals: $(a \not\approx b)$ defines that the named individuals a and b are explicitly not equivalent. This is important due to the absence of the unique name assumption.

Definition 3. *An export set E of an ontology O* is a user defined subset of $O.sig.Individuals$

We can now formulate the problem of the projection:

Definition 4. *Given a private ontology O_k and an export set E of O_k and a public ontology O, the projection $P(E, O_k, O)$ is defined as a set of Abox axioms PA such that all Abox-axioms of O_k that refer only to the signature of $O \cup E$ are axioms of PA.*

[1] In contrast to OWL2 [17] we do currently not address the export of negative role assertions since they are not used by any of the relevant ontologies for our scenario.

This definition regards the ontologies O_k and O as logical theories and makes no distinction between axioms that are explicitly stated and axioms that are entailed.

2.2 A Naive Implementation

A straightforward implementation of the projection of a sub Abox to a sub ontology in OWL according to the definition is to iterate over all individuals of E and project the inferrable Abox axioms to the signature of O. Thus, only axioms that relate to classes, roles or individuals of $O \cup E$ are added to the projection. The corresponding algorithm is shown in algorithm 1. We show here only how to handle OWL object properties (= DL roles). Nevertheless, data-type properties can be handled in analogy.

Theorem 1. *The output of alg. 1 is valid according to the problem definition.*

Proof. All possible Abox axioms over named entities are addressed and it is guaranteed that only axioms that refer to classes and roles that belong to the signature of O are added and that only instances that are $\in E$ or that are $\in O$ are added. Since the algorithm checks the axioms by querying the entailments from the reasoner it makes no difference if an axiom a was explicitly stated in O_k or if it was inferred ($O_k \models a$). Therefore, also Abox-Axioms that relates to O which can only be inferred in O_k are made explicit in the projection

2.3 Minimal Projections for Data-Exchange

Unfortunately the naive solution produces a potentially very large number of exported axioms because it makes all axioms in the projection explicit. For example, given the class expressions *doctor* $\sqsubseteq$ *Actor*, *pathologist* $\sqsubseteq$ *doctor experienced_pathologist* $\sqsubseteq$ *pathologist* and an exported individual *Mr_Smith* with the type assertion *experienced_pathologist(Mr_Smith)* we will get the additional explicit assertions: *Actor(Mr_Smith), doctor(Mr_Smith),and pathologist(Mr_Smith)* in the projection. While in a minimal solution there should only be the axiom *experienced_pathologist(Mr_Smith)*. The restriction to named entities makes the projection finite but still much too large for an export format, where the size of the exported data matters. For exporting purposes the number of axioms should be limited to those that are essentially required to preserve all entailments that are possible in the full projection. Therefore, we now define the minimality of axioms of one individual.

Definition 5. *Minimal set of Abox axioms of an individual:* The set A_i of all Abox axioms that refer to an individual $I \in O$ is minimal if there does not exist an Abox axiom $a \in A_i$ such that $O \setminus \{a\} \models a$.

Based on this definition we can define a minimal projection:

Algorithm 1. A naive implementation of the projection

```
procedure NaiveProjection($E$, $O_k$, $O$, $PA$)
    $PA = \{\}$
    for all $i \in E$ do
        for all $C \in O.sig.Classes$ do
            if $O_k \models C(i)$ then
                $PA = PA \cup \{C(i)\}$
            end if
        end for
        for all $R \in O.sig.Roles$ do
            for all $j \in O.sig.Individuals \cup E$ do
                if $O_k \models R(i,j)$ then
                    $PA = PA \cup \{R(i,j)\}$
                end if
            end for
        end for
        for all $j \in O.sig.Individuals \cup E$ do
            if $i \neq j \wedge O_k \models i \approx j$ then
                $PA = PA \cup \{i \approx j\}$
            end if
            if $O_k \models i \not\approx j$ then
                $PA = PA \cup \{i \not\approx j\}$
            end if
        end for
    end for
end procedure
```

Definition 6. *A minimal projection $P_{min}(E, O_k, O)$ is defined as a subset of of the projection $P(E, O_k, O)$ such that $P_{min}(E, O_k, O) = \bigcup_{i=1}^{|E|} A_i$ where A_i denotes one minimal set of Abox axioms of the individual $i \in P(E, O_k, O)$.*

Obviously, since there can be multiple minimal sets of Abox axioms for each individual a minimal projection is not unique. In this work we are interested in finding one minimal projection that is used for data exchange.

A black-box algorithm that computes a minimal projection for a given projection P and a target ontology O can be implemented straightforward. The corresponding algorithm is shown in algorithm 2. It is inspired by an algorithm for the construction of minimal unsatisfiable sub Tboxes of [11]. It simply iterates over all individuals in the signature of P and removes each explicitly defined axiom that can still be entailed when the explicit axiom is removed.

While this black-box solution obviously achieves a minimal projection according to the definition it has the drawback of a simply unfeasible performance. Each removal of an axiom from the projection requires to restart/re-synchronize the reasoner which is unfeasible for larger ontologies (see section 4).

Algorithm 2. Minimizing a projection P

```
1: procedure MinimizeProjection($P$, $O$)
2:     for all $i \in P.sig.Individuals$ do
3:         for all $a \in Axioms \in P$ $over$ $i$ do
4:             if $O \setminus a_i \vDash a_i$. then
5:                 remove $a_i$ from $P$
6:             end if
7:         end for
8:     end for
9: end procedure
```

3 Efficient Export of Sub Aboxes

The output of the naive exporter is very redundant. We could also show, that a black box minimizer is unfeasible. We are now interested in strongly reducing the number of axioms in the projection without changing the entailments. We call axioms that can be removed without modifying the entailments redundant axioms. We now discuss properties of a projection that can be exploited to generate a far less redundant projection with an incremental algorithm that aims to minimize the calls to the reasoner to provide proper scalability in section 3.1. The incremental algorithm never removes already inserted axioms from the generated output. This approach allows a very fast export but some reasons for redundancies cannot be processed in an incremental way. Therefore, we will discuss reasons for such redundancies and propose a post processing algorithm that removes such redundancies from the export in section 3.2.

3.1 Improving the Generation of the Projection

In this section we discuss properties of a projection that can be used to already reduce the number of redundant axioms using an incremental export algorithm.

Avoiding Redundancies due to the Class or Property Hierarchy

Theorem 2. *A type assertion for an individual x, $C(x)$ in a projection $P(E, O_k, O)$ is redundant, if there exists a type assertion $D(x) \in P(E, O_k, O)$ and the target ontology O entails $D \sqsubseteq C$.*

Proof. When the result of the projection $P(E, O_k, O)$ is merged with the target ontology O, all superclasses of C are implicitly assigned as types of x. Therefore, their explicit definition is redundant.

Theorem 3. *A role assertion $R(x, y)$ for an individual x to another individual y in a projection $P(E, O_k, O)$ is redundant, if there exists another role assertion $S(x, y)$ in $P(E, O_k, O)$ and the target ontology O entails $S \sqsubseteq R$.*

Proof. When the result of the projection $P(E, O_k, O)$ is merged with the target ontology O, the axiom $R(x, y)$ is entailed by $S(x, y)$ and $S \sqsubseteq R$. Therefore, the explicit definition of $R(x, y)$ is redundant.

Theorem 4. *An equivalent- or subclass S of a class C has always equivalent or less direct or indirect descendants than C .*

Proof. Any subclass S_2 of a subclass S_1 is always also a subclass of the superclass S of S_1. Therefore, the superclass must have more child classes than any of its subclasses. Cyclic subclass relations do not change this behavior because in DL classes that are pairwise subclasses of each other are equivalent classes. Equivalent classes have the same number of descendants.

We could simply remove the previously discussed redundancies from a given projection by a minimizer that computes the transitive reduction [1] of the type or role assertions. Algorithmically this can be realized by a nested loop over the classes or roles and corresponding entailment checks. While this is still much more feasible than the black-box minimizer it still requires a high number of reasoner calls. Therefore, we exploit theorem 4 to get a more efficient exporter: We can sort the types and roles ascending by the number of subclasses/subroles. In a next step an advanced exporter can iterate over the sorted subroles/subclasses and only insert classes or roles when no subclass or role of the current class/roles has already been inserted. For role assertions we also need to ensure, that the role points to the same entity. Our improved algorithm effectively generates the transitive reduction without computing the transitive reduction in a nested loop fashion. While this improvement already strongly avoids redundancies - especially for ontologies with big T/R-boxes, there are still other reasons for redundant type assertions:

Avoiding Redundancies due to Role Assertions

Theorem 5. *A type assertion of an individual a, $C(a)$ is redundant in the projection $P(E, O_k, O)$, if there exists a role assertion $R(a, b)$ and C is an equivalent or superclass of the domain of R.*

Proof. The domain C of a property P is defined as $\exists P.\top \sqsubseteq C$. Consequently it is redundant to explicitly state that an individual that has a property assertion $P(a, \top)$ is of the type C.

Theorem 6. *An explicit role assertion $R(a, b)$ in a projection $P(E, O_k, O)$ is redundant, if there exists an explicit role assertion $R^{-1}(b, a)$ in $P(E, O_k, O)$.*

Proof. The proof directly follows from the definition of inverse roles.

Theorem 7. *An explicit role assertion $R(a, b)$ in a projection $P(E, O_k, O)$ is redundant, if there exists an explicit role assertion $R(b, a)$ in $P(E, O_k, O)$ and R is defined to be symmetric.*

Proof. The proof directly follows from the definition of symmetry.

Theorem 8. *For every symmetric role R, there exists an inverse role R^{-1} such that $R^{-1} = R$*

Proof. The proof directly follows from the definition of symmetric roles.

Theorem 9. *An explicit role assertion of an individual a, $R(a, b)$ is redundant in the projection $P(E, O_k, O)$, if there exists an explicit role assertion $R'(a, c)$ and $O \models \{R \equiv R'\}$ and $O \models \{b \approx c\}$*

Proof. The proof is a direct consequence of the definition of equivalence of individuals.

Theorem 10. *An explicit role assertion of an individual a, $R(a, b)$ is redundant in the projection $P(E, O_k, O)$, if the role is reflexive in O and a is in the domain of R.*

Proof. The proof is a direct consequence of the definition of reflexivity in OWL-DL: A reflexive property p with the domain d is defined as $d \sqsubseteq \exists p.Self$. Consequently the explicit definition of p to any individual of the type d is redundant.

An improved exporter can use theorem 5 by only asserting types to the exported individuals that are not equivalent to the domain of any assigned property. Theorems 6 and 8 allow us to add only property assertions to the projection, where the inverse has not yet been added. According to theorem 8 this also handles redundancies that are induced due to symmetric properties. Actually it is sufficient to only handle inverse properties. Theorem 9 and 10 allow the exporter to realize the following optimization: Assertions to reflexive roles of the form $R(a, b)$, where a is the current individual and $a \approx b$ are skipped if a is in the domain of R and role assertions are only exported once for each equivalent individual.

Avoiding Redundancies due to Same Individuals Assertions

Theorem 11. *An explicit same individual assertion $a \approx c$ in a projection $P(E, O_k, O)$ is redundant if there exists an assertion $a \approx b$ and an assertion $b \approx c$ in $P(E, O_k, O) \cup O$.*

Proof. The equivalence relation is transitive. Therefore, any assertion that is not an element of the transitive reduction is redundant.

We can now use theorem 11 for an efficient exporter that avoids the described redundancies already during the export. We need a way to ensure, that we do not export redundant relations due to transitivity. The problem here is that we cannot calculate the transitive reduction of the axioms in the projection unless the projection is complete. Fortunately, there is another property that allows an efficient export:

Theorem 12. *An explicit same individual assertion $a \approx b$ in a $P(E, O_k, O)$ is redundant if there exists an assertion $b \approx a$ in the projection.*

Proof. The equivalence relation is symmetric.

Exploiting theorem 11 and theorem 12 allows us to apply the following approach: Export only equivalence axioms for an individual j, if there exists no equivalence axiom of the form $k \approx j$ in the current set of axioms of the incrementally generated projection, where k is already an element of the projection. According to theorem 12 it is equivalent to query for $k \approx j$ or to query for $j \approx k$. Fortunately $k \approx j$ is already part of the projection because when k was exported all equivalence axioms that can be entailed in $O_k \cup E$ were added to the projection.

We can actually broaden theorem 11 to any axiom over an equivalent individual:

Theorem 13. *Given the individuals a and c, where $a \approx c$. Any explicitly stated axiom over c is redundant, if there exists a corresponding axiom over a.*

Proof. The proof is a direct consequence of the definition of equivalence.

We can now exploit theorem 13 to make the exporting algorithm even more efficient: We only export any type of axiom over an individual j, if there exists no equivalence axiom of the form $k \approx j$ in the current set of axioms of the incrementally generated projection, where k is already an element of the projection.

3.2 Efficient Removal of Redundant Assertions

While the discussed properties of a projection can already be used to produce a pre-minimized export, the result is not yet minimal because there are reasons for redundancies that cannot be removed during the incremental creation of the projection. For a complete minimization we can of course again use our black-box algorithm 2. The combination of the improved exporter and the black-box minimizer is much faster than minimizing a completely redundant projection because less axioms need to be tested. Nevertheless, there are still properties that can be used to efficiently remove redundancies from the projection:

Removing Redundancies due to Transitive Roles

Theorem 14. *An explicit role assertion of an individual a, $R(a, c)$ is redundant in the projection $P(E, O_k, O)$, if there exists a role assertions $R(a, b)$ and $R(b, c)$ in $P(E, O_k, O) \cup O$ and R is a transitive role.*

Proof. The proof is a direct consequence of the definition of transitivity.

We will now show as one example why redundant role assertions that are induced by transitive roles cannot be avoided by an incremental export algorithm.

Theorem 15. *The reduction of transitive role assertions cannot be done incrementally during the export.*

Proof. We provide the proof using a counter-example: We assume that we first add the axioms to an individual $R(a, c)$ and $R(a, b)$. Unfortunately we did not yet export the individual b. Therefore, we cannot find the later inserted axiom $R(b, c)$ which would render $R(a, c)$ redundant in the incrementally produced projection. We could still change the order of exports and export c and b first. This could solve the example problem. But it is not a general solution because we may have a cycle in the property assertions that prevents us from finding a suitable order: Given the transitive and disjoint roles R and S and the axioms $R(a, b)$, $R(b, c)$, $R(c, d)$, $R(a, c)$, $R(a, d)$, $R(b, d)$ and $S(d, c)$, $S(c, b)$, $S(b, a)$, $S(d, b)$, $S(d, a)$, $S(c, a)$. In order to eliminate the redundancies of R we would need to begin the incremental export with d, in order to eliminate those of S we would need to start with a, which is impossible for an incremental export.

According to the theorem we cannot guarantee that the transitive role assertions are redundancy free when the check is done incrementally during the creation of the projection. In order to remove redundancies that are induced by transitive properties we use a simple nested loop algorithm that iterates over all transitive properties and all pairs of individuals in order to compute the transitive reduction [1] of the property assertions.

Another source of redundancies that cannot incrementally be handled during the export are redundancies that occur due to (global) ranges of properties (roles):

Removing Redundancies due to Ranges of Roles

Theorem 16. *An explicit type assertion of an individual a, $C(a)$ is redundant in the projection $P(E, O_k, O)$ if there exists a role assertion $R(b, a)$ in $P(E, O_k, O) \cup O$ and C is equivalent to the range of R.*

Proof. The range C of a role R is defined in DL as: $\top \sqsubseteq \forall R.C$. Consequently it is redundant to specifically state that an instance a that is the target of a role assertion $R(x, a)$ is of the type C.

A minimizing algorithm can iterate over all role assertions and remove corresponding type assertions of the individuals that are in the range of the property assertions. A similar behavior exists for local ranges of a property:

Theorem 17. *An explicit type assertion of an individual a, $C(a)$ is redundant in the projection $P(E, O_k, O)$ if there exists a role assertion $R(b, a)$ in $P(E, O_k, O) \cup O$ and there exists a qualified universal role restriction on the class of b of the form $b \sqsubseteq \forall R.C$*

Proof. The proof is in analogy to the one of theorem 16 with the only difference that instead of the top class a specific class is used.

This property can easily be integrated into the proposed algorithm for the removal of redundancies that are induced by global ranges by additionally querying universal restrictions on the current individual.

Theorem 18. *The reduction of type assertions that are redundant due to local or global property ranges cannot be realized incrementally.*

Proof. As shown in proof of theorem 15 a cyclic dependency between properties cannot be resolved in an incremental way. The proof for the removal of redundant type assertions due to local or global ranges can be done in analogy.

Removing Redundant Same Individuals Assertions

Theorem 19. *An explicit same individual assertion $a \approx c$ or $c \approx a$ in the projection $P(E, O_k, O)$ is redundant if there exists a functional role assertion $R(a, x)$ and $R(c, y) \in P(E, O_k, O) \cup O$ and $x \approx y$.*

Proof. The proof is the direct consequence of the definition of functional roles.

This property can be used to minimize the set of same individuals by nested loops over each functional property and each individual with each other individual.

Theorem 20. *The removal of redundant same individual assertions due to functional properties cannot be realized with an incremental projection algorithm.*

Proof. The incremental algorithm exports each complete individual in E and never removes already exported axioms. It will export the individuals beginning at i_1 and ending at i_n. Given the role assertion $R(i_n, i_1)$ and R is a functional property and $R(i_{n-1}, i_1)$ it could potentially not export the same individual assertion $i_{n-1} \approx i_n$ but it will export $i_{n-1} \approx i_n$ because, when processing i_{n-1} it has no knowledge about i_n.

4 Evaluation

The discussed properties of Abox projections can be used to efficiently reduce the number of exported axioms. We have focused on frequent reasons for large numbers of redundant axioms but did not intend to guarantee the general minimality for the efficient algorithms. For example we do not consider redundancies that are induced by data-type / data-value reasoning and we do not handle redundancies that are inducted by property chains yet.

We have implemented both naive algorithms and an efficient exporter that exploits the theorems of section 3.1 and an efficient pre-minimizer according to the theorems in section 3.2. All algorithms were implemented in Java using the

Manchester OWL API[2] together with the pellet[3] reasoner. The experiments were executed on a virtual machine with dedicated 8GB of RAM and two dedicated CPU cores (AMD Opteron 2382 @ 2,61 Ghz). The aim of the evaluation was to evaluate the applicability of the proposed algorithms with regard to the performance (exported instances/second) and the degree of minimality that can be reached by our implementation. We used different real-world and benchmarking ontologies with big Aboxes (LUBM[4], Financial[5], Restaurant[6] and a generated dataset of provenance data based on ICD-10[7]) to cover a broad spectrum of ontologies.

4.1 Experimental Results

Scalability with regard to Tbox and export size: Our first experiment is the export of randomly generated data of patients, their diagnosis and their medical doctors in analogy to the example in section 1. The private source data only refers to private concepts and roles that are an extension of the target ontology using a generated nested hierarchy of 10 subclasses / subroles. To evaluate the performance with regard to big Tboxes we used an OWL version of the ICD 10 disease classification for the diagnosis. ICD 10 is organized in different chapters. To evaluate the scalability with regard to the size of the Tbox we executed the tests with different numbers of chapters. The Abox for all tests has a total size of 15,143 instances. The results are shown in figure 1. It shows the number of exported instances per second (exp. scale!) for different numbers of exp. individuals (x-axis) and different Tbox-sizes (A-F: 2907 classes, A -J: 4516 classes, A-O: 7058 classes and ICD all 14509 classes) and different algorithms. The efficient exporter together with the efficient minimizer achieves a performance between 3 and 36 instances per second. The results of the simple but full minimizer implementation that is executed on the result of the efficient exporter and efficient minimizer reaches a performance of 0.8 and 0.05 instances per second. Actually it turned out that for all experiments with the ICD 10 ontology the result of the efficient exporter with the efficient minimizer was already minimal. The experiments show that the efficient exporter reaches a good and nearly linear performance with regard to the number of exported instances. Only the startup costs make the export of very small numbers of instances inefficient. In contrast the full minimizer shows an exponential decrease in speed when the number of exported instances growth. We have also partly evaluated the naive exporter with the full minimizer. While the naive exporter performs better than the efficient exporter with regard to inst./sec. the minimizer gets much slower when executed on a large set of axioms.

[2] http://owlapi.sourceforge.net
[3] http://clarkparsia.com/pellet
[4] http://swat.cse.lehigh.edu/projects/lubm/
[5] http://www.cs.put.poznan.pl/alawrynowicz/semintec.htm
[6] https://files.ifi.uzh.ch/ddis/oldweb/ddis/fileadmin/ont/nli/restaurant.owl
[7] https://dkm.fbk.eu/index.php/ICD-10_Ontology

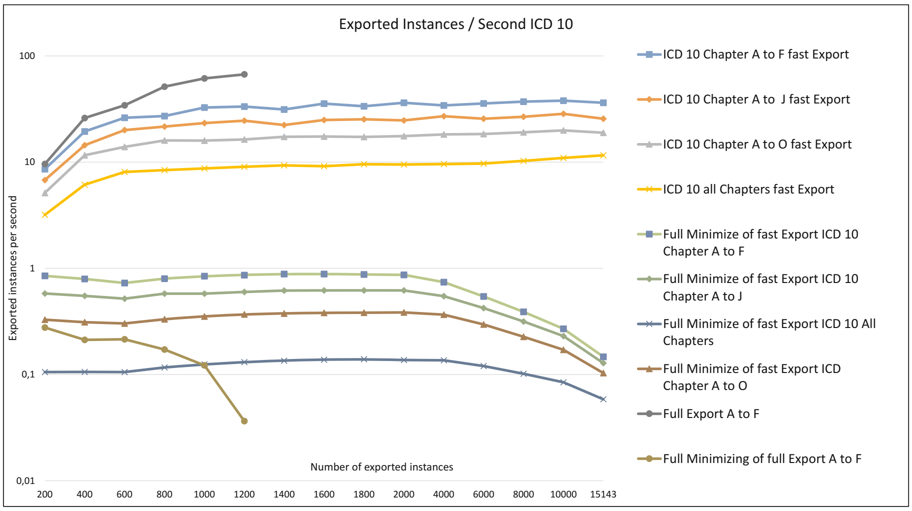
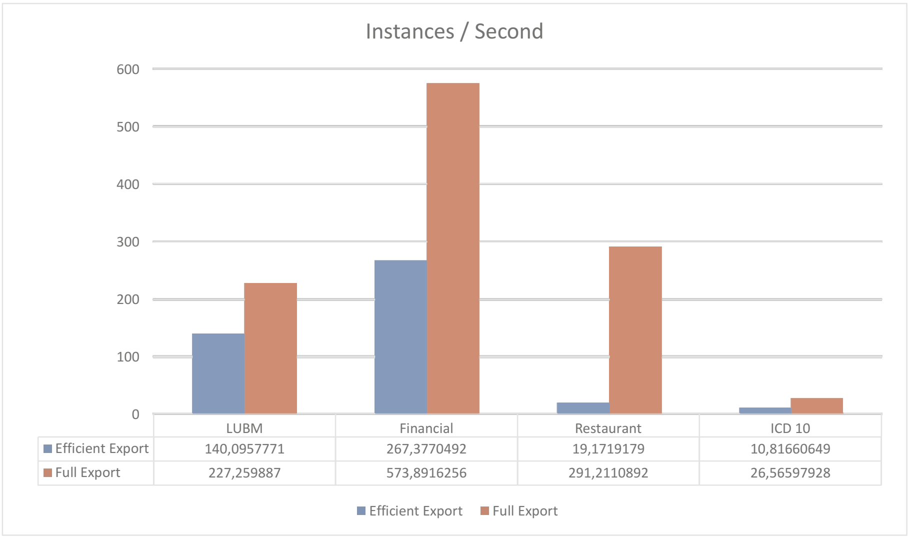


Fig. 1. Instances per second ICD10 with various Tbox sizes

Export performance and degree of redundancy with various benchmarking ontologies: The second experiment compares the number of redundant axioms and the exported axioms per second of our efficient algorithm with the naive implementation. We have used typical benchmarking ontologies with large numbers of instances (LUBM (1 University, 1609 instances), Financial (17941 instances), Restaurant (9748 instances) and the ICD 10 testcase from the last experiment (15092 instances)). All ontologies were extended with a hierarchy of 10 subclasses / subroles and all instances were rewritten to only refer to the extended (private) classes and roles. We have always exported the whole Abox to a target ontology with an initially empty Abox. The number of exported axioms of our efficient exporter in comparison to the naive exporter for the different datasets is shown in figure 3. It is obvious that the structure of the ontology has a large impact on the number of redundant axioms that can be saved. The highest number of savings could be achieved in the ICD 10 test case due to the large Tbox. The redundancy of the restaurant ontology is also big because many redundancies are caused by transitive and inverse roles. The efficient exporter creates an export set that is around 70% smaller in comparison to the full export. In case of the financial ontology the number of exported axioms is still around 50% smaller. In case of the LUBM dataset the efficient exporter saves around 35% of the axioms of the full export.

The reduction of the number of axioms has also consequences for the file size of the export depending on the serialization method (we use the RDF/XML serialization). In the ICD 10 test case, the fully redundant export file (only Abox) has a size of 22.4 MB. In comparison the minimized set has a size of 7.58 MB. Therefore, it is around three times smaller. The same ratio holds, when both files are zipped (647 KB vs. 211 KB). The reduced size has also consequences



	LUBM	Financial	Restaurant	ICD 10
Efficient Export	140,0957771	267,3770492	19,1719179	10,81660649
Full Export	227,259887	573,8916256	291,2110892	26,56597928

Fig. 2. Exported instances per second full vs. efficient export

for the time that is required to load the ontology into the target system. In an average of 100 experiments the time to load the minimized Abox was 0.8 seconds in comparison to 2.1 seconds for the non minimized version. Therefore, the minimized version can be loaded 2.6 times faster than the full export.

Of course the efficient export requires more time to compute the output. The comparison of both approaches with regard to exported instances per second is shown in figure 2. While for LUBM, ICD 10 and Financial the number of instances per second of the efficient exporter is moderately lower (LUBM: 38%, ICD 10: 59%, Financial: 53%) the performance for the Restaurant data-set is 93% lower. This limited performance for the restaurant ontology is due to the fact that the restaurant ontology heavily uses inverse and transitive properties. All values for instances per second include the time for the efficient export and the post-minimizer. The amount of time for the post-minimizer in comparison to the total time was 17% for LUBM, 93% for Restaurant, 40% for Financial, and 5% for ICD-10.

Degree of minimality. We have executed the black-box minimizer on all results to check the degree of minimality that could be achieved by our solution. In fact we produced minimal results for the ICD 10 and Restaurant datasets. For the LUBM benchmark the efficient exporter and minimizer did not eliminate one redundant axiom of 8000 axioms. In case of the Financial dataset our efficient algorithms did not eliminate around 50 redundant axioms of 50000. The cause of the redundant axioms are redundant type assertion due to defined (top-level) classes. In this case it can be deduced by the properties/roles of an instance that it has a specific type. Therefore, an additional assertion to this type is redundant. While one could argue that this is a major shortcoming of our approach it does

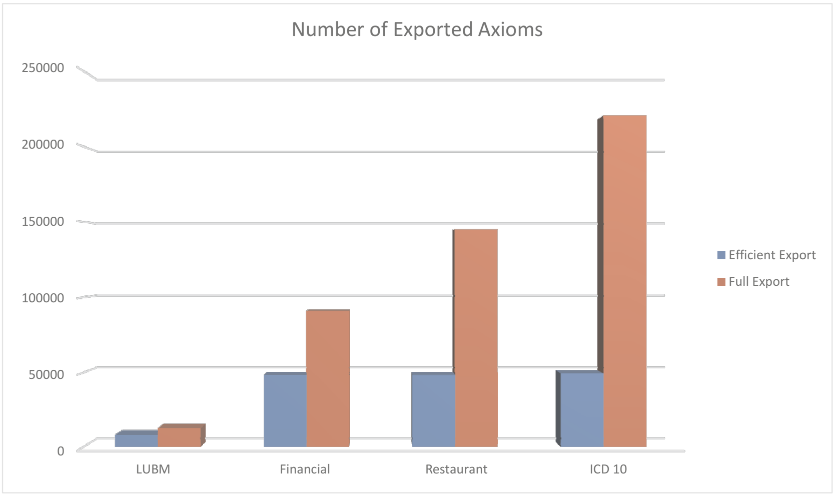


Fig. 3. Number of exported axioms full vs. efficient export

not really result in big numbers of redundant axioms (1 of 8000 in case of LUBM and 50 of 50000 in case of financial). The reason for the still very low degree of redundancy is that in many cases our approach can still reach minimality when defined classes are used and the definition also addresses other classes in the hierarchy. For example a type assertion to the defined class *youngperson (person and hasAge* $\leq$ *20)* will not be redundant, for a *person* of the age of 10 because a *youngperson* is a subclass of *person*. Other redundancies that we do currently not address are redundancies by OWL 2 object property chains. However they can potentially be addressed in analogy to the processing of redundancies due to transitive properties.

Different Individuals Axioms. Ontologies, where all sister classes are defined to be disjoint result in individuals with a very large set of different individuals assertions. Those are typically all redundant because they are implicitly defined by the Tbox of the shared ontology. A typical scenario is that for each individual, all other individuals are different. Even only querying all those different from axioms for each individual using the reasoner in the naive exporter is unfeasible for larger sets of instances. Therefore, for all tests with large Aboxes we only exported explicitly defined disjointness axioms.

5 Related Work

There exists numerous upper ontologies [15] and domain ontologies for nearly every domain. To our surprise methods to efficiently project instance data from private ontologies to corresponding domain- and upper ontologies have not been addressed so far. Related to our research are works for ontology translation such as [6,20]. In contrast to our scenario of projecting private instances to upper ontology instances the goal of ontology translation is to translate between arbitrary ontologies. The approach in [6] follows a similar idea as our approach: First different ontologies are mapped and then merged to one ontology. In a second step the data is projected from a source ontology to a target ontology. However, the setting is different because the translation between arbitrary ontologies requires more expressive mappings (such as translation functions). As a consequence the approach [6] translates the source ontologies to a more expressive language (first order logics), applies specific reasoning methods for translation and lowers the data back to the target ontology. In our setting of the translation to domain/upper ontologies case we can stay on the same (tractable) logical representation as the ontology itself and use any OWL reasoner. Another major difference is that our aim was to achieve minimal results, which, is only partly addressed in [6]. The ontology translation approach in [20] is based on distributed description logics, where different ontologies are mapped with specific bridging axioms. Similar to [6] it allows to reason over a merged ontology but it avoids to use first order logics as an intermediate format. The approach does not care for the miniamlity of the results and requires a specific reasoner. A major difference to the ontology translation approaches is that in our scenario the merged ontology is already given and is based on standard description logics. We are actually interested in extracting specific sets of entailments. Different approaches for the extraction of entailments are discussed in [3]. However, the goal is to construct sets that are interesting for the user for debugging purposes.

Another related field are approaches for modular ontologies [23]. In particular module extraction [16,13,5] is closely related. The idea is to extract a (minimal) sub-ontology for a specific signature from an ontology that preserves the entailments of the input ontology regarding the given signature. Therefore, module extraction solves a more general problem than our problem definition. It was shown in [5] that module extraction is undecidable for OWL DL. Therefore, current implementations use some approximations or impose additional constraints. Since the goal is to create sub-ontologies the generated modules also include additional concepts of the input ontology that describe the input signature but that are not elements of it and the input signature is typically limited to concepts. In contrast we are only interested in Abox axioms that can be expressed using the domain ontology and private concepts should never be part of the export due to privacy requirements.

Finally Abox abduction [18,12] is somehow related to our approach. Reasoning methods for Abox abduction allow to answer queries like: Provide all minimal sets of assertions that must exist for a specific observation to hold. This can potentially also be used for minimizing of Abox axioms. In this case simply all derived or explicit axioms of an individual are observations. However, this is actually a trivial case for Abox abduction. The methods are tailored for non trivial cases and require non standard reasoning facilities that require multiple transformation steps resulting in a limited performance for our scenario. Our solution is based on standard description logics reasoners that are used in the information systems of the partners anyway. The same reasoners can be used to compute the set of instances and to actually perform the export. No additional transformations are required.

6 Conclusion

In this paper we have addressed the problem of exporting instances of a private ontology to instances of a shared upper / domain ontology. Methods to generate such an export either suffer from very high degrees of redundant axioms or of simply unfeasible performance. The major contribution of this paper is a new efficient method for exporting Abox axioms to an upper ontology which generates export sets with low redundancy. The method is based on an systematic investigation of properties of Abox projections that result in redundant axioms. Of course, it would be desirable to guarantee export sets without any redundancy, however, we were more concerned about the performance of the export procedures and were ready to accept low degrees of redundancies. On the one hand the experiments showed that the method we propose here achieves very low degrees of redundancy between 0% and 0.01% in standard benchmarking ontologies. On the other hand the achieved performance is very promising and allows an application in real world use-cases such as the export of detailed provenance information in the biobank domain.

Acknowledgements. We would like to thank the anonymous reviewers for their detailed and valuable comments.

References

1. Aho, A.V., Garey, M.R., Ullman, J.D.: The transitive reduction of a directed graph. SIAM J. Comput. 1(2), 131–137 (1972)
2. Asslaber, M., Abuja, P.M., Stark, K., Eder, J., Gottweis, H., Trauner, M., Samonigg, H., Mischinger, H.-J., Schippinger, W., Berghold, A., et al.: The genome austria tissue bank (gatib). Pathobiology 74(4), 251–258 (2007)
3. Bail, S., Parsia, B., Sattler, U.: Extracting finite sets of entailments from owl ontologies. In: Rosati, R., Rudolph, S., Zakharyaschev, M. (eds.) Description Logics. CEUR Workshop Proceedings, vol. 745. CEUR-WS.org (2011)
4. Buneman, P., Chapman, A., Cheney, J.: Provenance management in curated databases. In: Proc. of the ACM International Conference on Management of Data (SIGMOD), pp. 539–550. ACM Press, New York (2006)
5. Cuenca Grau, B., Horrocks, I., Kazakov, Y., Sattler, U.: Modular reuse of ontologies: Theory and practice. J. Artif. Intell. Res. 31, 273–318 (2008)
6. Dou, D., McDermott, D., Qi, P.: Ontology translation on the semantic web. In: Spaccapietra, S., Bertino, E., Jajodia, S., King, R., McLeod, D., Orlowska, M.E., Strous, L. (eds.) Journal on Data Semantics II. LNCS, vol. 3360, pp. 35–57. Springer, Heidelberg (2005)
7. Eder, J., Dabringer, C., Schicho, M., Stark, K.: Information systems for federated biobanks. T. Large-Scale Data- and Knowledge-Centered Systems 1, 156–190 (2009)

8. Eder, J., Gottweis, H., Zatloukal, K.: IT Solutions for Privacy Protection in Biobanking. Public Health Genomics 15(5), 254–262 (2012)
9. Gupta, A., Condit, C., Qian, X.: Biodb: An ontology-enhanced information system for heterogeneous biological information. Data Knowl. Eng. 69(11), 1084–1102 (2010)
10. Hernandez, T., Kambhampati, S.: Integration of biological sources: current systems and challenges ahead. SIGMOD Rec. 33(3), 51–60 (2004)
11. Kalyanpur, A., Parsia, B., Horridge, M., Sirin, E.: Finding all justifications of OWL DL entailments. In: Aberer, K., et al. (eds.) ISWC/ASWC 2007. LNCS, vol. 4825, pp. 267–280. Springer, Heidelberg (2007)
12. Klarman, S., Endriss, U., Schlobach, S.: Abox abduction in the description logic alc. J. Autom. Reason. 46(1), 43–80 (2011)
13. Konev, B., Lutz, C., Walther, D., Wolter, F.: Semantic modularity and module extraction in description logics. In: Proc. of ECAI 2008, pp. 55–59. IOS Press, Amsterdam (2008)
14. Krötzsch, M., Simancik, F., Horrocks, I.: A description logic primer. CoRR, abs/1201.4089 (2012)
15. Mascardi, V., Cordì, V., Rosso, P.: A comparison of upper ontologies. In: Baldoni, M., Boccalatte, A., Paoli, F.D., Martelli, M., Mascardi, V. (eds.) WOA, pp. 55–64. Seneca Edizioni Torino (2007)
16. Noy, N.F., Musen, M.A.: Specifying ontology views by traversal. In: McIlraith, S.A., Plexousakis, D., van Harmelen, F. (eds.) ISWC 2004. LNCS, vol. 3298, pp. 713–725. Springer, Heidelberg (2004)
17. W3C OWL Working Group. OWL 2 Web Ontology Language: Document Overview. W3C Recommendation (October 27, 2009), `http://www.w3.org/TR/owl2-overview/`
18. Pan, J.Z., Du, J., Qi, G., Shen, Y.-D.: Towards practical abox abduction in large description logic ontologies. Int. J. Semant. Web Inf. Syst. 8(2), 1–33 (2012)
19. Ram, S., Liu, J.: A new perspective on semantics of data provenance. In: Freire, J., Missier, P., Sahoo, S.S. (eds.) SWPM. CEUR Workshop Proceedings, vol. 526. CEUR-WS.org (2009)
20. Serafini, L., Tamilin, A.: Instance migration in heterogeneous ontology environments. In: Aberer, K., et al. (eds.) ISWC/ASWC 2007. LNCS, vol. 4825, pp. 452–465. Springer, Heidelberg (2007)
21. Sheth, A.P., Larson, J.A.: Federated database systems for managing distributed, heterogeneous, and autonomous databases. ACM Comput. Surv. 22(3), 183–236 (1990)
22. Simmhan, Y., Plale, B., Gannon, D.: A survey of data provenance in e-science. SIGMOD Record 34(3), 31–36 (2005)
23. Stuckenschmidt, H., Parent, C., Spaccapietra, S. (eds.): Modular Ontologies. LNCS, vol. 5445. Springer, Heidelberg (2009)
24. Teppo, L., Pukkala, E., Lehtonen, M.: Data quality and quality control of a population-based cancer registry. Experience in Finland. Acta Oncologica 33(4), 365–369 (1994), PMID: 8018367
25. Wichmann, H.-E., Kuhn, K.A., Waldenberger, M., Schmelcher, D., Schuffenhauer, S., Meitinger, T., Wurst, S.H., Lamla, G., Fortier, I., Burton, P.R., et al.: Comprehensive catalog of european biobanks. Nature Biotechnology 29(9), 795–797 (2011)
26. Yuille, M., van Ommen, G.-J.B., Bréchot, C., Cambon-Thomsen, A., Dagher, G., Landegren, U., Litton, J.-E., Pasterk, M., Peltonen, L., Taussig, M., Wichmann, H.-E., Zatloukal, K.: Biobanking for Europe. Briefings in Bioinformatics 9(1), 14–24 (2008)

Using a Reputation Framework to Identify Community Leaders in Ontology Engineering (Short Paper)

Christophe Debruyne and Niels Nijs

Semantics Technology and Applications Research Lab, Vrije Universiteit Brussel
`firstname.lastname@vub.ac.be`

Abstract. Ontology-engineering methods often prescribe roles and responsibilities. A problem is to identify who will play what role in an ontology-engineering project. We present a generic reputation framework for identifying leaders in a community of stakeholders. Reputation scores are based on analyzing the interactions users have with each other and a collaborative ontology-engineering platform. The framework was applied in an experiment involving 36 users that lasted for 8 weeks. We compared the results with the leaders identified by the participants via a survey and noticed an important overlap. Our results thus show the feasibility of identifying leaders in an ontology-engineering project based on reputation, which can then be assigned additional responsibilities.

Keywords: Ontology Engineering, Roles and Responsibilities.

1 Introduction

Ontology engineering is a social process and methods for building ontologies often prescribe roles and responsibilities. We argue that certain roles can be identified by analyzing one's interactions on a collaborative ontology-engineering platform. The role we will focus on is that of a "community leader", the person driving the ontology-engineering project, e.g., by guiding the discussions within a community of stakeholders. To this end, we identify the characteristics of a community leader and propose a reputation framework that assesses a person via sensors monitoring those characteristics.

The paper is organized as follows: Section 2 presents related work and the characteristics of a community leader, Section 3 proposes a reputation framework for a collaborative setting, Section 4 describes the reputation sensors that give scores to particular leader characteristics, Section 5 presents the results of an ontology-engineering experiment in which the framework was applied and Section 6 concludes the paper and presents future work.

2 Background and Related Work

Trust and reputation systems offer a mechanism to *"collect, distribute and aggregate feedbacks [from a community member's history]"* [16]. These feedbacks help others to make a better judgement and encourage trustworthy service or behavior in future interactions [13]. Different ways to use, calculate and represent reputation have been presented in literature for different goals: (i) increase the reliability and trust between agents [17], (ii) contribute to quality identification [10], (iii) improve contribution quality [13,12,10,16,15,3], (iv) filter and recommend content [13,11,10], (v) build or increase co-operation [9,4], (vi) offer improved & real-time business intelligence [5].

Few examples of assigning responsibilities based on trust and reputation exist. One example is Slashdot[1], where moderators are randomly selected to rate contributions based on the ratings from others on their own contributions. Slashdot offers more features to users with a good reputation during a certain period of time. This period is not in function of fixed time, but in function of amount of actions. In [20], influential users are being identified within microblogging services such as Twitter[2] by proposing TwitterRank, an extension of the PageRank[3] algorithm. An approach for identifying experts in a social network by analyzing the agent's local information and their network using graph-traversal was introduced in [22]. A graph representation is also used in [19], where social influence is modeled in their proposed influence propagation method.

In this paper, we aim to identify leaders by means of reputation scores. Leadership is similar to socially influencing group's members to improve collaboration between peers for achieving a goal [18]. Since our goal is to identify community leaders in a collaborative setting, we first need to know what characterizes such leaders. Defining these characteristics allows us to determine what needs to be observed and rated in a reputation framework. The following list of characteristics is based on [7,14,2,1]: (C1) Energy, passionate persistence & optimism, (C2) Goal-Driven, (C3) Build Trust, (C4) Willing to take risks, (C5) Pull and communicate with others, (C6) Work systematically, (C7a) Share knowledge, power and credit, (C7b) Work interdependently, and (C8) Understand others. Items C7a and C7b were considered as one in [1]. We choose to separate both aspects as we aim to measure these aspects separately.

3 Reputation Framework

Based on [13,16,15,6], we define a reputation framework as follows: *"a Reputation Framework is a general independent mediation system to facilitate trust and reputation within social platforms by rating the quality of the users' contribution and use these ratings to calculate a user's reputation score."* The core of a reputation framework is a *reputation computation engine* [13]. This engine will return the reputation score for a user. In this section, we describe such an engine. We define $\mathcal{A}$ as the set of all human agents and $\mathcal{P}$ as the set of all platforms. A platform is an abstract notion for a setting where agents interact with each other. A platform can be an information system (a forum), or a part

[1] `http://www.slashdot.org`
[2] `https://twitter.com/`
[3] `http://en.wikipedia.org/wiki/PageRank`

of an information system (a specific thread in a forum). The set of reputation results $\mathcal{R}$ is defined as $[0;100] \cup \{\oslash\}$, where $\oslash$ will be used when a (sub)result for a user does not exist. A reputation computation engine provides a mapping from a subset of agents $A \in 2^{\mathcal{A}}$ to elements of $\mathcal{R}$ by means of a set of platform configurations C. A platform configuration is a triple $\langle p, w_p, S\rangle$, where $p \in \mathcal{P}$ is a platform, w_p is the platform's weight and S a set sensor configurations. A sensor configuration is a pair $\langle s, w_s\rangle$ where s is a reputation sensor and w_s is the weight for that sensor on that platform. A reputation sensor is a function $s : \mathcal{P} \times \mathcal{A} \to \mathcal{R}$ returning a result for a given user on a given platform. The platform configuration result for user a and for platform configuration $\langle p, w_p, S\rangle$ is described as:

$$\rho_{\langle p,S\rangle}(a) = \begin{cases} \oslash & \text{if } |S'| = 0 \\ \dfrac{\sum_{\langle s,w_s\rangle \in S'} s(p,a) \times w_s}{\sum_{\langle s,w_s\rangle \in S'} w_s} & \text{if } |S'| > 0 \end{cases}$$

where $S' = \{\langle s, w_s\rangle | \langle s, w_s\rangle \in S \wedge s(p,a) \neq \oslash\}$

With the platform configuration result for a user described above, we can now describe the result of a set of platform configurations C for user a as:

$$\rho_C(a) = \begin{cases} \oslash & \text{if } |C'| = 0 \\ \dfrac{\sum_{\langle p,w_p,S\rangle \in C'} \rho_{\langle p,S\rangle}(a) \times w_p}{\sum_{\langle p,w_p,S\rangle \in C'} w_p} & \text{if } |C'| > 0 \end{cases}$$

where $C' = \{\langle p, w_p, S\rangle | \langle p, w_p, S\rangle \in C \wedge \rho_{\langle p,S\rangle}(a) \neq \oslash\}$

A user will be filtered when the value for each platform configuration for that user is $\oslash$, which means that this user had no monitored activity. Platform configuration for which **at most one** user has a value between $[0;100]$ are also not taken into account. In the case of no users with such a value, the platform will not provide any information. In the case of only one user with such a value, we only have one user that is "active" with respect to the monitored activity and thus it would make no sense to find a leader for that platform configuration.

Then, for each platform configuration, we compute the z-scores for each value not equal to $\oslash$. Z-scores give an indication of the distance between the given value and the mean in a number of standard deviations and allow us to compare the z-score of a user across communities. We denote the z-score for platform configuration C_i and user a_j as $\zeta_{C_i}(a_j) = (\rho_{C_i}(a_j) - \mu_{C_i})/\sigma_{C_i}$ when $\rho_{C_i}(a_j) \neq \oslash$, otherwise $\oslash$ remains. According to the empirical rule, about 99.7 percent lie within 3 standard deviations from the mean. For each platform configuration, we rescale these z-scores such that they fit the range of [0;100] using this rule; for a platform configuration C_i and a user a_j, the z-scores are rescaled as follows:

$$\rho'_{C_i}(a_j) = \begin{cases} \oslash & \text{if } \rho_{C_i}(a_j) = \oslash \\ 0 & \text{if } \zeta_{c_i}(a_i) \leq -3\sigma_{C_i} \\ \frac{-50 \times (\zeta_{C_i}(a_j)+3)}{-3} & \text{if } \zeta_{c_i}(a_i) \in (-3\sigma_{C_i}; 0) \\ \frac{50 \times \zeta_{C_i}(a_j)}{3} + 50 & \text{if } \zeta_{c_i}(a_i) \in [0; 3\sigma_{C_i}) \\ 100 & \text{if } \zeta_{c_i}(a_i) \geq 3\sigma_{C_i} \end{cases}$$

Values above 50 indicate a better performance w.r.t. the mean on that platform configuration. For every user a_j, we compute the average of rescaled z-scores where values are not equal to $\oslash$. We then obtain a mapping between users and average rescaled z-scores. These average normalized z-scores provide us a ranking of leaders, a mapping of human agents $A' \subseteq A$ to $[0; 100] \cup \{\oslash\}$ representing the Final User Reputation (FUR), computed for a user a_j as $\Sigma_{c_i \in C'} \rho'_{c_i}(a_j)/|C'|$ where $C' = \{c | c \in C \wedge \rho_c(a_j) \neq \oslash\}$, and returning $\oslash$ when $|C'| = 0$.

4 Reputation Sensors

This section introduces objective and subjective measurements by means of reputation sensors and show how are they related to the leader characteristics listed in Section 2.

Average Activity Rating. This *objective* reputation sensor returns a result for a particular agent, based on his activity within a discussion of the platform. The agent also gets credit for the intention of being active.

Reply Rating. Discussions drive the agreement processes. Even while agents are not obliged to participate, they can be involved by expressing their opinion on statements made by others (e.g., its relevance, constructiveness, etc.). A user can thus be given a score based on how others assess his contribution. As this sensor takes into account the opinion of others, we call this sensor *subjective.* An implementation of this sensor can be based on Slashdot's moderation system, which has been chosen as the system to implement for this paper.

Engagement Rating. This *objective* reputation sensor looks in what extent an agent allows other agents to be engaged within the discussions. In other words, does he allow enough time for others to participate and give an opinion?

Quality Assessment. People collaborate for a particular purpose and the outcome of this collaboration can be evaluated: e.g., to what extent is the artifact usable, correct, complete, etc. Assessing the usability of an artifact can be done via surveys, for instance, and is then a *subjective* sensor. In ontology engineering, one can for instance assess to what extent services are timely and correctly annotated with the evolving ontologies, which can be measured *objectively.* In our work, we will focus on the latter.

Interdependently Rating. We adopt the idea of [21] to build a social graph based on interactions, and apply it to the context of a collaborative setting with the agent's community as social network. Based on a user's participation, the user builds up his social graph with other community-members. This sensor looks at the graph of a user's graph. With this reputation sensor, we can measure the user's *communication characteristic objectively.*

Table 1 depicts how each sensor covers the characteristics of a community leader. There are two characteristics that are difficult to measure with above

Table 1. Leader Characteristics versus Reputation Sensors

	C1	**C2**	**C3**	**C4**	**C5**	**C6**	**C7.a**	**C7.b**	**C8**
Average activity rating	✓	✓			✓	✓			
Reply rating	✓		✓		✓				✓
Engagement rating		✓			✓				
Quality assessment	✓	✓				✓			
Interdependently rating					✓			✓	

mentioned reputation sensors. These characteristics are C4 and C7a, which are both difficult to assess since most take place as interaction outside a system. These aspects could be assessed via surveys, but would not scale well as these attributes can also evolve over time.

5 Experiment

The context of our experiment is a course on ontology engineering, part of the MSc in Computer Science curriculum of the Vrije Universiteit Brussel. In this experiment, 36 participants were divided into 10 groups. Each group had to build their own information system. The resulting conceptual schemas not only contained a view of reality that depended on the information chosen, but also on the perspective of each group. The whole group then had to build ontologies to enable semantic interoperability between their autonomously developed information systems as well as to annotate the database of an existing system. Ontology engineering started on March 10, 2013 and ended on May 10, 2013. Participants were aware that their interactions were taken into consideration to compute their reputation on the framework, but participants were also explicitly told they were not going to be evaluated on their reputation score. The method and tool the students used for ontology engineering is GOSPL[4] [8], in which (i) concepts are both described in terms of natural language definitions called glosses and using a formalism grounded in natural language, (ii) all ontology evolution operators have to be agreed before they are performed, and (iii) glosses drive the negotiation process. The group created multiple communities by separating concerns where each community had the goal to create an ontology for a particular (group of) concepts. Examples are the "Research Project Community" and "Publication Community" that created ontologies for respectively describing research projects and publications. Links between communities can be established by declaring synonyms, which is an agreement that two terms in two communities are deemed to refer to the same concept. In this experiment, we created a platform configuration for each such community. Each platform configuration was given the same weight. All reputation sensors were given weight 1 except for engagement rating, which was given weight 2 since discussing and voting is important for reaching agreements in GOSPL.

[4] Which stands for Grounding Ontologies with Social Processes and natural Language.

As no benchmark is available, we validate the result by comparing the ranking in the reputation framework with a survey sent to the participants. In this survey, we asked the participants up to three names of persons who they deemed to be leading the ontology project. The reputations scores for the participants are shown in Table 2. The column "Rep" represents the Final User Reputation in descending order. Each C_i depicts the platform configuration results for that user. 17 of the 36 participants replied to the survey. Table 3 depicts the number of times a certain participant was chosen as being a leader.

Table 2. Reputation scores of users

User	Rep.	c1	c2	c3	c4	c5	c6	c7	c8	c9	c10	c12	c13	c14	c15	c16	c17	c18	c19	c20	c21	c22	c23
x33	77			70														83					
x34	71		71																		73	69	
x4	63	67			71	64			63										62	50			
x14	62					66					88			32									
x17	62				64	59	39		88	72	74		42	64	75	65	45	66	53				
x22	61			23	70	60		67	76		63	52			66		64	55	72				
x3	59	70			58	58			58										61	50			59
x23	59	62			50				71										61	50			
x8	59			59														58					
x19	56			46														65					
x1	55			72														39					
x31	55			49														61					
x35	55		46																		57	61	
x20	54			58														50					
x30	52				57	72	69		59	32	46			64			37	23	64				
x11	52			32	70	68	63		54	46	64		35	69	52	27	75	46	49			34	50
x6	52		58								38				60								
x36	52			78					43		25				35			78					
x27	51			45														58					
x9	48	65							30														50
x21	48	39				62			41		50												
x10	48			47														48					
x2	47		57			46					58										40	60	23
x7	46	35				57			40		53			45									
x16	46	35				55			35		37			67									
x18	43	56			30	46			30										22				77
x5	43				34	26		35	47		44	52			62		30		60				
x24	41	14			28	17		66	58		61	24			47			40	59				
x29	41	47			43														33				
x12	41			48				32	29		39	71			29			40	40				
x32	40	59				47			44		23			25									41
x37	36		51			22					53										30	26	
x25	35				25	27	29		34		33		73	35	24	59		33	16				
x15	30			24														36					
x26	22																	22					
x13	17		17																				

Table 3. Number of votes per person in survey

Participant	x23	x17	x22	x11	x33	x36	x19	x8	x30	x14	x37	x31	x26
# votes	8	7	6	4	3	3	3	3	2	2	2	2	2

Some users are highly ranked even while they have no recognition by respondents. This can be explained by the fact that not all respondents participated in all communities, and thus their answers are based on the interactions they were involved with. We also note that the respondents did not participate in all communities; hence their answers are based on the interactions they have been involved in. When we then look at the FUR, a weighted average of all the user his rescaled z-scores, we observe that almost all indicated leaders from the survey have a FUR higher then 50. Which means that these users – in general – performed better than average in their respective communities.

6 Conclusions

In this paper, we defined and applied a reputation framework for identifying community leaders in an ontology-engineering project. The characteristics of a community leader were identified, for which different sensors were proposed. These sensors have been implemented as functions in our reputation framework. Our reputation framework was integrated in a collaborative ontology-engineering tool and conducted an experiment with 36 participants. As there is no benchmark available, we compared the results with a survey. We observed an important overlap between users identified as leaders in our framework and the persons regarded as leaders by the participants.

A limitation of the proposed framework is the absence of interactions *outside* the system. Face-to-face communication is difficult to capture and the analysis of – for instance – email communication, while more feasible, might rise privacy issues. One could draw inspiration from solutions such as Nepomuk[5] to collect this data. Future work also includes additional experiments and surveys to calibrate the weights for each sensor, and even for each platform. One can easily imagine ontology projects that are easier (e.g., light weight ontologies for annotation) than others (e.g., constraints needed for reasoning to support certain business processes). As future research we should also investigate whether the complexity of the ontology-project could have an impact on the scores.

Acknowledgements. We thank Prof. dr. Robert Meersman and Sven Van Laere for their valuable comments.

References

1. Archer, D., Cameron, A.: Collaborative Leadership: How to Suceed in an Interconnected World. Routledge (2009)
2. Carter, M.: The importance of collaborative leadership in achieving effective criminal justice outcomes. Center for Effective Public Policy (2006)
3. Casaló, L., Flavián, C., Guinalíu, M.: The role of perceived usability, reputation, satisfaction and consumer familiarity on the website loyalty formation process. Computers in Human Behavior 24(2), 325–345 (2008)
4. Cascella, R., Battiti, R.: Social networking and game theory to foster cooperation Input Paper at the 2nd ENISA Workshop on Authentication Interoperability Languages held at the ENISA/EEMA European eIdentity Conference (2007)

[5] `http://nepomuk.kde.org/`

5. Chang, E., Dillon, T., Hussain, F.: Trust and Reputation for Service-Oriented Environments: Technologies for Building Business Intelligence and Consumer Confidence. John Wiley & Sons (2005)
6. Chen, W., Zeng, Q., Wenyin, L., Hao, T.: A user reputation model for a user-interactive question answering system: Research articles. Concurr. Comput.: Pract. Exper. 19(15), 2091–2103 (2007)
7. Chrislip, D.: The Collaborative Leadership Fieldbook. J-B US non-Franchise Leadership. Wiley (2002)
8. Debruyne, C., Meersman, R.: GOSPL: A method and tool for fact-oriented hybrid ontology engineering. In: Morzy, T., Härder, T., Wrembel, R. (eds.) ADBIS 2012. LNCS, vol. 7503, pp. 153–166. Springer, Heidelberg (2012)
9. Du, P., Li, Y., Xu, W.: Cooperation evolution of indirect reciprocity by discrimination. In: Deng, X., Ye, Y. (eds.) WINE 2005. LNCS, vol. 3828, pp. 768–776. Springer, Heidelberg (2005)
10. Farmer, R., Glass, B.: Building Web Reputation Systems, 1st edn. Yahoo! Press, USA (2010)
11. Golbeck, J., Hendler, J.: Reputation network analysis for email filtering. In: Proceedings of the First Conference on Email and Anti-Spam, vol. 44, pp. 54–58 (2004)
12. Javanmardi, S., Ganjisaffar, Y., Lopes, C., Baldi, P.: User contribution and trust in wikipedia. In: CollaborateCom, pp. 1–6. IEEE (2009)
13. Jøsang, A., Ismail, R., Boyd, C.: A survey of trust and reputation systems for online service provision. Decision Support Systems 43(2), 618–644 (2007)
14. Linden, R.: The discipline of collaboration. Leader to Leader 2003(29), 41–47 (2003)
15. Noorian, Z., Ulieru, M.: The state of the art in trust and reputation systems: A framework for comparison. JTAER 5(2), 97–117 (2010)
16. Resnick, P., Kuwabara, K., Zeckhauser, R., Friedman, E.: Reputation systems. Commun. ACM 43(12), 45–48 (2000)
17. Sabater, J., Sierra, C.: Review on computational trust and reputation models. Artif. Intell. Rev. 24(1), 33–60 (2005)
18. Stogdill, R.: Handbook of leadership: a survey of theory and research. Free Press (1974)
19. Tang, J., Sun, J., Wang, C., Yang, Z.: Social influence analysis in large-scale networks. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 807–816. ACM (2009)
20. Weng, J., Lim, E.P., Jiang, J., He, Q.: Twitterrank: finding topic-sensitive influential twitterers. In: Proceedings of the Third ACM International Conference on Web Search and Data Mining, pp. 261–270. ACM (2010)
21. Wilson, C., Boe, B., Sala, A., Puttaswamy, K.P., Zhao, B.Y.: User interactions in social networks and their implications. In: Proceedings of the 4th ACM European Conference on Computer Systems, EuroSys 2009, pp. 205–218. ACM, New York (2009)
22. Zhang, J., Tang, J., Li, J.: Expert finding in a social network. In: Kotagiri, R., Radha Krishna, P., Mohania, M., Nantajeewarawat, E. (eds.) DASFAA 2007. LNCS, vol. 4443, pp. 1066–1069. Springer, Heidelberg (2007)

Dealing with Context Ambiguity in Context-Based Information Re-finding

Tangjian Deng, Liang Zhao, and Ling Feng

Dept. of Computer Science and Technology, Tsinghua University, Beijing, China
{dtj08,jingzhao-11}@mails.tsinghua.edu.cn, fengling@tsinghua.edu.cn

Abstract. Inspired by human memory that context can often play as important recall cues, access context under which information is previously accessed is being exploited to assist users in information re-finding, *e.g.*, allow users to search their accessed web pages by context. Due to the nature that human memory will decay and could be impacted by interference which will lead to context misremembering, users may refer to unreliable contextual cues in context-based re-finding. In this paper, we focus on how to re-find users' desired results under the circumstance of users' misremembering of precise contextual retrieval cues. To this end, we first categorize three kinds of ambiguity in context-based information recall queries, which are *context degradation, context confusion,* and *context error*, and organize user's access context in associated probabilistic context trees. We then propose an approximate matching approach to deal with users' re-finding requests (formed as contextual keywords) carrying possible ambiguity by taking advantages of context associations, which are extracted guided by human memory's decay and interference characteristics. Experimental results on both synthetic and real data confirm the effectiveness of our solution that can enable users to achieve a good performance in ambiguous context-based information re-finding.

Keywords: Ambiguous context, context association, information re-finding.

1 Introduction

Re-finding previously accessed information is a common behavior in people's daily lives [22,23]. To support web revisitation, a number of web history techniques and tools like bookmarks and history tools are developed [16]. Considering that context under which information was accessed is sometimes more easily to remember than data content itself [3,25], access context like *time, location, concurrent activity, etc.* has also been exploited to enhance information re-finding [11,5,6,4]. "*Re-find the African sweet recipe I saw during a trip to Africa in 1998*" is a context-based information re-finding example. In the literature, [11] developed a *YouPivot* system, which allows users to search through their digital history through the context they remember. The *ReFinder* system [5,6] leverages human's natural recall characteristics and builds a query-by-context model over a context memory snapshot, linking to the previously accessed information.

So far, context-based re-finding efforts in the literature target at precise answers, which are only concerned with the data matching contextual keywords exactly. The ambiguity of human memory about context as re-finding cues, however, is not considered. In fact, human memory exhibits life-cycle degradation nature, and human's recall process may also easily be influenced by various interferences [24]. For instance, newly access events may better be recalled than old ones. The ability to recall a certain piece of information from memory could be impaired by either the newly learned or previously learned information [13,26]. Some nontarget items that are similar to the target ones may compete with the target ones as potential responses in memory recall [20,15]. This may lead to some mismemorized and ambiguous contextual keywords that are used as re-finding requests by users. To illustrate, let's see a real case.

Lily wants to buy a sofa. After comparing many similar styles of sofas for a while, she finally selects the one which she saw at 2013/4/17 when listening to Adele's songs on the Internet, for that one had a good discount. Unfortunately, she cannot re-locate it any more on the web by simple "sofa" keyword, since she forgets any useful information about it, such as detailed title, name of the online shop, etc. What Lily can remember is that she was listening to some music someday in April when coming across that sofa. Finally Lily has to recall that sofa web page through ambiguous contextual keywords (`in-April, listen-to-music`), rather than the exact ones (`2013/4/17, listen-to-Adele-music`).

Due to human memory decay and interference nature, ambiguous contextual keywords may be used in information recall requests. We categorize such context ambiguity into the following three kinds.

– *Context Degradation.* Owing to human memory decay, contextual keywords for searching degrade to a more general level, *e.g.*, instead of giving the exact date `2013/4/17`, the user may only recall via an approximate date `in-April`.
– *Context Confusion.* Due to memory interference and lots of information access events, access context mixes up, *e.g.*, `in-April` is mixed with `in-March`. This happens frequently when the events occur temporally close by. Also, access context may be interfered with each other due to context similarity, *e.g.*, the user may input `listen-to-Ying-music` as the concurrent activity rather than the correct one `listen-to-Adele-music`, since both Ying's and Adele's songs are listened frequently by the user. This may also happen due to the similarity in the accessed information contents themselves.
– *Context Error.* Unlike context confusion, the user may take some non-access-context wrongly as an access context for information re-finding. For example, the use may input `in-April` to query the information which she accessed in March, while no information access happened in April at all.

From the re-finding system perspective, it is not easy to get users' desired results, given such ambiguous context-based re-finding requests, which would often result in plenty of irrelevant results or null results. It would be quite frustrating for users, as unlike information finding, information re-finding is a more directed process, where users have already seen the information before, and can recognize the popped-up target [1]. For the existing context-based re-finding systems, this usually means that the user is forced to try repeatedly with alternative context values until it finally matches the desired data. If the user fails in recalling the access context, then even this solution is infeasible. The aim of this paper is to support users' ambiguous queries for context-based information recall. To the best of our knowledge, this is the first attempt in the literature to deal with ambiguous context-based information re-finding requests. The main contributions of this paper lie in the following three aspects.

– We propose an approximate contextual search solution to deal with users' re-finding requests with ambiguous contextual retrieval cues.
– Guided by human memory decay and interference characteristics, we build three types of context associations between user's contextual retrieval cues: neighborhood associations, similarity associations, and inference associations, to facilitate ambiguous context-based information re-finding.
– We implement an approximate matching algorithm by utilizing the constructed context associations, and conduct experiments on both synthetic and real data to evaluate its effectiveness.

Our experimental results show that approximate matching can achieve a much better recall rate (about 90% and 80% in synthetic and real data experiments respectively) than exact or partial matching (below 60% in both synthetic and real data experiments) under the situation of query requests carrying mistaken information, and it acts similarly to or better than the other two matching methods for precision rate. Whereas, approximate matching takes a bit more time than exact matching.

The remainder of the paper is organized as follows. We review some closely related work in Section 2. We then present a general framework for flexible context-based information re-finding in Section 3. Two important components (*i.e., associated context memory construction* and *ambiguous contextual keyword search*) are addressed in Section 4 and 5, respectively. We evaluate the performance of our solution in Section 6, and conclude the paper in Section 7.

2 Related Work

Context-Based Re-finding. There have been recent studies on context-based information re-finding, incorporating contextual memory cues [3,9]. Dumais *et al.* [8] developed a system *Stuff I've Seen* to facilitate personal information reuse by building index for what a person has seen, and using some cues like file-type, access date, and author for filtering and sorting results. Soules and Ganger [21] presented a file search tool that combines content-based search with contextual information (temporal relationships between files) which is gathered from user activities (system file calls). There are two steps in using this file search tool: first locates files through content-based search and second, extends those results with contextually related files. Chen *et al.* [2] also built a desktop search tool that exploits semantic associations among files, mining from contents such as similar-to relationship and users' such operations as jump-to, copy-from, same-task, and so on. Hailpern *et al.* [11] developed a *YouPivot* system, which keeps and visualizes access context and visited web pages, and allows users to search the context they remember, so that users can see what was going on under that context. Deng *el al.* [5,6] developed a *ReFinder* system to allow users to manually annotate such access context as time, location, and activity for the visited web pages or local files, with which users can pose structured re-finding requests to previously accessed web pages or files. Besides, an approach for web revisitation by context is also presented in [4], which automatically captures user's access context and manages it in a probabilistic context tree for each accessed web page. These existing methods did not consider the context ambiguity in users' re-finding requests and the issue of approximate matching in context-based information re-finding.

Approximate Querying. The techniques of approximate matching in information retrieval and database querying have been explored extensively in the literature [17,18,14,12,7,19]. Commonly, in order to equip with vague retrieval capabilities, these methods employ a distance or similarity function to compare data objects. For example, the predefined distances between values of the same database domain, the edit distance between two strings, and the cosine similarity between two vectors, *etc.*, are frequently adopted. Two data objects are considered to be similar if their similarity score is greater than a predefined threshold value, and the similar objects are considered as the relevant results for the query request. In addition, a priori knowledge about synonyms sometimes is also utilized in finding relevant data. Basically, the approximate matching techniques of these existing work take two aspects into account in performing data comparison: 1) content-based matching, *i.e.*, focus on the data values; and 2) structure-based matching, *i.e.*, focus on the structure in which data is represented, *e.g.*, path, tree, graph, *etc.* Differing from these existing work, our work exploits context associations in approximate matching in order to refrain from the influences of users' misremembering on contextual retrieval cues.

3 The Framework

Fig. 1 outlines the basic idea and framework of our approach to process users' ambiguous context-based re-finding requests. It is comprised of two major components, namely, *associated context memory construction* and *ambiguous contextual keyword search.*

– *Associated context memory construction.* Contextual cues for re-finding are managed in a set of probabilistic context trees (context memory) which link to accessed target items. As shown in Fig. 2, three kinds of user's access context, *i.e.*, access time, access location, and concurrent activity, are considered. Based on human memory interference and misremembering characteristics, we construct three types of context associations between probabilistic context trees' keyword nodes: Neighborhood Associations (NA), Similarity Associations (SA) and Inference Associations (IA). NAs represent that the two associated nodes are neighbors according to their occurring time. SAs are constructed based on the similarity between the two nodes' contextual keywords. IAs refer to associations that one node is inferred by the other.

– *Ambiguous contextual keyword search.* Given a set of associated probabilistic context trees and an ambiguous query request consists of a set of contextual keywords, our idea is to exploit the associations between context trees' keyword nodes and apply them in getting the matched context trees, which include the ones containing all the query keywords (exact matching) and the ones associating with all the query keywords (approximate matching). A contextual keyword is considered to be associated by a context tree if there exists an association between one node of the tree and one node of another tree which exactly contains the keyword. Since the mappings between accessed information and probabilistic context trees have been built, we can easily return the final results to users.

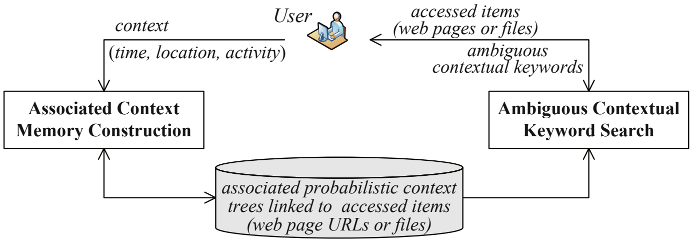


Fig. 1. The framework for ambiguous context-based information re-finding

4 Associated Context Memory Construction

In this section, we present the construction of context associations between probabilistic context trees.

4.1 Probabilistic Context Tree

For an accessed item (web page or file), the access context like time, location and activity is organized in a probabilistic context tree linking to the target item. The details of how to construct probabilistic context trees are addressed in our previous work [4]. Access time is determinate. Access location is obtained based on the IP address of user's computing device or his/her possible GPS information. Concurrent activity is inferred from user's computer applications running before and after the target item access. It is depended on a sliding time window that encompasses the related contexts, which are represented as contextual keywords extracted from focused windows's title. An example of probabilistic context tree is demonstrated in Fig. 2. The leaf nodes attached with contextual keywords are also called keyword nodes, each of which is assigned by a probability from 0 to 1 reflecting the potential that the user will refer to the keywords for re-finding, The probability is set based on 4 factors: 1) the context's focused time length; 2) the context's frequency; 3) the time distance between the context and the access target; and 4)the content similarity between the context and the access target, where the longer the context's focused time length and the more similar the context to the target, the larger the probability between them, while the context's frequency and the time distance between them lead to the opposite case. Note that the hierarchies of time, location and activity at the middle level of the context tree are used to process generalized queries.

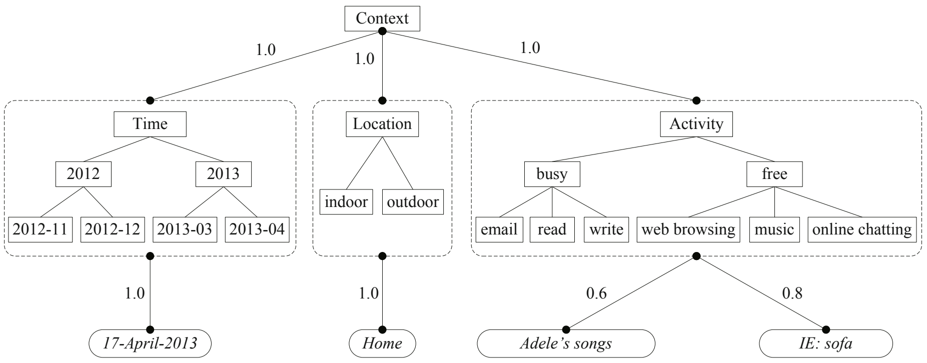


Fig. 2. Example of a probabilistic context tree

4.2 Building Context Associations

As mentioned before, in this work we consider three types of context associations between probabilistic context trees, the details of which are described as follows.

Neighborhood Associations (NA). The related contextual cues of the to-be-revisited data items are organized and formed as a sequence of probabilistic context trees $\langle ct_1, ct_2, \ldots \rangle$. For every $i \geq 1$, ct_i that links to an accessed data item has a start time t_{start} and an end time t_{end}.

Definition 1. *Let ct_i, ct_j be two probabilistic context trees, where ct_i occurs before ct_j ($ct_i(t_{start}) < ct_j(t_{start})$). If $Length(ct_j(t_{start}) - ct_i(t_{end})) \leq \tau_d$ (a predefined time length threshold), then ct_i and ct_j are neighbors to each other.*

Note that if $ct_j(t_{start})$ is before $ct_i(t_{end})$, $Length(ct_j(t_{start}) - ct_i(t_{end}))$ would be negative. Here we set $\tau_d = 30$ min. The contextual information of two neighbor trees may be remembered confusedly by the user. For example, assume ct_i and ct_j are two context trees linking to the data items d_i and d_j respectively. k_i and

k_j are two contextual keywords of two neighbors ct_i and ct_j separately, and the user may remember that d_i is linked by k_j or d_j is linked by k_i.

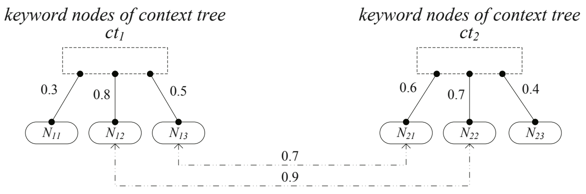

keyword nodes of context tree ct_1
keyword nodes of context tree ct_2
0.3
0.8
0.5
0.6
0.7
0.4
N_{11}
N_{12}
N_{13}
N_{21}
N_{22}
N_{23}
0.7
0.9

Fig. 3. Neighborhood associations between contextual keyword nodes

To overcome such problem, we construct NAs for context trees. Assume two context tree ct_1 and ct_2 are neighbors, where ct_1 has three keyword nodes N_{11}, N_{12} and N_{13}, and ct_2 also has three keyword nodes N_{21}, N_{22} and N_{23}. We do not need to construct NAs for all keyword nodes (otherwise there will be $3 \cdot 3 = 9$ NAs in Fig. 3 in total), since some of which would not be referred to by users in most cases. We should construct NAs for the ones that will be utilized by users with the most potential, *e.g.*, the top two keyword nodes with the biggest probabilities. Moreover, we can divide the four involving keyword nodes into two pairs and build only one NA for each pair, as shown in Fig. 3. Based on NAs, the user can get ct_2 through ct_1's keyword nodes and vice versa. Consider that some keyword nodes may be associated more closely, while some less closely, we define an association strength for each NA.

Definition 2. *Let N_i and N_j be two keyword nodes of two neighbor context trees respectively, $N_i(t_{start})$ and $N_j(t_{start})$ denote the start time of the corresponding context, τ_d is a time length threshold used in Definition 1, the association strength between N_i and N_j is computed as:*

$$NA\_Strength(N_i, N_j) = 1.0 - \frac{|N_i(t_{start}) - N_j(t_{start})|}{2\tau_d} \tag{1}$$

$NA\_Strength(N_i, N_j)$ indicates the closer the two keyword nodes, the larger their association strength. Taking the example of Fig. 3, assume $|N_{12}(t_{start}) - N_{22}(t_{start})| = 6$ min, $|N_{13}(t_{start}) - N_{21}(t_{start})| = 18$ min, then $NA\_Strength(N_{12}, N_{22}) = 1.0 - \frac{6}{2 \cdot 30} = 0.9$, $NA\_Strength(N_{13}, N_{21}) = 1.0 - \frac{18}{2 \cdot 30} = 0.7$.

Similarity Associations (SA). To mimic the phenomenon that the similar items may compete with each other in human memory recall, we construct SAs for the keyword nodes whose contextual keywords are similar to a certain extent. Like NAs, we do not build SAs for each keyword node, but rather choose the ones with the biggest probabilities (*e.g.*, the top two ones) to construct their SAs. We first define a function to measure the similarity of two keyword nodes.

Definition 3. *Let $TermCount(N)$ be a function computing the number of contextual keyword terms of a keyword node N. For N_i and N_j, their similarity, denoted as $C\_Sim(N_i, N_j)$, is computed as:*

$$C\_Sim(N_i, N_j) = \frac{TermCount(N_i \cap N_j)}{TermCount(N_i \cup N_j)} \tag{2}$$

Clearly, $0 \leq C\_Sim(N_i, N_j) \leq 1$. If the keywords of N_i and N_j are completely different, then $C\_Sim(N_i, N_j) = 0$. While at the other extreme, if their contextual keyword terms are exactly the same, then $C\_Sim(N_i, N_j) = 1$.

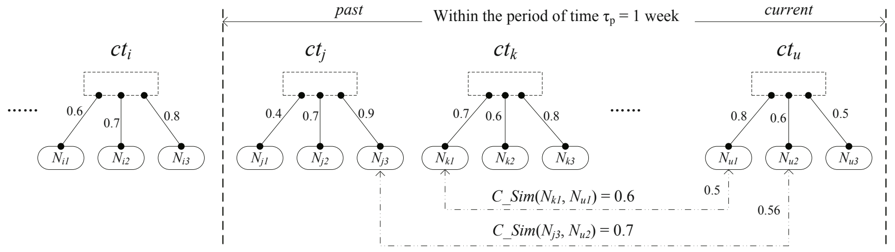


Fig. 4. Similarity associations between contextual keyword nodes

For a newly emerging context tree, it's keyword nodes may be associated by one or more keyword nodes of previous context trees due to their similarities. In this work we focus on the context trees over a period of time (denoted as τ_p) in building SAs for a given keyword node.

Definition 4. *Let N_u and N_j be two keyword nodes of two context trees respectively, satisfying $0 \leq |N_u(t_{start}) - N_j(t_{start})| \leq \tau_p$ and $\theta_{sim} \leq C\_Sim(N_u, N_j) < 1$, the association strength between N_u and N_j is computed as:*

$$SA\_Strength(N_u, N_j) = (1 - \frac{|N_u(t_{start}) - N_j(t_{start})|}{2\tau_p}) \cdot C\_Sim(N_u, N_j) \tag{3}$$

Consider the circumstance that the closer the two similar keyword nodes, the more likely they will be associated, we set τ_p as the past week ($\tau_p = 1$ week), as shown in Fig. 4. For a current keyword node N_u and a previous keyword node N_j within τ_p, if their similarity $\theta_{sim} \leq C\_Sim(N_u, N_j) < 1$ (where θ_{sim} is a predefined a threshold value, here we set $\theta_{sim} = 0.5$), then an SA will be built for N_u and N_j. Note that there is no need to build association between N_u and N_j if $C\_Sim(N_u, N_j) = 1$. Taking the example of Fig. 4, assume $|N_{u2}(t_{start}) - N_{j3}(t_{start})| = 4$ days, $|N_{u1}(t_{start}) - N_{k1}(t_{start})| = 1$ day, $C\_Sim(N_{u2}, N_{j3}) = 0.7$, $C\_Sim(N_{u1}, N_{k1}) = 0.6$, then $SA\_Strength(N_{u2}, N_{j3}) = (1 - \frac{4}{2 \cdot 7}) \cdot 0.7 = 0.5$, $SA\_Strength(N_{u1}, N_{k1}) = (1 - \frac{1}{2 \cdot 7}) \cdot 0.6 = 0.56$.

Inference Associations (IA). Some contextual cues that did not appear actually are inferred to exist owing to users' past experiences. It often occurs in the cases that the involving target items share some common features. Assume there are two target items T_i and T_j linked by context trees ct_i and ct_j respectively, where N_i is one of ct_i's keyword nodes. Because of the similarity of T_i and T_j, N_i is considered as one of ct_j's keyword nodes by the user, *i.e.*, the user would refer to the contextual keywords of N_i in re-finding the target T_j. Towards such circumstances, we particularly construct IAs for the involving context trees. For example, we would build a virtual keyword node N_j with null value for ct_j, and N_i is then linked to N_j by an IA. In this case, we say N_j is inferred by N_i.

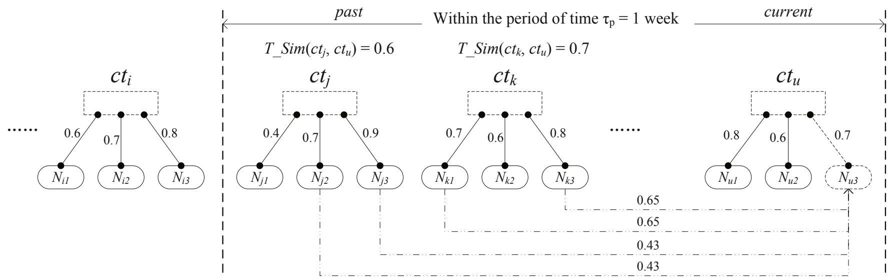


Fig. 5. Inference associations between contextual keyword nodes

For a newly emerging context tree, we first need to identify the context trees to which the one should be linked by IAs. Like SAs, we focus on the context trees over the period of time τ_p in building IAs. For a previous context tree ct_j within τ_p and a current context tree ct_u, we denote $T\_Sim(ct_u, ct_j)$ as the similarity of their linking target items, which can be computed based on Formula 2. If $T\_Sim(ct_u, ct_j) \geq \theta_{sim}$, IAs should be built between ct_u and ct_j. Hence, for the current context tree ct_u, we can obtain a set of context trees $TS = \{ct_j | T\_Sim(ct_u, ct_j) \geq \theta_{sim}$ and $|ct_u(t_{start}) - ct_j(t_{start})| \leq \tau_p\}$. The next step is to build a virtual keyword node N_u for ct_u, and its probability in the context tree denoted as $Pr(N_u)$ is computed as:

$$Pr(N_u) = \max\{T\_Sim(ct_u, ct_j) | ct_j \in TS\} \quad (4)$$

For each element of TS, we choose its two keyword nodes with the biggest probabilities to construct their IAs with the virtual keyword node. We also define an association strength for each IA.

Definition 5. *Let N_u be the virtual keyword node of a context tree ct_u, N_j be a keyword node of a context tree ct_j, the association strength of IA between N_u and N_j is computed as:*

$$IA\_Strength(N_u, N_j) = (1 - \frac{|ct_u(t_{start}) - ct_j(t_{start})|}{2\tau_p}) \cdot T\_Sim(ct_u, ct_j) \quad (5)$$

An example of IAs is demonstrated in Fig. 5, where ct_u is the current context tree, N_{u3} is a virtual keyword node, $|ct_u(t_{start}) - ct_j(t_{start})| = 4$ days, $|ct_u(t_{start}) - ct_k(t_{start})| = 1$ day, $T\_Sim(ct_u, ct_j) = 0.6$, $T\_Sim(ct_u, ct_k) = 0.7$, then $IA\_Strength(N_{u3}, N_{j2}) = IA\_Strength(N_{u3}, N_{j3}) = (1 - \frac{4}{2 \cdot 7}) \cdot 0.6 = 0.43$, and $IA\_Strength(N_{u3}, N_{k1}) = IA\_Strength(N_{u3}, N_{k3}) = (1 - \frac{1}{2 \cdot 7}) \cdot 0.7 = 0.65$.

5 Ambiguous Contextual Keyword Search

A context-based re-finding request $Q = \{k_1, k_2, \ldots, k_n\}$ is a set of contextual keywords, and the query target $\mathcal{C}_T = \{ct_1, ct_2, \ldots\}$ is a set of probabilistic context trees. We first construct index for probabilistic context trees utilizing Dewey encoding [10,27]. In our context trees, the Dewey number of the root is the tree id. For each keyword node in a context tree, we build an index according to its keywords, and the mapping between the node and its probability in the context tree is also kept. Besides, we build an index for each context association. Note that NAs and SAs are bidirectional, while IAs are directional.

5.1 Probabilistic Context Tree Matching

Given Q and $\mathcal{C}_T$, the keyword nodes that contain at least one keyword of Q are called matched keyword nodes, which can be identified through scanning the keyword inverted node lists. Since the tree id can be easily got from the Dewey code of a keyword node, based on the matched keyword nodes, we can easily obtain the exactly matched context trees that encompass every keyword of Q.

For a matched context tree ct, we need to compute its probability of matching Q, denoted as $Prob(ct, Q)$. Assume there are m matched keyword nodes v_1, ..., v_m in ct, each of which contains a set of contextual keywords. The probability of v_i in the context tree is denoted as $Pr(v_i)$, $1 \leq i \leq m$. As there are n keywords in Q, we use a set of real numbers $U(v_i) = \{\mu_{2^n-1}, \mu_{2^n-2}, \ldots, \mu_0\}$ (the maximum size is 2^n) to represent the probabilities of v_i matching Q, *i.e.*, μ_{2^n-1} means the probability that v_i contains all the keywords in Q, and $\mu_{2^{n-i}}$ (where $i = 1, 2, ..., n$) means the probability that v_i only contains k_i, *etc.* Hence, we get m sets of probabilities, which should be merged together in the way that the subscript numbers use bitwise OR operation, while the probabilities use multiplication and add operations. An example of computing the probability of a context tree matching Q is shown in Fig. 6, where $Q = \{k_1, k_2, k_3\}$, the matched keyword nodes are v_1, v_2, v_3 and v_4, and their probabilities in the context tree are $Pr(v_1) = 0.5$, $Pr(v_2) = 0.6$, $Pr(v_3) = 0.7$ and $Pr(v_4) = 0.8$. The four sets of probabilities $U(v_1)$, $U(v_2)$, $U(v_3)$ and $U(v_4)$ are merged together ($\uplus$ denotes the merge operation), and the final μ_{2^n-1} (0.68) is the probability of the context tree matching Q.

5.2 Approximate Search

If the query request Q contains ambiguous information, exact matching will not work well. To eliminate the possible incorrectness of a query request Q with n

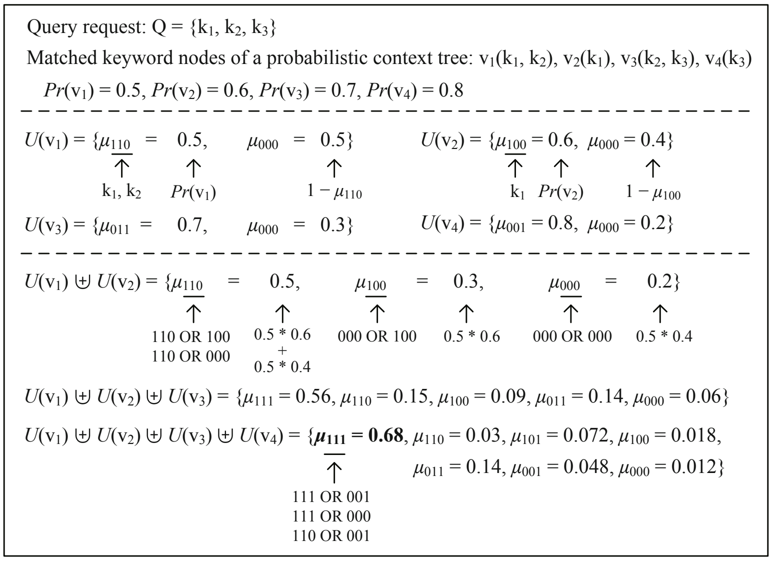

Query request: Q = {k1, k2, k3}
Matched keyword nodes of a probabilistic context tree: v1(k1, k2), v2(k1), v3(k2, k3), v4(k3)
Pr(v1) = 0.5, Pr(v2) = 0.6, Pr(v3) = 0.7, Pr(v4) = 0.8
U(v1) = {μ110 = 0.5, μ000 = 0.5}
k1, k2
Pr(v1)
1 − μ110
U(v2) = {μ100 = 0.6, μ000 = 0.4}
k1
Pr(v2)
1 − μ100
U(v3) = {μ011 = 0.7, μ000 = 0.3}
U(v4) = {μ001 = 0.8, μ000 = 0.2}
U(v1) ⊎ U(v2) = {μ110 = 0.5, μ100 = 0.3, μ000 = 0.2}
110 OR 100
110 OR 000
0.5 * 0.6 + 0.5 * 0.4
000 OR 100
0.5 * 0.6
000 OR 000
0.5 * 0.4
U(v1) ⊎ U(v2) ⊎ U(v3) = {μ111 = 0.56, μ110 = 0.15, μ100 = 0.09, μ011 = 0.14, μ000 = 0.06}
U(v1) ⊎ U(v2) ⊎ U(v3) ⊎ U(v4) = {μ111 = 0.68, μ110 = 0.03, μ101 = 0.072, μ100 = 0.018, μ011 = 0.14, μ001 = 0.048, μ000 = 0.012}
111 OR 001
111 OR 000
110 OR 001

Fig. 6. Example of computing the probability of a context tree matching Q

(≥ 2) keywords, we can adopt partial matching approach [19], *i.e.*, remove each keyword in turn, using the remaining $n-1$ keywords in creating multiple re-finding queries, and retrieve the results of each modified query via exact matching method. Since we do not know which keyword in Q is incorrect, partial matching is a good choice to some degree. However, it will inevitably bring many irrelevant results and meanwhile take more time to execute the queries. Moreover, it is based on the assumption that only one keyword of the query request Q is wrong. Thus, we address approximate matching in the following.

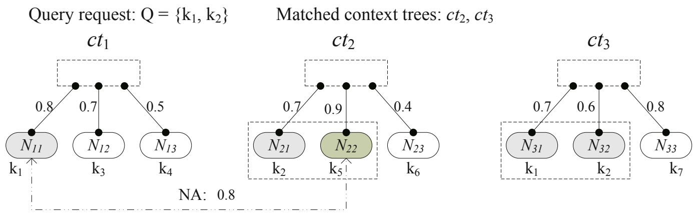


Fig. 7. Example of approximate matching

Generally, users' referring to unreliable contextual retrieval cues is caused by memory interference. For example, to re-find a target item, assume $Q = \{k_1, k_2\}$ is one of the correct query requests. However, the user imposes a wrong query $Q' = \{k_1, k_2'\}$. We consider that there should be some associations between k_2

Algorithm 1. Approximate Contextual Search Algorithm

```
input:
    a query request $Q = \{k_1, ..., k_n\}$ and a set of probabilistic context trees
output:
    a ranked list of context trees $R_T$ that approximately match $Q$
 1: load node list $\mathcal{L} = \{L_i\}$ based on $Q$, $1 \leq i \leq n$;
 2: for $i = 1;\ i \leq n;\ ++i$ do
 3:    $L_{tmp} \leftarrow L_i$;
 4:    for each $v \in L_{tmp}$ do
 5:       load $v$'s associated nodes $A$;
 6:       for each $a \in A$ do
 7:          if $a \notin L_i$ then
 8:             update $Pr(a)$ by multiplying the association strength;
 9:             add $a$ to $L_i$;
10: determine the matched context trees $T = \{ct_1, ct_2, ...\}$ based on $\mathcal{L}$;
11: for each $ct \in T$ do
12:    let $M$ be the set of matched and associated keyword nodes of $ct$;
13:    for each $v_i \in M$ do
14:       compute the probabilities of $v_i$ matching $Q$, $U(v_i) = \{\mu_{2^n-1}, \mu_{2^n-2}, ..., \mu_0\}$;
15:    merge all the $U(v_i)$ together, $U \leftarrow \biguplus_{i=1}^{|M|} U(v_i)$;
16:    $Prob(ct, Q) \leftarrow U.\mu_{2^n-1}$;
17:    if $Prob(ct, Q) > 0$ then
18:       insert $ct$ into $R_T$ according to $Prob(ct, Q)$;
19: return $R_T$;
```

and k_2'. We do not modify the query request Q. We consider the context trees that contain or associate every keyword of Q as matched context trees, where a contextual keyword is deemed to be associated by a context tree if there exists at least an association between one node of the tree and one node of another tree which exactly contains that keyword. Firstly, we will use each keyword of Q to get the matched keyword nodes, based on which the associated keyword nodes can be obtained passingly. Then the matched context trees (exact or approximate matching) can be determined by all the retrieved keyword nodes. Fig. 7 demonstrates an example of approximate matching against a query request $Q = \{k_1, k_2\}$. Four matched keyword nodes N_{11}, N_{21}, N_{31} and N_{32}, and one associated keyword node N_{22} (through the NA with N_{11}), are obtained. Based on these keyword nodes, two matched context trees ct_2 and ct_3 are got. Particularly, when computing the probability of a context tree ct matching Q, the association strengths are multiplied to the involving keyword nodes' probabilities. For example, we multiply $NA\_Strength(N_{11}, N_{12})$ with $Pr(N_{22})$ before computing $Prob(ct_2, Q)$. In the end, $Prob(ct_2, Q) = 0.504$, while $Prob(ct_3, Q) = 0.42$. Thus, ct_2 should be ranked before ct_3 in the result list.

The detailed procedure of approximate matching is illustrated in Algorithm 1. It scans keyword inverted node lists once (Line 1), and add them with the associated nodes (Line 2 - 9). Based on the matched and associated keyword nodes, the approximately matched context trees can be determined (Line 10).

To compute the probability of a matched context tree, it first compute the probabilities of each involving node v_i matching Q, and then merge them together (Line 12 - 15). The matched context tree *ct* will be inserted into the result list at the right position if its probability *w.r.t.* Q is larger than zero (Line 16 - 18).

Complexity Analysis: Adding associated keyword nodes (Line 2 - 9) takes $\mathrm{O}(\sum_{i=1}^{n}|L_i| \cdot |A|)$. Identifying the matched context trees T (Line 10) takes $\mathrm{O}(n \cdot |T|)$. Computing and merging the probabilities of a matched context tree's involving keyword nodes matching Q (Line 12 - 15) takes $\mathrm{O}(\prod_{i=1}^{|M|}|U(v_i)|)$. Thus, the total time cost is $\mathrm{O}(\sum_{i=1}^{n}|L_i| \cdot |A|) + \mathrm{O}(n \cdot |T|) + \mathrm{O}(|T| \cdot \prod_{i=1}^{|M|}|U(v_i)|) = \mathrm{O}(\sum_{i=1}^{n}|L_i| \cdot |A|) + \mathrm{O}(|T| \cdot \prod_{i=1}^{|M|}|U(v_i)|)$, depending on the number of matched and associated keyword nodes, and the number of matched context trees.

6 Experiments

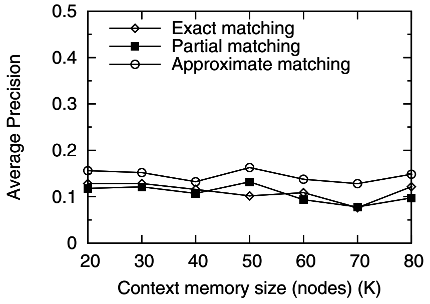
In this section, we conduct a set of experiments to demonstrate the effectiveness of our solution for approximate contextual search in both synthetic and real-world datasets, concentrating on two measurements: 1) query quality (precision and recall) and 2) query response time. The experiments are conducted on a PC with 2.2 GHz Intel Core 2 Duo CPU, and 2 GB memory on Windows 7 OS.

6.1 Experiments on Synthetic Data

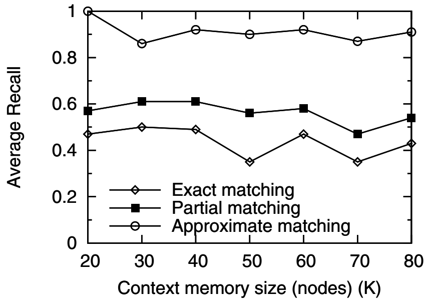
Setting. We generate 3-month data to simulate a user's computer activities. Each data item contains 6 attributes: start time, end time, focus time length, keywords, data type (web page or not) and activity type, which are set randomly, where its time span (end time - start time) is a random value from 30 seconds to 15 min, and its focus time length is from 5 seconds to half of its time span. There are 7 activity types, and we separately generate a set of phrases for each of which, where each phrase contains 3 to 7 words. In generating data items, the keywords are randomly selected from the corresponding set of phrases based on the chosen activity type. In the end, 27,667 data items are generated, based on which we build a probabilistic context tree for each web page type data item whose focus time length is not less than 60 seconds. We get 9,954 context trees with 86,648 keyword nodes in total.

For generating query requests, we first randomly select a part of the web page type data whose focus time length is not less than 60 seconds as the true to-be-revisited targets. For each of them, we then choose 2 to 5 keywords as a query request, complying with one of the following requirements: 1) the keywords are around the target; and 2) the keywords are within the past week taking the target as reference. For the former, we are prone to choose the keywords from the data items whose focus time length is longer than others. For the latter, the similarity and inference associations are taken into account, *e.g.*, the keywords which are similar to the former's candidate keywords will be considered. In total, we separately generate 100 queries for the 2- to 6-keyword query requests.

Results. The first experiment focuses on query precision and recall rates under different sizes of context memory (measured by the total number of keyword nodes in context trees). The performances of contextual keyword search with exact matching, partial matching and approximate matching are studied, and the comparison results are shown in Fig. 8. Because of the low frequencies of contextual keywords in the generated synthetic dataset, the number of returned results for a query request will not change observably along with the varying of context memory size, *i.e.*, the context memory size does not impact the average precision and recall rates very much. Since the query requests are with some flaws, the approximate matching outperforms the exact and partial matching in both average precision and recall rates.



(a) Average precision

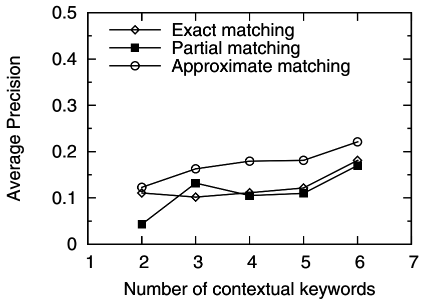


(b) Average recall

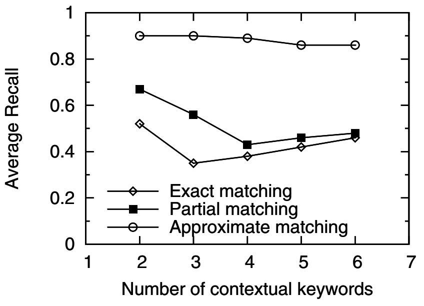
Fig. 8. Varying context memory size (nodes) on synthetic data, $|Q| = 3$

The second experiment also concerns on query precision and recall rates through varying the number of contextual keywords in a query request. Fig. 9 demonstrates the comparison results between approximate matching and exact/partial matching. Clearly, the approximate matching can achieve much better results than the other two matching methods. As with more contextual keywords, more query conditions are imposed, the precision rate get improved along with the increase of the number of contextual keywords from 2 to 6. Whereas, more contextual keywords will lead to a higher potential of carrying mistaken information, making the recall rates of exact and partial matching decrease. While the approximate matching approach is not impacted, the recall rate of which keeps at high level, over 85%.

The query response time under different sizes of context memory and different number of contextual keywords in a query request is also studied, as illustrated in Fig. 10. The three matching methods take more time to do re-finding along with the growth of context memory size. However, increasing the number of contextual keywords hardly affects the response time of exact matching, while the response time of the other two methods grows along with the keyword number. This is contributed to the factor that the exact matching method always chooses the



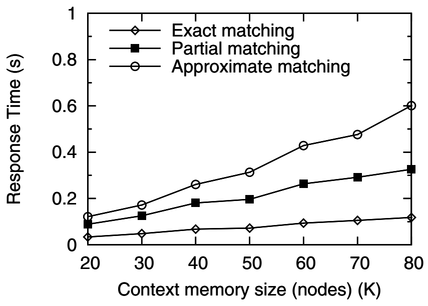
(a) Average precision



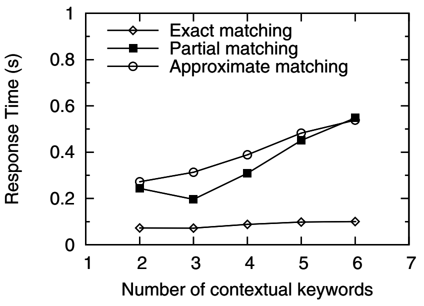
(b) Average recall

Fig. 9. Varying $|Q|$ on synthetic data, the number of nodes = 50K

shortest node list ($|Q|$ lists in total) to start determining the matched context trees and the increase of $|Q|$ just incurs a bit more time, while the number of matched context trees decreases, which requires less time to compute their probabilities. Of the three methods, exact matching takes the least response time. This is due to the reason that both approximate and partial matching have to deal with much more matched (associated) keyword nodes for a query request than exact matching.



(a) $|Q| = 3$

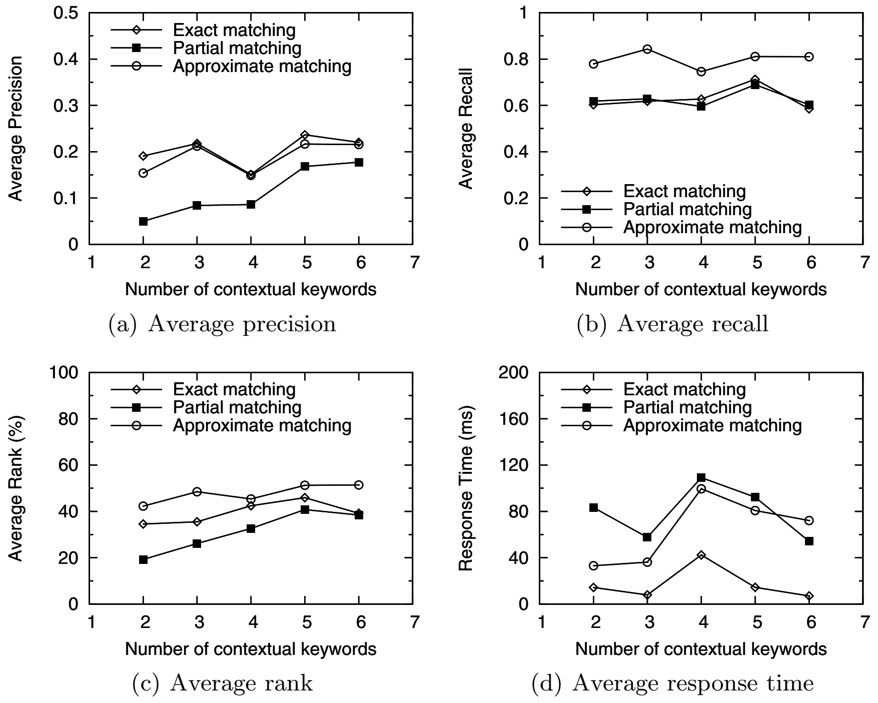


(b) number of nodes = 50K

Fig. 10. Query response time for varying context memory size (nodes) and $|Q|$ on synthetic data

6.2 Experiments on Real Data

Setting. To examine the performance of our solution on real data, we collected computer activity data of 8 participants (graduate students) over 6 weeks. The participants' running computer applications were continuously monitored, and related information of focused windows like window title, application type, the start and end time of being focused, *etc.*, were kept automatically. The browsed



(a) Average precision (b) Average recall (c) Average rank (d) Average response time

Fig. 11. Performance on real data for varying $|Q|$

web pages were considered as to-be-revisited targets, and we build a probabilistic context tree for each of them. The context tree's keyword nodes are inferred from participant's computer applications running before and after the web page access. In total, we collected 1,530 context trees (linking to browsed web pages) with 28,256 keyword nodes. For each participant's data, we separately select 20 queries of 2- to 6-contextual keyword as query requests.

Results. We study query quality (including precision, recall and rank) and query response time of our solution on the 8 collected real datasets separately. The average results are demonstrated in Fig. 11. For precision rate, both exact and approximate matching methods act much better than partial matching method. It is because that the latter retrieves more irrelevant results. For recall rate, approximate matching outperforms the other two methods clearly, since approximate matching is able to be immune to the impacts of query requests containing mistaken information in most cases. To measure results' ranking, we define $rank = true\_result\_ranking\_position/number\_of\_returned\_results \cdot 100\%$. Considering results' ranking positions are determined by their probabilities of matching Q, and approximate matched results involving association strengths could not get good rankings in many cases, both exact and partial matching perform better than approximate matching for average rank. For response time, exact matching still outperforms the other two methods.

7 Conclusion

In this work, we propose an approximate matching approach for contextual keyword-based information re-finding. It is aiming at eliminating the negative influences of users' misremembering on contextual retrieval cues which are organized in probabilistic context trees. Based upon the observation of human memory interference and the characteristics of users' misremembering in memory recall, we construct three types of context associations between probabilistic context trees: neighborhood associations, similarity associations and inference associations. The approximate matching algorithm takes advantages of such context associations to deal with users' unreliable re-finding requests, and the matched or associated results will be returned. The proposed solution is evaluated on both synthetic and real-world datasets. Under the situation of some contextual keywords of users' query requests containing mistaken information, our experimental results show that approximate matching performs much better than exact and partial matching in re-finding users' desired results.

Acknowledgement. The work is supported by National Natural Science Foundation of China (60773156, 61073004), Chinese Major State Basic Research Development 973 Program (2011C B302203-2), Tsinghua University Initiative Scientific Research Program, and Tsinghua-Tencent Joint Laboratory for Internet Innovation Technology.

References

1. Capra, R., Perez-Quinones, M.A.: Using web search engines to find and refind information. IEEE Computer 38(10), 36–42 (2005)
2. Chen, J., Guo, H., Wu, W., Wang, W.: iMecho: an associative memory based desktop search system. In: CIKM, pp. 731–740 (2009)
3. Chen, Y., Jones, G.: Integrating memory context into personal information refinding. In: The 2nd Symposium on Future Directions in Info. Access (2008)
4. Deng, T., Zhao, L., Feng, L.: Enhancing web revisitation by contextual keywords. In: Daniel, F., Dolog, P., Li, Q. (eds.) ICWE 2013. LNCS, vol. 7977, pp. 323–337. Springer, Heidelberg (2013)
5. Deng, T., Zhao, L., Feng, L., Xue, W.: Information re-finding by context: a brain memory inspired approach. In: CIKM, pp. 1553–1558 (2011)
6. Deng, T., Zhao, L., Wang, H., Liu, Q., Feng, L.: ReFinder: a context-based information re-finding system. IEEE TKDE (PrePrint, August 14, 2012)
7. Dorneles, C.F., Goncalves, R., dos Santos Mello, R.: Approximate data instance matching: a survey. Knowledge and Information Systems 27(1), 1–21 (2011)
8. Dumais, S., Cutrell, E., Cadiz, J., Jancke, G., Sarin, R., Robbins, D.C.: Stuff i've seen: a system for personal information retrieval and re-use. In: SIGIR, pp. 72–79 (2003)
9. Fuller, M., Kelly, L., Jones, G.: Applying contextual memory cues for retrieval from personal information archives. In: PIM, Workshop at CHI (2008)
10. Guo, L., Shao, F., Botev, C., Shanmugasundaram, J.: Xrank: ranked keyword search over xml documents. In: SIGMOD, pp. 16–27 (2003)

11. Hailpern, J., Jitkoff, N., Warr, A., Karahalios, K., Sesek, R., Shkrob, N.: Youpivot: improving recall with contextual search. In: CHI, pp. 1521–1530 (2011)
12. Ji, S., Li, G., Li, C., Feng, J.: Efficient interactive fuzzy keyword search. In: WWW, pp. 371–380 (2009)
13. Jonides, J., Nee, D.E.: Brain mechanisms of proactive interference in working memory. Neuroscience 139(1), 181–193 (2006)
14. Kailing, K., Kriegel, H.-P., Schönauer, S., Seidl, T.: Efficient similarity search for hierarchical data in large databases. In: Bertino, E., Christodoulakis, S., Plexousakis, D., Christophides, V., Koubarakis, M., Böhm, K. (eds.) EDBT 2004. LNCS, vol. 2992, pp. 676–693. Springer, Heidelberg (2004)
15. Lustig, C., Hasher, L.: Implicit memory is not immune to interference. Psychological Bulletin 127(5), 629–650 (2001)
16. Mayer, M.: Web history tools and revisitation support: a survey of existing approaches and directions. Foundations and Trends in HCI 2(3), 173–278 (2009)
17. Motro, A.: Vague: a user interface to relational databases that permits vague queries. ACM TOIS 6(3), 187–214 (1988)
18. Navarro, G.: A guided tour of approximate string matching. ACM Comput. Surv. 33(9), 31–88 (2001)
19. Qumsiyeh, R., Pera, M.S., Ng, Y.: Generating exact and ranked partially matched answers to questions in advertisements. In: VLDB, pp. 217–228 (2012)
20. Robinson, J.A., Swanson, K.L.: Autobiographical memory: The next phase. Applied Cognitive Psychology 4(4), 321–335 (1990)
21. Soules, C.A.N., Ganger, G.R.: Connections: using context to enhance file search. In: SOSP, pp. 119–132 (2005)
22. Tauscher, L., Greenberg, S.: How people revisit web pages: empirical findings and implications for the design of history systems. Int'l J. of Human Computer Studies 47, 97–137 (1997)
23. Teevan, J., Adar, E., Jones, R., Potts, M.A.S.: Information re-retrieval: repeat queries in yahoo's logs. In: SIGIR, pp. 151–158 (2007)
24. Tomlinson, T.D., Huber, D.E., Rieth, C.A., Davelaar, E.J.: An interference account of cue-independent forgetting in the no-think paradigm. Proceedings of the National Academy of Sciences 106(37), 15588–15593 (2009)
25. Tulving, E.: What is episodic memory? Current Directions in Psychological Science 2(3), 67–70 (1993)
26. Wohldmann, E., Healy, A., Bourne, L.: A mental practice superiority effect: Less retroactive interference and more transfer than physical practice. Journal of Experimental Psychology: Learning, Memory, and Cognition 34(4), 823–833 (2008)
27. Xu, Y., Papakonstantinou, Y.: Efficient keyword search for smallest lcas in xml databases. In: SIGMOD, pp. 527–538 (2005)

Cognitive Modeling for Topic Expansion (Short Paper)

Sumant Kulkarni, Srinath Srinivasa, and Rajeev Arora

International Institute of Information Technology Bangalore,
26/C, Electronics City, Bangalore India
{sumant,sri}@iiitb.ac.in, rajeev.arora@iiitb.org
http://www.iiitb.ac.in

Abstract. Expanding a topic into its constituent terms used by a population is an important problem in social space analytics. A topic is defined as a concept representing the semantic "aboutness" of a set of other concepts. What a topic is really "about" may differ across different populations, and discovering this provides sound insights about the population itself. In this paper, we propose an approach for topic expansion, inspired by models from cognitive science – specifically, episodic and semantic memory. We propose an *episodic hypothesis* that asserts a relationship between a topic and other concepts that are collectively about the topic. The canonical form of this algorithm, while shown to produce good results, does not run in interactive response time. We then propose several simplifications over the canonical form to provide interactive response times without reducing too much on the quality of the results. The proposed simplifications also help in creating topical clusters on repositories where there is not enough representation of different terms for the topic.

Keywords: Topic Expansion, Topic Modeling, Text Mining, Social Space Analytics.

1 Introduction

Online social spaces are becoming an integral part of our lives. The need to understand different viewpoints expressed in these spaces has generated interest in social space analytics.

In this paper, we look into the problem of *topic expansion.* The task here is to determine the collective world-view of a population on a given topic. For instance, on Wikipedia we found that the term *Europe* was largely associated with terms pertaining to history and socio-politics, while on an internal Indian corporate blog dataset, *Europe* was associated with terms pertaining to photography and tourism. These collective world-views for a given topic can be of interest to several stakeholders in policy making, strategic planning, marketing and beyond.

This paper uses the 3-layer cognitive model [7,8] for topic expansion. The model is based on proposing an *episodic hypothesis* that makes an assertion about term distribution across episodes to different kinds of semantic associations.

This paper proposes and tests an episodic hypothesis based on the property of "topical coherence" of an episode. This hypothesis is then tested with human subjects using term distributions from Wikipedia and compared with other approaches for topic modeling.

The proposed hypothesis provided satisfactory results but performance was a major concern. We then experimented with different relaxations to the hypothesis to improve performance without significant changes to the quality of results. Both algorithms are discussed in this paper and a prototype is available over the Internet[1]. A more detailed version of this paper can be found as a technical report [1].

2 Related Literature

The problem of topic expansion is similar to Word Sense Disambiguation (WSD) and Topic Modeling (TM). Automatic solutions for WSD can be classified in two broad categories: Supervised and Unsupervised; both using advancements in machine learning. Supervised WSD approaches [6] use training corpus manually annotated with word senses from a dictionary. Unsupervised approaches, based on the idea of distributional hypothesis [9], do not use them. WSD does not take care of ordering of the terms based on their importance generally.

A Topic modeling algorithm like LDA [2], can be seen to be closely related to topic expansion. Graber et al. [3] extended LDA to use WordNet random walk as an additional source of word generation. Budanitsky et. al [4] found that it is difficult to quantify adhoc semantic senses.

Rachakonda et al. [7,8] use models from cognitive science to develop a 3-layer model for inferring latent semantics. The work presented in this paper is built on top of the 3-layer cognitive model and applied to the problem of topic expansion. Section 3 describes the 3-layer cognitive model in detail. A more detailed and extensive literature survey can be found in [1].

3 3-Layer Cognitive Model

In this section, we briefly introduce the 3-layer cognitive model first proposed in [8], which forms the underlying framework for the topic expansion algorithm. Based on relevant theories in cognitive science, semantic communication in humans is modeled as comprising of three layers called the *analytic* layer, the *episodic* layer and the *linguistic* layer, respectively. Figure 1 depicts this.

The analytic layer is responsible for maintaining our semantic "worldview" in the form of concepts and relationships across them. An axiomatic unidirectional relationship called "aboutness" is defined between a pair of concepts c_1 and c_2 such that the aboutness of c_2 from c_1 is the measure of how much c_2 is about (is relevant to) c_1. The semantics used in the analytic layer is hosted in our

[1] Working prototype of topic expansion can be found at `http://tinyurl.com/topicexpansion`

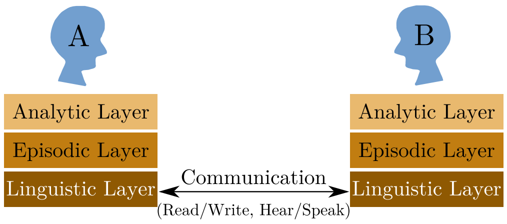


Fig. 1. Semantic communication in humans

semantic memory. Concepts in semantic memory are built from extracting and generalizing on experiences from the *episodic memory.* An episode corresponds to an autobiographical situation from where experiences are recorded.

Our approach towards mining latent semantics focuses on the interplay between semantic and episodic memory. Given a text corpus, the computer acts as the recipient of the semantic communication, and each document in the corpus can be seen as an episode. To generalize on episodic knowledge, we propose an interim data structure called the *co-occurrence graph* that maintains a weighted set of co-occurrences of terms across the corpus. The *co-occurrence graph* represents a primitive and "uncooked" form of analytic layer for machines. Finer semantic constructs are derived by using several algorithms over it. The basis for such constructs are based on proposing relevant *episodic hypotheses* that hypothesize on the patterns of co-occurrences of terms and their relationship with underlying semantics.

An episode is divided into several *occurrence contexts*, and pairs of terms occurring in the same occurrence context are added as edges to the co-occurrence graph. If two terms have co-occurred at least once in the corpus, then there will be an edge between them. We formally define the undirected co-occurrence graph $\mathcal{G}$ as, $\mathcal{G} = (T, C, w)$, where, T is the set of all terms in the given corpus. C is the set of all pair-wise co-occurrence between terms in T. The function $w : C \to \mathbb{N}$ is the corresponding pair-wise co-occurrence count. On top of a given co-occurrence graph, several "primitives" are defined. A pertinent set of primitives used in co-occurrence algorithms of topic expansion are introduced below.

Given a set of terms X, their *focus* $X_\perp$, is the set of terms that co-occur with *all* terms in X. For a term t, its *neighborhood* $N(t)$ is the set of all terms which have co-occurred with it in the corpus along with their co-occurrence weights. We formally define neighborhood as: $N(t) = (T_{N(t)}, C_{N(t)}, w)$, where $T_{N(t)} = \{t\} \cup \{u | u \in T, \{t, u\} \in C\}$, $C_{N(t)} = \{\{t, u\} | u \in T, \{t, u\} \in C\}$ and w is the corresponding edge weight in $\mathcal{G}$ for a given edge.

Given a set of terms X, the neighborhood of its focus $N(X_\perp)$ is defined as: $N(X_\perp) = \bigcap_{x \in X} N(x)$. The set of terms in X are treated as a hypothetical single concept represented by multiple terms when we are talking about neighborhood of $X_\perp$. This allows us to see this neighborhood once again as a star graph. Co-occurrence weights of a given neighbor u from $X_\perp$ is defined as the minimum of all edge weights to u from all the terms in $X_\perp$, corresponding to the multi-set (bag) intersection of co-occurrences. Formally: $w(X_\perp, u) = \min\limits_{x \in X} w(x, u)$.

While the co-occurrence count represents joint occurrences of pairs of terms, from the vantage point of a given term u, we can compute the probability of the occurrence of other terms in the same context as u. This results in a directed graph depicting the "generatability" of a given term u in the context of some other term t. Formally, then the generatability of this co-occurrence $\Gamma_{t \to u}$ is,

$$\Gamma_{t \to u} = \begin{cases} \dfrac{w(t,u)}{\sum_{x \in T_{N(t)}} w(t,x)} & u \neq t, u \in T_{N(t)} \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

4 Canonical Topic Expansion (CTE)

Topic expansion is a process of unfolding a topic t represented by a given concept, into related concepts, that are collectively *about* t. Terms are linguistic handles for concepts. Each term may represent multiple senses and each sense represents a concept in analytical layer. Hence, sense distinction becomes an implicit part of the topic expansion problem.

We approach topic expansion using the 3-layer cognitive model. First we define an *analytical definition* of topic expansion. Then, we propose an *episodic hypothesis* that asserts observable patterns that indicate topic expansion. Finally, we design a *co-occurrence algorithm* based on the episodic hypothesis that is used for running topic expansion on the given text corpus.

Analytical Definition. For a topic represented by a concept t, a topic expansion $TE(t) = \{c_1, c_2, ..., c_n\}$ is a set of concepts, which collectively display a high *aboutness* for t in our semantic memory. It also makes sense to view topic expansion as an *ordered list* of concepts, based on their individual *aboutness* scores for t. This makes $TE(t)$ as a tuple of terms: $\langle c_1, c_2, ..., c_n \rangle$, where, $A_{c_1}(t) \geq A_{c_2}(t) \geq \ ... \ \geq A_{c_n}(t)$. $A_{c_i}(t)$ is the aboutness score of c_i for t. A concept is always maximally "about" itself, making t the first concept in $TE(t)$.

Episodic Hypothesis *(Topical coherence hypothesis.)* In a long enough conversation (episode) *about* a concept t, terms that are *about* t including t itself tend to be uttered together, compared to terms that are not *about* t.

Co-occurrence Algorithm: The episodic hypothesis is converted into an algorithm over the co-occurrence graph through *cluster generation*, *cluster merging* and *ranking* steps.

Cluster generation: Given a term t with $|N(t)| = k$, we first look for clusters of terms in $N(t)$ containing t. Clustering is based on the generatability of the terms in neighborhood. For k neighbors, this step generates k clusters. The i^{th} run of the clustering algorithm will be based on the topic term t and i^{th} most generatable term. Algorithm 1 explains the i^{th} iteration of the step.

Cluster merging: This step merges redundant clusters based on their similarity calculated as, $O(C_a, C_b) = \frac{|C_a \cap C_b|}{\min(|C_a|,|C_b|)}$ for clusters C_a and C_b. Pairs of clusters having high similarity are merged iteratively till there are no highly similar clusters.

Algorithm 1. Cluster generation with the ith most generatable term

```
Data: Co-occurrence graph $G$, topic term $t$, co-occurring term $u_i$ where $u_i$ is
      the $i^{th}$ most generatable term in the neighborhood of $t$, when this
      algorithm is called for the $i$th time. $u_i$ is the "sense" of the cluster X.
Result: Cluster $C_i$ of terms containing $t$, $u_i$ and a subset of terms from $N(t)$
$X \longleftarrow \{t, u_i\}$ ;
while $N(X_\perp)$ exists do
    Let $v \in N(X_\perp)$ be the term in $N(X_\perp)$ with the highest generatability
    $\Gamma_{X_\perp \to v}$ ;
    $X \longleftarrow X \cup v$ ;
end
return $X$;
```

Ranking: This step orders the terms within each cluster in the order of their importance to the sense of t. This is done using a measure called the *exclusivity* score of a term to t. Exclusivity is the bidirectional generatability score between two terms and is calculated as $E(t_m, t_n) = \Gamma_{t_m \to t_n} \cdot \Gamma_{t_n \to t_m}$. The exclusivity score explains the importance of the term of topic (t) to the terms in the cluster and vice-versa. Detailed explanation on all these steps can be found in [1].

In this algorithm, the *cluster generation step* initially expands the topic for *all* the senses available, even though it might generate redundant clusters. We call this the Canonical Topic Expansion (CTE) algorithm.

4.1 Experimental Evaluation of CTE

Experimental evaluation for the quality of results were performed over a co-occurrence graph constructed over a Wikipedia dump of the year 2006. 25 polysemic terms were chosen as topic terms to evaluate the results. Results of CTE were compared with outcomes of (a) a topic modeling approach based on LDA, and (b) an algorithm for WSD [5]. We generated 3 clusters for each algorithm and asked users to rate the clusters over two factors: *cohesiveness*(C_{val}) and *relatedness*(R_{val}). The C_{val} of a cluster is a measure of how strongly interrelated were the terms in the cluster. The R_{val} is a measure of the relevance of the cluster to the topic term t. It was observed that CTE outperforms the results of both LDA and WSD. The details of the evaluations are found in [1].

The worst case time complexity of CTE is $O(N^5 \log N)$ [1] for corpus with N terms. For *cluster generation* step it is $O(N^3)$. Though it is polynomial, it is still unacceptable for very large N. Co-occurrence graphs tend to be extremely dense (diameter of Wikipedia co-occurrence graph is 4), making average running time close to the worst case.

Another shortcoming of CTE is that, on corpora where document lengths are small, the evidence of terms co-occurring with the *focus* of a large set of candidate terms is unlikely. This form of a co-occurrence requires the existence of a clique in the co-occurrence graph, which is unlikely when document sizes are small. To address these problems – of broadening topic expansion to work on corpora with small document sizes, and make it interactive in response times, we made some simplifications on CTE which are explained in the next section.

5 Interactive Topic Expansion (ITE)

The episodic hypothesis of CTE asserts that the concepts which are "about" a topic tend to cluster together around the topic. This clustering was represented as *cliques* in the co-occurrence algorithm. However, clusters need not always be cliques, and this is more likely in cases where articles in the corpus are not very long. Relaxing the interpretation of a cluster as a clique can broaden the result size of topic expansion as well as reduce the response time significantly.

5.1 Co-occurrence Algorithm for ITE

In CTE, the co-occurrence algorithm assumes that every time a new term is added into a cluster, it is due to its high generatability with *all* other terms already present in the cluster. This assumption is relaxed to give us a set of ITE algorithms. In ITE, initial clusters are formed based on only few *common terms* and then all terms in the focus of the common terms are added to the result set. Different variants of this are proposed, called *Super Fast* (*SF*) and $Fast_k$ (F_k). ITE algorithms comprise of the same three steps: *Cluster generation, Filtration* and *Ranking*. *Cluster generation* and *Filtration* steps are interleaved.

The *Cluster generation* in F_k methods is performed by first adding $k+1$ more terms to the cluster based on their generatability scores as it was in CTE and then adding all the terms in the focus of the topic term and the other k terms. The term added to the cluster after the topic term is called "key sense". The *k+1* terms are the *anchor terms*. The F_0 is called *SF*.

The generated cluster is given to the *Filtration* step. This step drops the generated cluster if it has overlap higher than the "drop" threshold β with any already generated valid clusters. *Cluster generation* and *Filtration* are performed until all the neighbors are accounted. The clusters surviving filtration are given to *Ranking* step which is same as of CTE. It gives the final ITE clusters.

The *Cluster generation* step in both methods executes faster than in CTE as the worst case complexity of it reduces to $O(N^2)$. The other steps have the same worst case time complexities as of CTE. However, as the worst case is highly improbable in ITE, we claim that ITE performs better than CTE. Following experimental evaluation confirm this. The asymptotic analysis of both CTE and ITE can be found in [1].

5.2 Experimental Results

The experimental evaluation setup for ITE was the same as detailed in section 5.1. The drop threshold (β) was set to 0.5. The performance of ITE was evaluated on four grounds namely: the result similarity with CTE, execution time, C_{val} and R_{val}. The same 25 topic used to evaluate CTE were chosen.

Evaluation of Similarity of Results with CTE: We compared the results generated by both the ITE methods with that of CTE based on the key senses.

For every cluster generated by ITE, we checked whether there is a cluster generated using the same *key sense* term in CTE. If there was such a cluster, we calculated the Jaccard coefficient and overlap of the two clusters. The Jaccard coefficient between two sets is, $J(A, B) = \frac{|A \cap B|}{|A \cup B|}$. If no corresponding cluster was found in CTE, the ITE score was 0. We calculated the average of these overlap and Jaccard for 25 terms to get average *Jaccard score* and *Overlap score.* These scores were calculated for 25 iterations starting with *SF* till F_{24}. We also calculated the *speedup* of the ITE compared to CTE for each of the iterations. Speedup signifies the execution time reduction achieved by the ITE compared to CTE. The speedup $S(k)$ for k^{th} iteration with k anchor terms is calculated as $S(k) = \frac{E_c(n) - E_i(n,k)}{E_c(n)}$, where, for n terms, $E_c(n)$ is the execution time of the CTE and $E_i(n, k)$ is the execution time of the ITE in the k^{th} iteration.

We observed that the Jaccard score started with 0.219 and increased with the increase in the number of anchor terms initially. It stabilized at 0.395 after the iteration 19. As the number of anchor terms increased in ITE, the similarity between the ITE and CTE clusters also increased. However, the stabilization of the Jaccard at 0.395 showed that, the ITE can not produce results identical to CTE and has a upper bound in the similarity it can achieve.

The *Overlap Scores* showed a decreasing tendency initially starting from 0.53 however, stabilizing after 19^{th} iteration to 0.47. This shows that initially there were many large ITE clusters for which the corresponding CTE clusters were their *near subsets.* However, as the number of anchor terms increased, the ITE clusters reduced in their size reducing the tendency of CTE clusters being their *near subset.* We also observed that there was a high *speedup* value for all the 25 terms, even though there was a decreasing trend. The *S(25)* still approached 1 (*S(25)* = 0.98) showing the very low execution time of ITE compared to CTE.

User Evaluation. We evaluated with the same 25 terms we used for CTE. Similar to CTE evaluation, we calculated the C_{val} and R_{val} for all three methods: CTE, F_1 and *SF.* We observed that F_1 gave marginally better R_{val} of 3.29, than CTE with a R_{val} of 3.25. *SF* with a R_{val} 3.10 performed worse compared to CTE. We also calculated the average of C_{val} of all terms for CTE, F_1 and *SF* methods. We observed similar trends here as well. F_1 with a C_{val} of 3.20 was the best performer. The CTE with C_{val} 3.19 performed better than *SF* with 3.03.

We observed that the F_1 method performed marginally better than CTE in the user evaluation in both R_{val} and C_{val}. Similarly, *SF* method marginally performed worse compared to CTE. However, there was larger reduction in the execution time (higher *speedup*) as shown earlier.

6 Conclusions and Future Work

The research presented in this paper detailed our explorations into the semantic topic expansion. While a canonical form of the solution was presented, it proved to be too slow to provide an interactive response time. We relaxed the strict interpretations of CTE in the form of ITE, and observed a significant gain in its response time. It was also shown that, as number of anchor terms increased, ITE results became more similar to CTE results. However, it was also observed that, this similarity gets stabilized at a point which is near 0.5. However, independent user evaluations show that ITE results are considered at least as good, and sometimes even better than the CTE results. The episodic hypothesis based on *topical coherence* stating that *all related terms of a given topic can not co-occur together in one context* seems to hold in human generated corpora.

This work can be extended to look at how *topic maps* can be created from a given text corpus. We can also look into using topic expansion for different forms of query expansion requirements on text corpora.

References

1. Kulkarni, S., Srinivasa, S., Arora, R.: Topic expansion using a term co-occurrence graph. Technical report (2012), `http://rootset.iiitb.ac.in/data/138_topic_expansion_using_tcg.pdf`
2. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent dirichlet allocation. Journal of Machine Learning Research 3, 993–1022 (2003)
3. Boyd-Graber, J., Blei, D., Zhu, X.: A topic model for word sense disambiguation. In: Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 1024–1033 (2007)
4. Budanitsky, A., Hirst, G.: Evaluating wordnet-based measures of lexical semantic relatedness. Comput. Linguist. 32(1), 13–47 (2006)
5. Dorow, B., Widdows, D., Ling, K., Eckmann, J.P., Sergi, D., Moses, E.: Using curvature and markov clustering in graphs for lexical acquisition and word sense discrimination. arXiv:cond-mat/0403693 (2004)
6. Lee, Y.K., Ng, H.T.: An empirical evaluation of knowledge sources and learning algorithms for word sense disambiguation. In: Proceedings of the ACL 2002 Conference on Empirical Methods in Natural Language Processing - EMNLP 2002, vol. 10, pp. 41–48. ACL, Stroudsburg (2002)
7. Rachakonda, A.R.: A cognitive model for mining latent semantics in unstructured text. In: Proceedings of VLDB PhD Workshop, Istanbul, Turkey (2012)
8. Rachakonda, A.R., Srinivasa, S., Kulkarni, S., Srinivasan, M.S.: Mining analytic semantics from unstructured text. Technical report (2012), `http://rootset.iiitb.ac.in/data/139_mas.pdf`
9. Zellig Harris, S.: Distributional structure. Word 10, 146–162 (1954)

Extended Tversky Similarity for Resolving Terminological Heterogeneities across Ontologies

DuyHoa Ngo, Zohra Bellahsene, and Konstantin Todorov

LIRMM, University Montpellier 2
firstname.lastname@lirmm.fr

Abstract. We propose a novel method to compute similarity between cross-ontology concepts based on the amount of overlap of the information content of their labels. We extend Tversky's similarity measure by using the information content of each term within an ontology label both for the similarity computation and for the weight assignment to tokens. The approach is suitable for handling compound labels. Our experiments showed that it outperforms existing terminological similarity measures for the ontology matching task.

1 A Typology of Terminological Heterogeneities

In a reduced view, a terminology can be defined as a collection of symbols, where each symbol evokes a concept and refers to a concrete object in the real world. Often across different ontologies a concept is denoted by identical or highly similar labels. However, in many cases, these labels differ significantly because of different conventions in the naming process – a phenomenon known as terminological heterogeneity. This heterogeneity is understood as any difference in spelling between two given terms or labels which are assumed to refer to the same concept, i.e. that have the same meaning. *Spelling*[1] is about the lexical expression of concepts, the actual string of characters that is used to label them. *Meaning* refers to the definitions of these labels found in a thesaurus or a lexical database. We assume that the meaning defines a concept, thus identical meanings imply identical concepts. With respect to their spelling, two terms can be very similar or totally different and still mean the same thing in a given context. Moreover, there are different degrees to which this (dis)similarity is manifested. Mind that the more complex the labels, the higher the probability of observing a heterogeneity. Therefore, mapping labels that are composed by multiple tokens is harder than mapping one-token labels. We introduce a scale of the orthographical closeness of two terms representing the different heterogeneity types starting from the lightest expression of heterogeneity and ending with the use of entirely distinct labels to denote a given concept.

[1] The term "syntax" is used instead by other authors but we find that slightly abusive to its meaning in linguistics.

Terminological heterogeneity types

1. (Almost) identical labels
   – example: in the presence of typos, the use of plural versus singular, etc.
2. Token-wise similarity
   – example: "SumbissionDeadline" vs. "Deadline_Submission".
3. Partial token-wise similarity
   – example: a label is a token of another label, e.g., "Document" vs. "ConferenceDocument".
4. Acronyms / Abbreviations
   – example: “WWW” vs. “WorldWideWeb”, “Misc.” vs. “Miscellaneous”. Token-wise: "MiscTopics" vs. "MiscellaneousTopics".
5. Synonyms
   – example: “booklet” vs. “brochure” (general), “Article” vs. “Paper” (domain specific), “ConferenceDinner” vs. “ConventionBanquet” (token-wise).

Although the heterogeneity types are expressed on orthographical level only, in order to find associations between terms one often needs to apply not only purely spelling-based measures (which would help for heterogeneity type 1), but also semantic similarity measures in order to identify that two in appearance different terms have the same meaning (as in heterogeneity type 5). Finally, mind that linguistic heterogeneity (labels in different natural languages) is not included in this typology, since it goes way beyond spelling mismatches.

Efforts have been made to adapt existing terminological similarity measures from other fields to the ontology matching (OM) task, but there still remain heterogeneities that existing approaches are not able to deal with, especially in the presence of compound concept labels. We address this issue by defining and applying a similarity measure based on techniques coming from the field of information retrieval (IR). More precisely, we make use of the information content (IC) of the tokens composing each label w.r.t. a given ontology. We extend Tversky's similarity measure by using an IC-based weighting of the tokens forming a label.

The paper is structured as follows. We proceed to discuss basic and advanced terminology similarity measures, which have been applied for OM, in Section 2. They are related to the method that we propose in Section 3 and that is defended experimentally in Section 4.

2 Terminological Similarity Measures

In what follows, we assume basic knowledge of the reader on terminological similarity measures for OM [2]. We will use little space for introducing the different measures and focus instead on their strengths and weaknesses in relation to the OM task and the typology presented above.

2.1 Basic Similarity Measures

Here, we consider string-based and language-based measures [2]. *String-based measures* make use of information only relevant to spelling. Following [1], they can be split into edit-based and token-based measures. *Edit-based similarity* of two strings is based on the count of the edit operations required to transform one string into the other (e.g., **Levenstein**). A recent extension called **ISUB** [6] considers not only the similar but also the different parts of two strings. *Token-based similarity* measures compute similarity by splitting the strings into tokens, comparing the tokens by the help of an internal measure and giving the overall similarity of two strings seen as two collections of tokens (e.g., **Q-Grams**). *Language-based measures* rely on linguistic information in order to find the association between terms. They can rely only on the internal linguistic properties of words, or make use of external knowledge resources such as dictionaries, lexicons or thesauri, and look into the semantic relationships in the respective hierarchies.

Discussion. The main advantage of token-based over edit-based measures is their capability to handle compound labels, since they are less sensitive to word swaps (e.g., "MemberConference" and "ConferenceMember"). To overcome tiny variations (e.g., typos), a token-based measure uses an edit-based one as an internal measure. Therefore, token-based measures can be used to deal with heterogeneity type 2, while edit-based measures are appropriate for handling type 1 heterogeneities. Several token-based measures like **TFIDF**, **Jensen-Shannon** and **Fellegi-Sunter** [1] need an external resource to assign weights to tokens. They face the limitation of relying on a large corpus related to a given domain that may not always be available, especially in the OM field. None of these measures can deal with heterogeneity types 3, 4 or 5, since they compute similarity by using spelling-related features only. To overcome heterogeneity type 5, hybrid measures have been proposed (a combination of string-based and language-based similarities). Heterogeneity types 3 and 4 are more difficult to handle and require the use of external knowledge.

Language-based measures, both intrinsic and extrinsic, suffer from two common problems. First, they mainly deal with single words and not with compound labels. For example, neither of the labels "DoctoralThesis" or "PhdDissertation" is found in WordNet although each of their tokens is. The second problem appears when the input words cannot be found in the dictionary due to typos. Therefore, although language-based measures are appropriate to deal with heterogeneities of type 5, they need to be supported by string-based measures.

2.2 Advanced Similarity Measures

Hybrid similarity measures are a combination of string-based and language-based similarity measures. In particular, a hybrid measure can be applied on two levels: between tokens and between compound labels.

One-token labels are aligned by using morphological methods that look at all possible basic forms of each of the two tokens in a dictionary. If the basic forms of both tokens exist, an extrinsic similarity measure is used. Otherwise, the similarity score is computed by a string-based measure. *Compound labels* are first split into sets of tokens. Having the similarity scores for every pair of tokens, we can apply one of the two widely used aggregation methods, **ExtendedJaccard** and **Monge-Eklan** [3], or the **SoftTFIDF** measure [1].

Discussion. Hybrid similarity measures can be used to deal with both type 2 and type 5 of terminological heterogeneity. When two strings have a high number of shared tokens, which are highly similar in spelling or in meaning, the hybrid similarity measure can detect them as a match. But if the number of shared tokens of two strings is small, both token-based and hybrid measures return a low similarity value and they possibly detect these strings as unmatched. Therefore, for type 3 of terminological heterogeneity, we need to exploit other feature information of entities in order to discover mappings between them.

Weighted techniques have several downsides. In SoftTFIDF, a weight is computed by using the TFIDF approach which requires a large corpus – rarely available for a given matching scenario in the OM world. The Extended Jaccard similarity lacks discriminating power. It would assign similar scores to pairs of tokens which have different relations in a semantic network. For example, $ExtendedJaccard(Publication, Magazine) = ExtendedJaccard(Publication, Journal) = ExtendedJaccard(Publication, Periodical) = 1.0$, although according to WordNet, "journal" and "magazine" are siblings and they are both children of "periodical". Finally, asymmetry appears to be a serious drawback of the SoftTFIDF. To overcome these weaknesses, we have designed a (symmetric) similarity measure extending Tversky's method.

3 Extending Tversky's Similarity Measure

In an ontology which represents a given aspect of the knowledge of a specific domain, certain (non-stop) words frequently appear together with other words in concept labels. For example, in the **conference.owl** ontology, which models the conference organization domain, the total number of concepts is 60. The labels of 14 of these concepts contain the word "conference", and 10 contain the word "contribution", whereas, other words like "author" and "speaker" appear only once as a part of a concept label. Therefore, if the words "conference" and "contribution" are found in a compound label, they are unlikely to be keywords. Instead, they are used to emphasize the specific meaning of the associated words in the domain of interest and disambiguate the meaning of the associated words in different domains.

The proposed measure is inspired by the comparison methods of documents in the IR field and will be applied to deal with type 3 of terminological heterogeneity. After stop-words removal, a weighting method is used to assign a weight to each remaining word which represents the relative importance of that word in the document. Finally, a computation method is applied to calculate a similarity score between two documents.

The main difference between label comparison in OM and generic document comparison in IR is that the former is a comparison of short strings, whereas, the latter is a comparison of long texts. In the OM domain there cannot be found a large corpus from which we can extract the necessary statistical information for the similarity computation. Therefore, the techniques used in comparison of documents have to be adapted to the label comparison task. In particular, we are going to discuss weight assignment and similarity computation issues which are strongly related to one other.

Weight Assignment. There are many weight assignment approaches proposed in the IR literature (TF, IDF, TFIDF), mainly based on the frequency of occurrence of each word in a document and in a large corpus. In OM, in the first place, there is a lack of a large corpus (a large number of ontologies describing the same domain). Commonly, only two ontologies in a matching scenario are given. Moreover, because of their high heterogeneity, ontologies may slightly overlap or may be totally disjoint w.r.t. terminology. Thus, the words used in one ontology may differ from those used in the other. Consequently, there may be no benefit of calculating the frequency of words across multiple ontologies. In the second place, the weight of a word depends on the ontology that contains that word. As mentioned above, common words in a specific domain may explicitly appear many times in one ontology. They also may not appear but be implicitly represented in the other ontologies. Therefore, if we take multiple ontologies into account, the frequency of occurrence of the common words and keywords may not be significantly different. Consequently, there is not much difference between common words and keywords in the way they are handled by the similarity computation approach.

In our method, a normalization of the IC of each word appearing in an ontology is considered as its weight. In information theory [5], the IC of an object is inversely proportional to the probability of occurrence of that object. We give the IC of a word t appearing in a label as $IC(t) = \log \frac{|T|}{|N|}$, where, $|T|$ is a total number of concepts in a given ontology and $|N|$ is the number of concepts whose label contains t. On this basis, a word t is assigned a weight as follows:

$$weight(t) = \frac{IC(t)}{\max_{i=1..|T|}\{IC(t_i)\}}. \tag{1}$$

Similarity Computation. An appropriate similarity computation method has two of the desired properties described in [6]: *(i) intelligent*: it should recognize the amount of informativeness that each token carries in a label and reflect that on the the similarity score between labels, and *(ii) discriminating*: it should rarely assign the same similarity value when it compares a label to several other similar labels. For example, it should distinguish the similarity of “Publication” to “Journal” from that to “Magazine” and to “Periodical”. To fulfill these requirements, the similarity measure should take both the weight values of tokens and the similarity values between tokens into account.

The well-known Tversky similarity measure [7] for two objects A and B seen as sets of features and a function f can be given as: $\mathrm{Tv}(A, B) = \frac{2 * f(A \cap B)}{f(A)+f(B)}$. In our case, the objects are labels denoting concepts. Let s_1 and s_2 be two labels and let the function $Tokenize$ return the set of tokens composing a label. Further, let $TokenSim$ be a similarity measure for two terms and let $Share(s_1, s_2) =$

$\{t' \in Tokenize(s_1) \mid \exists t'' \in Tokenize(s_2) \wedge TokenSim(t', t'') \geq \theta\}$. By following Tversky's rule, we give the following definition of the similarity of two labels:

$$\mathrm{ET}(s_1, s_2) = \frac{Common_{s_1,s_2} + Common_{s_2,s_1}}{Total_{s_1} + Total_{s_2}}, \tag{2}$$

where, for $i, j \in \{1, 2\}$ and $i \neq j$, we have

$$\begin{aligned} Common_{s_i,s_j} &= \sum_{t' \in Share(s_i,s_j)} weight(t') \cdot \max_{t'' \in Tokenize(s_j)} (TokenSim(t', t'')), \\ Total_{s_i} &= \sum_{t \in Tokenize(s_i)} weight(t). \end{aligned}$$

The weighting function used in the calculation of the similarity measure can be any function known from the literature. In the particular definition of our similarity measure based on the IC of terms, we have applied the IC-based weight given in Eq. (1). In the next section, we provide a comparison of the outcomes of this measure when different weighting functions are applied.

4 Experiments and Evaluation

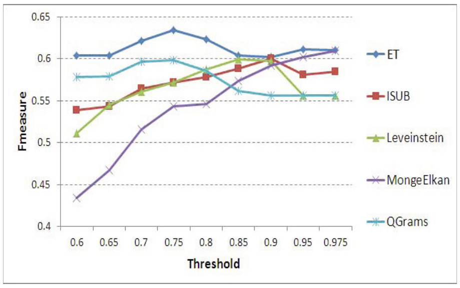
Evaluating an OM task consists in comparing the discovered alignments to a reference alignment by the help of evaluation measures corresponding to the harmonic means of Precision (Pr), Recall (Re) and the F-measure (Fm) computed on a set of n tests per matching scenario. A test corresponds to a particular choice of two input ontologies (a source and a target) and a scenario – to a particular matching task (see [4] for details).

We have conducted a series of experiments on two datasets containing terminologically heterogeneous ontologies – the well-known conference dataset from the OAEI[2] and the dataset from the I3CON conference[3]. We have compared our method to basic "weightless" similarity measures and to more advanced similarity measures using a weighting function in the similarity computation. Among the weightless similarity measures, we have chosen the ISUB, Levenstein, Q-Grams, and Monge-Elkan, for reasons of their successful application in the OM field. In addition, each of these measures is representative for its group (string-based, token-based and hybrid, respectively). The weighted measures that we have used are the SoftTFIDF and the Extended Jaccard. As a weighting function for these measures and our Extended Tversky (ET) measure, we have used the IC-based weight proposed in (1) and the standard TFIDF weighting. A mapping selection module is introduced to filter at a given threshold the best candidate mappings. Our results are presented as a function of the different choices of this threshold. They are given in the figures in Tables 1 and 2 for the conference dataset and Tables 3 and 4 for the I3CON dataset.

[2] The Ontology Alignment Evaluation Initiative, `http://oaei.ontologymatching.org.`

[3] `http://www.atl.external.lmco.com/projects/ontology/i3con.html`

Table 1. Conference dataset. Our method is compared to weightless methods by using IC-based weighting (left) and TF-IDF weighting (right).



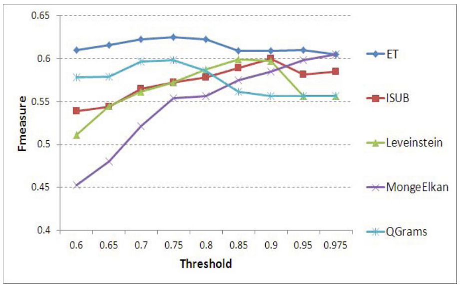
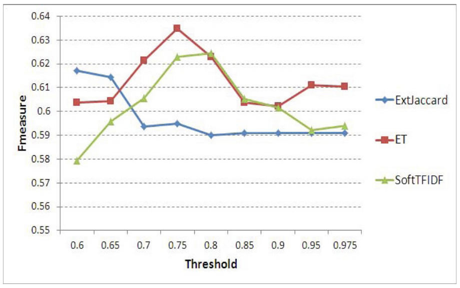
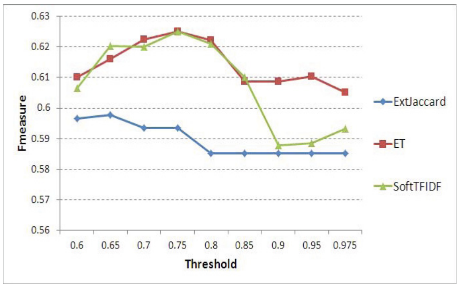


Table 2. Conference dataset. Our method is compared to weighted methods by using IC-based weighting (left) and TF-IDF weighting (right).



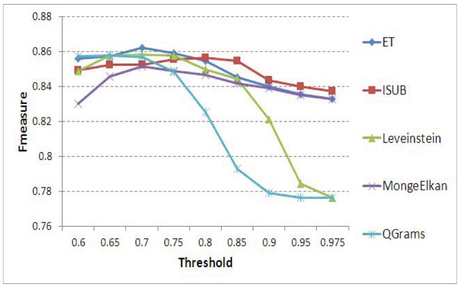


As seen in the figures, the ET measure proposed here clearly outperforms the weightless and weighted state-of-the-art measures on the conference dataset for almost all choices of a mapping filter threshold. On the I3CON dataset, we notice that our measure is dominated by the ISUB and the SoftTFIDF measures for certain threshold values. This behavior is explained by the fact that I3CON almost does not contain pairs of heterogeneity type 3. We can see, nevertheless, that the overall highest F-measure values on this dataset are achieved by the ET measure and this measure does not perform globally worse than the other methods on this dataset.

5 Conclusion

The bigger part of the terminological similarity measures that are currently applied to the OM task are borrowed from neighboring domains. However, due to the differences between these domains and the OM domain, these similarity measures need to be adapted in order to perform well. The greatest challenge is to be able to make use in the best possible way of the austere textual information that comes with the ontologies. Addressing this challenge, we have presented a novel similarity measure that is able to deal with certain terminological heterogeneity types in the ontology matching task, that existing techniques cannot handle.

Table 3. I3CON dataset. Our method is compared to weightless methods by using IC-based weighting (left) and TF-IDF weighting (right).



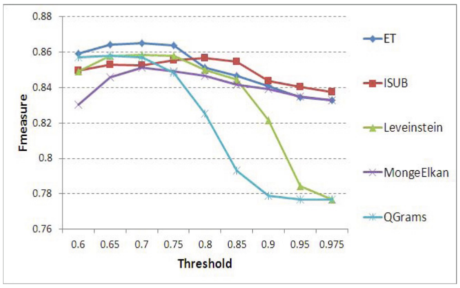
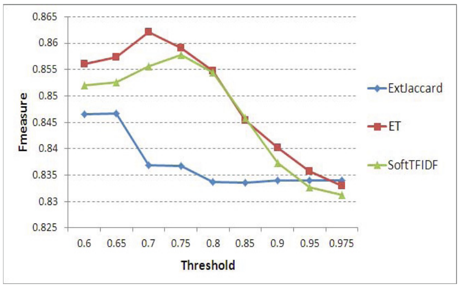
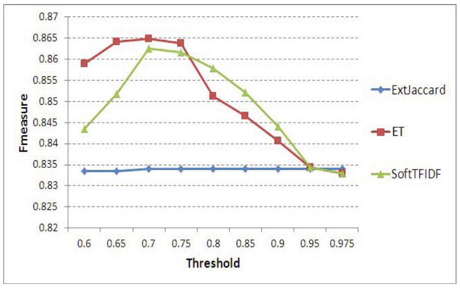


Table 4. I3CON dataset. Our method is compared to weighted methods by using IC-based weighting (left) and TF-IDF weighting (right).





It extends Tversky's similarity and uses the information content of each term within an ontology both for the similarity computation and for the weight assignment to terms. The experimental results show that this similarity measure globally outperforms all existing state-of-the-art techniques, including simple weightless measures and more advanced approaches.

References

1. Cohen, W.W., Ravikumar, P.D., Fienberg, S.E.: A comparison of string distance metrics for name-matching tasks. In: IIWeb, pp. 73–78 (2003)
2. Euzenat, J., Shvaiko, P.: Ontology matching. Springer (2007)
3. Monge, A.E., Elkan, C.P.: An efficient domain-independent algorithm for detecting approximately duplicate database records. In: SIGMOD WS on Research Issues on Data Mining and Knowledge Discovery, pp. 23–29 (1997)
4. Ngo, D., Bellahsene, Z., Todorov, K.: Opening the black box of ontology matching. In: Cimiano, P., Corcho, O., Presutti, V., Hollink, L., Rudolph, S. (eds.) ESWC 2013. LNCS, vol. 7882, pp. 16–30. Springer, Heidelberg (2013)
5. Shannon, C.E.: Prediction and entropy of printed English. Bell Systems Technical Journal, 50–64 (1951)
6. Stoilos, G., Stamou, G., Kollias, S.D.: A string metric for ontology alignment. In: Gil, Y., Motta, E., Benjamins, V.R., Musen, M.A. (eds.) ISWC 2005. LNCS, vol. 3729, pp. 624–637. Springer, Heidelberg (2005)
7. Tversky, A.: Features of similarity. Psychological Review 84, 327–352 (1977)

Ontology-Based Semantic Annotation of Documents in the Context of Patient Identification for Clinical Trials

Peter Geibel[1], Martin Trautwein[4], Hebun Erdur[2], Lothar Zimmermann[1], Stefan Krüger[1], Josef Schepers[1], Kati Jegzentis[3], Frank Müller[1], Christian Hans Nolte[2], Anne Becker[4], Markus Frick[4], Jochen Setz[4], Jan Friedrich Scheitz[2], Serdar Tütüncü[2], Tatiana Usnich[2], Alfred Holzgreve[4], Thorsten Schaaf[1], and Thomas Tolxdorff[1]

[1] Institut of Medical Informatics, Charité - Universitätsmedizin Berlin
[2] Department of Neurology (CBF), Charité - Universitätsmedizin Berlin
[3] Center for Stroke Research Berlin (CSB), Charité - Universitätsmedizin Berlin
[4] Vivantes - Netzwerk für Gesundheit GmbH

{peter.geibel,thomas.tolxdorff}@charite.de, martin.trautwein@vivantes.de

Abstract. In this paper, we describe the use of ontologies in the context of a system for identifying patients that are eligible for clinical trials. The main purpose of this clinical research data warehouse (CRDW) is to support patient recruitment based on routine data from the hospital's clinical information system (CIS). In contrast to most other systems for similar purposes, the CRDW also makes use of information present in clinical documents like admission reports, radiological findings and discharge letters. The linguistic analysis recognizes negated and coordinated phrases. It is supported by clinical domain ontologies that enable the identification of main terms and their properties, as well as semantic search with synonyms, hypernyms, and syntactic variants. The CRDW system is currently being tested at hospitals of the *Charité – Universitätsmedizin Berlin* and the *Vivantes – Netzwerk für Gesundheit GmbH.* In the paper, we provide an evaluation of the system based on real world data obtained from the daily routine work of the study assistants.

Keywords: Ontologies, RDFS, secondary use of health data, patient recruitment, clinical data warehouse.

1 Introduction

In recent years, the secondary use of clinical data has been considered an important topic of research since it enables medical progress based on data that are currently only used for treatment, administrative, and billing purposes. In this paper, we present research on a data warehouse system that is currently being developed in the context of a collaboration between *Charité – Universitätsmedizin Berlin*, the largest German university hospital, *Vivantes – Netzwerk für Gesundheit GmbH*, Germany's largest state-owned healthcare corporation, and a Berlin-based SME software partner.

The main purpose of the clinical research data warehouse CRDW is to support patient identification for clinical trials based on routine data from the clinical information system (CIS). The CRDW allows finding patients that meet the inclusion and exclusion criteria of clinical trials. These criteria, for instance, correspond to patient data (age, sex), lab values, and coded information on diagnoses (ICD-10 codes) and procedures (OPS codes). Using also unstructured data, i.e., doctor's letters and other clinical documents, allows investigators to formulate more fine-grained criteria compared to using coded information alone. Furthermore, most medications are only documented in admission reports and discharge letters.

Other than our project, relevant efforts include I2B2 [13], EHR4CR (`http://www.ehr4cr.eu`), Cloud4Health (`http://www.cloud4health.de/`), and the KIS REK project [6]. Our system has a strong focus on combining computational linguistics and ontology-based information extraction.

In this paper, we would like to report our experiences in modeling and using ontologies [24], i.e., clinical knowledge bases, which are used by our software for extracting structured information from texts [15,4]. Also, the system is currently being tested at the Department of Neurology of the Charité and at the Clinic of Neurology – Stroke Unit – Center for Epilepsy (Vivantes Humboldt-Klinikum). We will describe qualitative and quantitative evaluation results in the context of these departments.

For the Clinic of Neurology (Charité), we present an evaluation that is based on real patient data and a series of real clinical trials. We compared the predictions of our system to the assessments of the trial team of the Center for Stroke Research Berlin (CSB) in order to obtain estimates of precision, recall, sensitivity, and other quantities. In particular, we will demonstrate that using information from unstructured data improves the performance of the system. This evaluation on real world data is the second main contribution of our paper.

This paper is structured as follows. After an overview of the system that is given in section 2, we describe the requirements of our specific ontology model (section 3) including a short overview of existing medical ontologies. Section 4 describes evaluation results for the pilot phases at the Clinic of Neurology (Charité) and the Clinic of Neurology (Vivantes). The conclusions can be found in section 5.

2 Overview of the System

The current deployment of the CRDW system at the Charité Berlin is based on data from the IS-H/i.s.h.med CIS modules (SAP/Siemens), which are integrated with patient information extracted from documents of the GE radiological system. The data is loaded into the data warehouse by an ETL (Extract, Transform, Load) process. In the case of Vivantes, a custom-built HL7 adapter provides selected data of neurological case records for an instance of the system.

All data is stored in a pseudonymous manner in the CDS (clinical data storage), integrating data from both structured and unstructured sources. This means

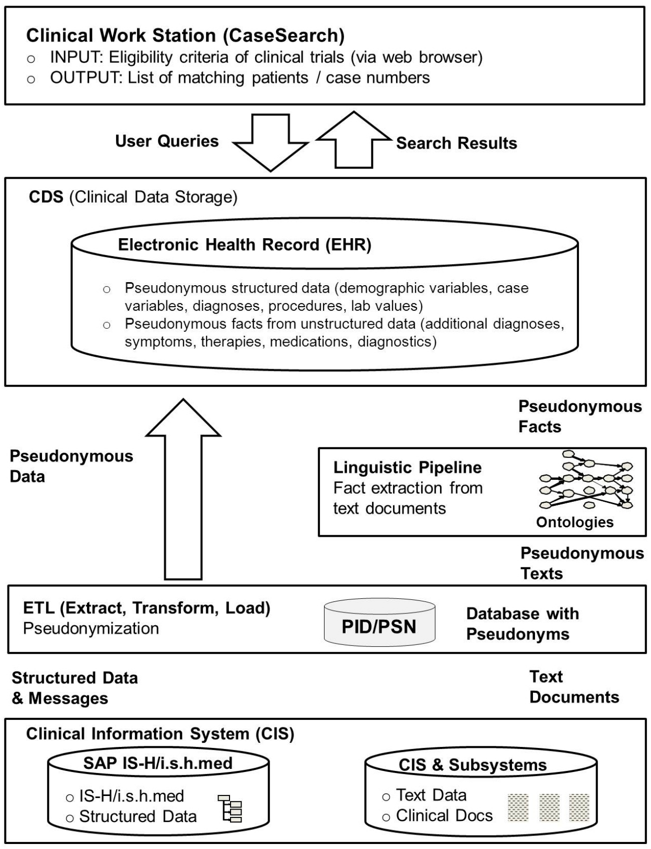

Clinical Work Station (CaseSearch)
INPUT: Eligibility criteria of clinical trials (via web browser)
OUTPUT: List of matching patients / case numbers
User Queries
Search Results
CDS (Clinical Data Storage)
Electronic Health Record (EHR)
Pseudonymous structured data (demographic variables, case variables, diagnoses, procedures, lab values)
Pseudonymous facts from unstructured data (additional diagnoses, symptoms, therapies, medications, diagnostics)
Pseudonymous Facts
Pseudonymous Data
Linguistic Pipeline
Fact extraction from text documents
Ontologies
Pseudonymous Texts
ETL (Extract, Transform, Load)
Pseudonymization
PID/PSN
Database with Pseudonyms
Structured Data & Messages
Text Documents
Clinical Information System (CIS)
SAP IS-H/i.s.h.med
IS-H/i.s.h.med
Structured Data
CIS & Subsystems
Text Data
Clinical Docs

Fig. 1. Architecture of the CRDW System

that the patients' names, addresses, birth dates, etc. are replaced by aliases generated by the system in order to meet data protection regulations. In addition to the pseudonymization of structured data, our system is also able to remove identifying data in text, roughly following the American HIPAA privacy rule, cf. [25]. For removing personal information such as names and addresses, we combine using information from the CIS (e.g., name, address, date of birth) with computer linguistic methods, which, for instance, look for textual clues like "Herr" (Mr.), "Dr.", etc. in order to find occurrences of names in a document.

Within the system, the so-called *linguistic pipeline* is responsible the extraction of information from texts. The extraction is based on a linguistic analysis [12,8], which identifies terms, phrases, and sentences together with grammatical constructs such as negation ("not") and coordination ("and", "or"). This first, morphosyntactic part of the pipeline is based on the GATE framework [5]. For parsing, we use JAPE, the finite state transducer that comes with GATE.

The extraction of negated phrases is based on NegEX [3]. The POS-tagger (Apache OpenNLP) was trained on a tagged corpus of documents from both hospitals. In addition to the knowledge base, the pipeline also comprises resources that support spell-checking and the morphological analysis of words (lemmatization and compound resolution).

In the second part of the linguistic pipeline (so-called concept mapping), the identified terms and phrases are mapped to medical concepts in a semantic knowledge base, which contains etiological, morphological, topological and procedural information. This knowledge allows identifying the main pieces of information, assigning them to classes like "diagnosis", "therapy", "anatomical structure", and so on. For instance, in a phrase like "infarction of the MCA", "infarction" will be identified as a diagnosis, whereas the anatomical structure, "MCA", provides information about the location of the infarction.

Using the linguistic pipeline together with a knowledge base enables semantic search. For instance, the user can execute queries for synonymous terms like "stroke" and "cerebral infarction". The availability of taxonomical information allows searching with generalized queries like "infarction of a cerebral artery". This query then matches "infarction of the MCA" and variants hereof like "middle cerebral artery infarct", "media infarction", and even "The MRI depicts an infarct in the territory of the MCA". Note that a plain text search is bound to fail in these cases. Also, negated phrases do not count as hits in our system.

The software component that is responsible for finding eligible patients is called *CaseSearch*. It allows defining clinical trials together with their inclusion and exclusion criteria. These criteria are matched against both structured patient data and medical facts, which were extracted from documents by the linguistic pipeline. The *CaseSearch* provides web-based user interfaces that show current and past patients together with their eligibility for the defined trials. Translating the criteria of the study protocol into the language of the system is an important step of the overall process, since it has to take into account the availability and quality of data. It also requires knowledge of both the respective disease (and its treatment), and of processes at the respective clinic (patient paths, documentation, recruitment).

3 Constructing the Ontologies

In this section, we describe the requirements of our application, the chosen ontology model, and the knowledge engineering approach taken.

3.1 Analyzing Clinical Trials and Data Sources

We approached the problem of finding the right ontology formalism, structure and content from several directions. Since the main purpose of the CaseSearch is to find patients for clinical trials, we analyzed clinical trials in the field of *ischemic strokes*, in particular such that where conducted by the Clinic of Neurology of the Charité.

As an example, the clinical trial TRELAS [21,20] investigates Troponine T elevation in patients with ischemic stroke. In order to find patients meeting the criteria of this trial, we have to determine patients that suffer from a stroke or a non-ST elevation myocardial infarction (NSTEMI), and have a Troponine T level above $0.05\mu g/l$. Patients with a Creatinine value above $1.2mg/dl$ are excluced. Patients with a stroke can be found by looking at ICD-10 coded diagnoses in the CIS system, and by analyzing admission reports and radiological findings, which additionally allows determining the location of a stroke.

We analyzed stroke studies conducted at Charité with respect to their semantic structure and typical content. We were able to determine the following groups of criteria:

– *Main diagnosis* (IS-H/i.s.h.med)
– *Age and Sex* (IS-H/i.s.h.med)
– *Localization of infarction and other findings* (radiological report)
– *Time of identifying event* (admission report)
– *Severity:* in particular the NIH stroke score (admission report)
– *Symptoms* (admission report, discharge letter)
– *Lab data* (IS-H/i.s.h.med)
– *Prior medication, therapies* (admission report, previous cases)
– *Other diseases* (admission report, IS-H/i.s.h.med)
– *Medicolegal aspects:* pregnancy, ability to consent, risk factors, etc.

In a similar manner, our analysis of epilepsy trials conducted at Vivantes identified the following criteria as the most relevant:

– *Main diagnosis* (structured value, HL7)
– *Age and sex* (structured value, HL7)
– *Epilepsy type/type of seizures* (first-aid report, HL7/electronic document exchange)
– *Frequency of seizures and/or time of last seizure* (first-aid report)
– *Symptoms:* physical effects of seizures such as a bitten tongue, incontinence etc. (first-aid report)
– *Lab data (structured values)*
– *Prior medication and therapies* (first-aid report, HL7/electronic document exchange; admission report(s) of previous case(s), CIS)
– *Other diseases:* e.g., severe renal, hepatic, cardiologic or neurologic diseases (dto.)
– *Medicolegal aspects*

The analysis of the trials showed that in addition to features of the patient like age and sex, obviously diagnoses, symptoms, findings, and therapies are most important. When building the ontologies, we therefore focused on two main concept classes: *observations* (diagnoses, symptoms, and findings) and *therapies* (including medications). A concept in these classes can have properties like localizations (for which we introduced an *anatomy* class), and modifiers like *acute*, *chronic*, *left*, *right*, *frontal*, *parietal*, etc. These attributes for observations and therapies were collected in an *attribute* class.

Aside from the "has-attribute" relation, the criteria of trials rarely require semantic relations *between*, for instance, diagnoses and therapies. As an example, the doctor's letter might mention that a medication was prescribed in order to treat a specific disease. However, the criteria of trials rarely ask for this relationship. They usually just specify the diagnosis and the medication without detailing the semantic relationship between them. This means that this relation does not have to be extracted from documents and it does not have to be part of the ontologies.

Aside from determining what we needed to model, the question was what we actually did not need to model. For instance, for patient identification, the system does not have to be able to derive diagnoses based on symptoms: This was already done by the treating physicians. Also, we we found that, other than "part-of", we did not have to model functional relationships within the body or processes related to the respective disease or its treatment. We found that this knowledge is important when formulating the right criteria for the system, but it does not have to be part of the knowledge base.

Regarding the logical structure of the trial protocols, we found that both inclusion and exclusion criteria are relatively simple logical expression that can be expressed using AND, OR, and NOT. However, additional complexity arises from the fact that some criteria are time-dependent. For instance, it might be required that a patient has not been treated with a specific type of medication within in the last three months. Moreover, documents frequently state negated and uncertain information. This must be taking into account when matching trial criteria and extracted facts, see section 3.4. Also, quantification and counting is required to some extend.

As a general result of the analysis, we were able to verify that we cannot rely on a single type of information alone but need the combination of structured data and facts extracted from documents. For instance, the location of a stroke can only be found in text documents whereas the NIHSS (NIH stroke score) is frequently documented in the admission report in a structured manner. Both sources, however, are not 100 % complete. This means that the original criteria frequently have to be translated into system queries that combine (CRDW) criteria that are semantically similar, but pertain to different data sources.

Regarding the question of data quality and availability, we found that frequently the required information is not documented in the CIS (e.g., medicolegal aspects). Or it is not yet available, when searching for eligible patients. This is, for instance, the case for certain stroke studies, which require that a specific treatment is begun within only a couple of hours of the stroke event. Coded information is frequently only available after a couple of days. The same holds true of discharge letters, which are only available after the patient has already left the hospital.

3.2 Technical Considerations

When starting to develop a prototype of the system, it was necessary to make a decision for a specific semantic technology. In order to get started, we decided to set up a SESAME server (`http://www.openrdf.org/`) and to use RDFS (RDF Schema) for modeling some basic concepts.

In RDFS, one can, for instance, establish subclass relationships between concepts and subproperty relationships between properties. In our ontologies, we use the following primitives:

- `rdfs:subClassOf` for the subclass relationship
- `rdfs:subPropertyOf` for properties
- `rdfs:label` for specifying synonyms

However, it is not possible in RDFS to define a concept as the conjunction, disjunction or negation of other concepts. Neither is it possible to state rules that derive properties or class membership for instances.

The lack of rules is partly compensated by the query interface of the CRDW. In our system, we allow conjunction and disjunction of positive criteria plus a conjunction of negated criteria. This means that although we cannot state rules or complex concept definitions, the user can use logical combinations of search criteria. For instance, instead of inferring that a patient has diabetes from the fact that he or she is treated with Metformin, the user of our system can issue a search for `diabetes OR metformin` (potentially plus other indicators for diabetes).

An example, in which (simple) rules are helpful, are implicit attributes. For instance, we have the concept `epilepsy` in our ontology plus its potential attributes `generalized` or `focal`. There is a special type of epilepsy, called matutinal epilepsy, which is always of type `generalized`. Therefore, in a medical document, one does never find the term "generalized matutinal epilepsy", since "generalized" is redundant. This means that this information must be inferred by the system. In the current state of the system, the user has to define the query `epilepsy(generalized) or matutinal epilepsy`. Alternatively, we allow internal rewriting of linguistic expressions to include more information.

As part of our Scrum [22] software development process, we carefully evaluated several alternatives and extensions to RDFS: PROLOG [11], F-Logic/Object Logic [10], Datalog [7], Production Rules (Drools) [2], RIF [9], OWL 2 [27], SPARQL Rules (SPIN) [14]. Some of these languages are very powerful but lack built-ins for modeling ontologies (e.g., PROLOG, Drools, Datalog). The remaining approaches pertain to ontologies, however many of them could not be considered mature enough to be included into a commercial software product, which is the ultimate goal of our project. In general, we found SPIN the most attractive approach since it is relatively powerful but lightweight, and it features negation as failure. However, we found that the problems introduced by not having rules were not critical. We therefore postponed the introduction of SPIN to future versions of our system.

3.3 Extracting Patient Information from Documents

An important insight of the project was that not only the ontology contents but also its structure has to support the task of extracting information from texts.

As stated in the previous section, we wanted to extract diagnoses and therapies from text sources. In the first versions of the system, one of the problems was that attributes were frequently attached to the wrong diagnoses. In order to help the concept mapping algorithm with attaching properties to the right main terms, we decided to specify possible attributes for each concept in order to reduce ambiguities.

Consider, for instance, the phrase "acute MCA infarction". In the ontology, infarction is specified to have potential attributes "`:Acute`" and `:ArteriaCerebriMedia`. This is achieved by the following declaration:

```
:Infarction rdf:type rdfs:Class ;
       rdfs:subClassOf :Observation;
       :label "Infarction" ;
       :hasLocalization :Artery ,
              :Brain ,
              :Heart ;
       :hasAttribute :Position ,
              :Modifiers ,
              :InfarctionAttributes .
```

`:Acute` is defined to be a subclass of `:Modifiers`.

As can be seen from the example, we extensively made use of the meta-modelling feature of RDFS that allows to treat classes as instances. In terms of general AI terminology, our approach can be seen as frame-based [19]: We specify slots for each disease, which are filled by the linguistic pipepline based on information given in the text.

3.4 Mapping Patient Information to Criteria

Clinical trials usually pertain to information that is assumed to be certain. Mapping positive facts to criteria is therefore straightforward. However, radiological findings, admission reports, and doctor's letters frequently contain negated and uncertain information. The handling of such information can be a relatively difficult topic in the clinical context since there does not seem to be a strategy that is always the right one.

The criteria of a clinical trial are divided into inclusion and exclusion criteria, the latter of which can be considered negated criteria. Exclusion criteria are usually evaluated in a "closed world manner". This means that an exclusion criterion only matches whenever there is no corresponding fact that can be considered certain. In some cases, however, doctors preferred a more conservative "open world approach", with exclusion criteria matching only explicit negative statements. We are therefore planning on allowing the user to choose between open and closed world semantics.

A related issue is the handling of uncertain facts (e.g., suspected diagnoses). For instance, we found it crucial that the system avoids *false negatives*, i.e.

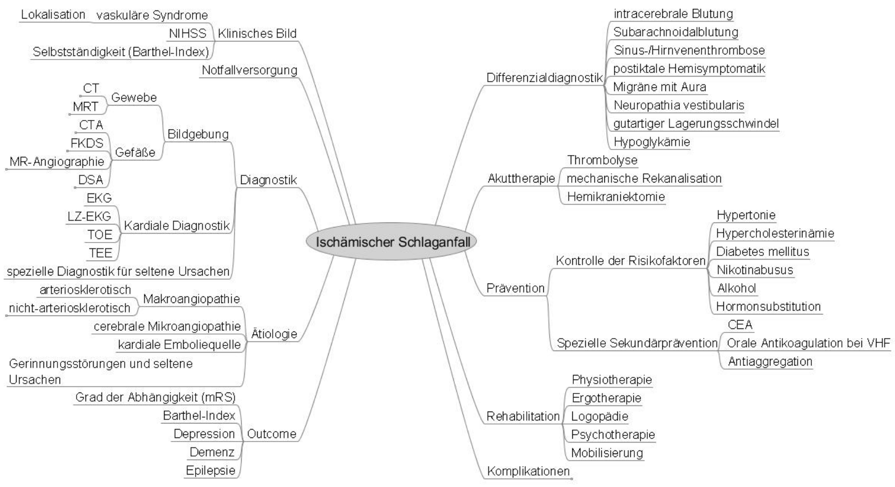


Fig. 2. Mindmap for ischemic stroke (simplified)

patients that are not suggested for a trial although they are potential candidates. This means that if there is an *uncertain* fact for an exclusion criterion, the respective criterion is not supposed to match. In contrast, inclusion criteria are (usually) required to match also uncertain facts. Since this is not always the case, though, we are planning on letting the user decide the matching behavior of each criterion.

3.5 Modeling Diseases

In order to better understand the structure of disease models, we asked the doctors in our group to draw mindmaps of the diseases. We were primarily interested in the following diseases: *ischemic stroke*, *idiopathic parkinson disease*, *multiple sclerosis*, and *epilepsy*. For instance, the mindmap of *ischemic stroke* consists of the branches: clinical picture, etiology, diagnostics, differential diagnoses, acute therapy, emergency medical care, complications, outcome, rehabilitation, prevention, see fig. 2. Similar branches can also be found in the mindmaps for the other diseases.

For the task of patient identification, however, we found that is only necessary to represent the concepts present in the mindmap, but not the semantic relations corresponding to the branches and their labels: For instance, we can can find stroke patients suffering from a paresis by simply issuing the query `stroke AND paresis`. In order to be able to answer this query, however, it is not necessary to represent the fact that a paresis is a potential symptom of a stroke in the ontology. This confirmed that we could use the relatively simple ontology model sketched in the previous section.

Other than understanding the required structure of our ontologies, in the beginning the mindmaps served as a basis for defining the ontologies. Later on, we added concepts mentioned in annotated documents and also incorporated concepts that are found in the ICD-10 classification of disease.

3.6 Enriching the Ontologies

In order to identify further concepts that needed to be part of the ontology, and also in order to be able to construct a set of test sentences, the clinical doctors were asked to annotate phrases in a set of example documents chosen by them. In the beginning, they just used a text marker on a print-out of the documents. Based on the annotated documents, the ontology models were extended by a knowledge engineer. In the future, however, we plan to utilize an annotation tool that allows the doctors to annotate the relevant phrases graphically followed by a semi-automatic step of ontology extension.

At the moment, the ontologies are being developed by several people who work at different locations and have different backgrounds. Yet, there is a strong overlap between ontologies for separate diseases because co-morbidity has also to be modeled to some extend, and anatomical and attributive information have to be shared between ontologies. For now, we decided to use independent ontology modules. In order to be able to use different modules in parallel, we plan to use techniques of ontology alignment [23,26].

3.7 Available Clinical Ontologies

In our project, we also investigated if we can use already existing knowledge bases. The results are summarized in the following.

UMLS (Unified Medical Language System) [1] is a so-called meta-thesaurus, which combines several thesauri by the means of a common semantic network. Since most of the resources are not available in German, and license conditions are frequently problematic for the use in a commercial software, we were not able to use this powerful resource in our project. UMLS is also used for the linguistic component in I2B2 [13], a system that has a similar purpose as the CRDW.

MESH (Medical Subject Headings) [16] is a controlled vocabulary for indexing the MEDLINE/PUBMED database. There exists also a German version (MESH GER), which we licensed for the project. However, in the end we did not use it in our software since the concepts did not meet the requirements of an ontology and were not consistent with our modeling strategy. Consequently we favored modeling the ontology from scratch with the help of domain experts.

As an example, there is a MESH GER concept called "Infarkt, A. cerebri media". The first problem is that we need two separate concepts in our ontology: a diagnosis and a localization. Moreover, the MESH GER concept comprises non-synonymous labels like "A.-cerebri-media-Syndrom", "A.-cerebri-media-Embolus", "A.-cerebri-media-Thrombose", "Left Middle Cerebral Media Infarction", "Right Middle Cerebral Media Infarction" and several variants of

"Infarkt, Arteria cerebri media". As a third problem, we found that many concepts relevant for us where missing.

OpenGalen: The GALEN Common Reference Model (CRM) is a clinical terminology, which was developed in a project funded by the European Union. The English version is available as an OWL download whereas we could not find any German version. In general, we found the structure of the GALEN common reference model much too complex for our project. We had the feeling that constructing a simpler ontology from scratch is preferable to adapting the structure of the GALEN model for our purposes.

SNOMED/SNOMED CT: SNOMED CT (Systematized Nomenclature of Medicine – Clinical Terms, [18]) is a well-known health care terminology. SNOMED CT consists of a "IS_A" hierarchy along with the possibility to define concepts based on attributes. SNOMED, the predecessor of SNOMED CT, was defined using 11 groups of concepts. Since there is no (available) German version of either SNOMED or SNOMED CT, we were not able to use it in our project. The same reason prevented us from using the Foundational Model of Anatomy (**FMA**, [17]) and **RADLEX**.

ICD-10: The "International Statistical Classification of Diseases and Related Health Problems, 10th Revision" (ICD-10) is the most important classification of diagnoses. It is widely used for billing purposes in German hospitals. Although the ICD-10 is quite broad on the one hand, it is not fine-grained enough for our purposes: for instance, with respect to the diagnosis "stroke", one is usually interested in the specific location of the stroke. However, the ICD-10 only distinguishes between cerebral and pre-cerebral arteries. **OPS** plays a similar model as ICD-10. We use it in our software for searching structured data, but do not use it for extracting therapeutic information from texts.

4 Evaluation

We evaluated the performance of the CRDW with respect to the assessments of the trial team of the Center for Stroke Research Berlin (CSB), Charité, see section 4.1. The results of this evaluation are based on both structured and unstructured data. In order to evaluate how useful unstructured data for patient recruitment are, we also compared different versions of the data set. Details of this experiment are given in section 4.2. In cooperation with the trial team of the Center for Epilepsy at the Vivantes Humboldt-Klinikum, we furthermore evaluate the usability of the CRDW in productive use. First results of the qualitative evaluation can be found in section 4.3.

4.1 Charité – Universitätsmedizin Berlin

In the evaluation, we considered 16 stroke trials, which are currently being conducted at the Clinic of Neurology, and compared the performance of the CRDW to that of the trial team. whose assessment of patients was considered the gold standard. The data were collected from January to March 2013. For each trial, we determined the following quantities:

Table 1. System performance for several clinical trials: sample size, number of true positives, false positives, true negatives, false negatives, recall (sensitivity), specificity, precision (positive predictive value), negative predictive value, F-measure

Trial	N	TP	FP	TN	FN	Recall	Spec.	Prec.	NPV	F
Trial 1	268	1	20	246	1	0.50	0.92	0.05	1.00	0.09
Trial 2	257	39	126	82	10	0.80	0.39	0.24	0.89	0.36
Trial 3	177	0	2	175	0	1.00	0.99	0.00	1.00	0.00
Trial 4	268	1	23	242	2	0.33	0.91	0.04	0.99	0.07
Trial 5	136	1	11	124	0	1.00	0.92	0.08	1.00	0.15
Trial 6	253	51	90	101	11	0.82	0.53	0.36	0.90	0.50
Trial 7	255	44	120	80	11	0.80	0.40	0.27	0.88	0.40
Trial 8	267	17	22	228	0	1.00	0.91	0.44	1.00	0.61
Trial 9	259	13	27	219	0	1.00	0.89	0.33	1.00	0.49
Trial 10	251	0	0	250	1	0.00	1.00	1.00	1.00	0.00
Trial 11	268	0	7	261	0	1.00	0.97	0.00	1.00	0.00
Trial 12	236	8	61	166	1	0.89	0.73	0.12	0.99	0.21
Trial 13	269	6	35	226	2	0.75	0.87	0.15	0.99	0.24
Trial 14	254	52	118	79	5	0.91	0.40	0.31	0.94	0.46
Trial 15	245	2	31	209	3	0.40	0.87	0.06	0.99	0.11
Trial 16	245	3	2	236	4	0.43	0.99	0.60	0.98	0.50

- N: total number of patients considered
- TP (true positives): number of eligible patients (according to the trial team) that were found eligible by the CaseSearch
- FP (false positives): number of non-eligible patients that were found eligible by the CaseSearch. False positives increase the workload when using our system.
- TN (true negatives): number of non-eligible patients that were found non-eligible by the CaseSearch
- FN (false negatives): number of eligible patients that were found non-eligible by the CaseSearch. This is the most critical quantity since it means that the user of the system might miss potential candidates.

Table 1 shows the evaluation results when using both structured and unstructured data. For the 16 selected neurological trials, the table shows the recall (sensitivity) $\frac{\mathrm{TP}}{\mathrm{TP+FN}}$, specificity $\frac{\mathrm{TN}}{\mathrm{TN+FP}}$, precision $\frac{\mathrm{TP}}{\mathrm{TP+FP}}$ (positive predictive value), the negative predictive value $\frac{\mathrm{TN}}{\mathrm{TN+FN}}$, plus the so-called F-measure. The F-measure is the harmonic mean of precision and recall. It is defined as

$$F = \frac{\mathrm{P} \cdot \mathrm{R}}{\mathrm{P} + \mathrm{R}} .$$

There is a number of trials, for which there is only a small number of candidates (= TP + FN). This means that the computed quantities are not very reliable and might attain very high or low values. For instance, if there are no

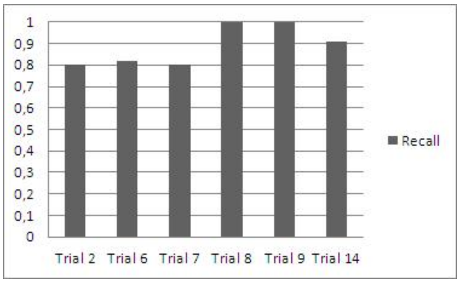


Fig. 3. Sensitivity/Recall (only trials with 10 or more candidates)

candidates for a clinical trial, the recall is defined as 1.0. In the following, we therefore considered only trials with more than 10 candidates.

Fig. 3 shows the recall (sensitivity) of the system for the six remaining trials. The diagram shows that, in all six cases, we were able to achieve a good to high sensitivity. This means that the system suggests most of the eligible patients. Achieving good recall was of uttermost importance for the task of patient recruitment.

The specificity (i.e., recall of non-candidates) corresponds to the probability that a patient is not suggested by the system if he or she is not eligible for a trial (see fig. 4). The specificity values are medium to high.

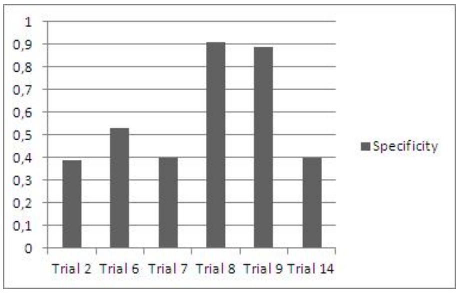


Fig. 4. Specificity

Precision is an estimate of the probability that whenever the system suggests a patient for a clinical trial, he or she is actually eligible. Compared to recall and specificity, the precision attains lower values (see Fig. 5). This means that the system tends to incorrectly suggest patients as candidates. This reduces the potential amount of time that can be saved when working with the system.

One problem regarding precision is that not all necessary information is documented in the clinical information system. Some information is just missing or incomplete (e.g., NIH stroke score, medications) due to the work load in the ER. Other information can only be obtained by talking to the patient or by further

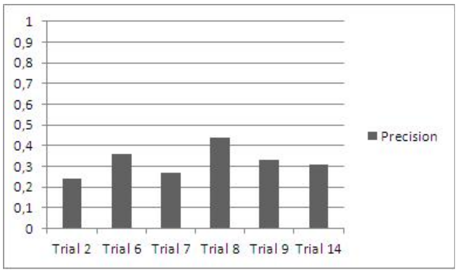


Fig. 5. Precision

examinations. This means that it is usually not possible to attain a precision of 1.0. In order to still improve the performance of the CRDW, we are currently in the process of increasing the logical expressiveness of our query interface, which does not correspond to full SPARQL yet. Other improvements concern the handling of negation and uncertainty as described earlier.

The negative predictive value (i.e., precision for non-candidates) corresponds to the probability that, if the system predicts non-eligibility, the corresponding patient is not a candidate. The negative predictive values are relatively high meaning that negative predictions have a relative high probability of being correct. For reasons of space, we leave out the respective diagram.

4.2 Structured vs. Unstructured Data

In order to determine the usefulness of our ontology-based approach, we considered two variants of the data and the study criteria:

– S+U: This corresponds to the complete data set, comprising unstructured data (U) as well as structured data (S). The trial criteria for “S+U” were defined by medical doctors, specialized in the field of neurology. When setting up the criteria, their focus was on recall while at the same time achieving acceptable levels of specificity and precision.
– U: Unstructured data only. Possible criteria correspond to coded diagnoses (ICD-10) and procedures (OPS), lab values, age, sex, and some structured information from the admission report like the stroke scores. For each trial, the criteria for “version S” were obtained by removing those conditions for “version S+U” that pertain to information contained in documents.

The table 2 shows recall, specificity, precision, negative predictive value, and F-measure averaged over all 6 trials. Using both structured and unstructured data results in a high recall and a medium specificity. Dropping conditions from the criteria that pertain to texts results in lower averaged recall, precision, specificity, negative predictive value, and F-measure.

Since the number of trials, 6, is relatively small, we could not show that the differences are statistically significant. We therefore performed a statistical

Table 2. Recall, specificity, precision, negative predictive value, F (averages over all trials)

	Rec.	Spec.	Prec.	NPV	F
S+U	0.89	0.59	0.32	0.94	0.47
S	0.81	0.54	0.31	0.92	0.43

analysis of the single trials. For recall and specificity it is possible to determine significant differences using McNemar's test. The list of patients is considered a related sample for the two versions of the trial. In the case of recall, a patient is labeled with 1.0 if he or she is found eligible by both the trial team and our software. If our software fails to identify an eligible patient, he or she is labeled with 0.0. All other case are not included in the sample. Specificity is handled analogously.

Table 3 summarizes recall and precision for both versions of all trials. With respect to recall, version "S+U" is better or equal than "U" in 14 cases out of 16. We ran the SPSS McNemar's test in order to determine the trials for which there are differences which are statistically significant. The table gives the p-values for the one-sided tests. In the table, a "+" occurs whenever "S+U" performs significantly better than "U", i.e., if $p \leq 0.05$ holds. If "U" is significantly better than "S+U", there is a "-". For two trials, we found that "S+U" performed significantly better than "U". "=" means that the null hypothesis cannot be rejected, i.e., equal probabilites are assumed. There are no trials, for which "U" performed significantly better with respect to recall. With respect to specificity, "S+U" is better or equal than "U" in 11 cases out of 16. In eight cases, "S+U" is significantly better than "U". The reverse is true in only 4 cases.

4.3 Vivantes - Netzwerk für Gesundheit GmbH

As a state-owned hospital group, the prime objective of Vivantes is to provide community health care services. Nevertheless, clinical research is in strong focus, and a large number of clinics engage in the execution of (mostly sponsor-initiated) clinical trials. The crucial requirement for the CRDW thus is that it has to fit perfectly in to the interwoven processes of health care and clinical research. Starting from this situation, we evaluate the usability of the CRDW system in production use.

In our experiment, a custom-built HL7 adapter provides selected data of neurological case records for an instance of the CRDW. The data comprise both structured data and clinical documents out of the HL7 data stream. Since March 2013, members of the trial team of the Neurological Clinic at the Vivantes Humboldt-Klinikum are using the CaseSearch application for case identification. At present, the trial team is exclusively conducting epilepsy trials.

Table 3. Recall and specificity: results of McNemar's test (SPSS). Explanation see text.

Trial	Rec. (S+U)	Rec. (S)	p-value	Decision	Spec. (S+U)	Spec. (S)	p-value	Decision
Trial 1	0.50	0.50	0.500	=	0.92	0.77	0.000	+
Trial 2	0.80	0.63	0.019	+	0.39	0.59	0.000	-
Trial 3	1.00	1.00		n.a.	0.99	0.77	0.000	+
Trial 4	0.33	0.33	0.500	=	0.97	0.97	0.500	=
Trial 5	1.00	1.00	0.500	=	0.92	0.77	0.000	+
Trial 6	0.82	0.74	0.090	=	0.53	0.59	0.022	-
Trial 7	0.80	0.65	0.028	+	0.40	0.57	0.000	-
Trial 8	1.00	0.94	0.500	=	0.91	0.95	0.001	-
Trial 9	1.00	1.00	0.500	=	0.89	0.29	0.000	+
Trial 10	0.00	0.00	0.500	=	1.00	1.00	0.500	=
Trial 11	1.00	1.00		n.a.	0.97	0.22	0.000	+
Trial 12	0.89	1.00	0.500	=	0.73	0.34	0.000	+
Trial 13	0.75	0.75	0.500	=	0.87	0.78	0.000	+
Trial 14	0.91	0.91	0.500	=	0.40	0.44	0.170	=
Trial 15	0.40	0.60	0.500	=	0.87	0.69	0.000	+
Trial 16	0.43	0.43	0.500	=	0.99	0.99	1.000	=

We evaluate the experiment by conducting semi-structured qualitative interviews (i.e., interviews with open questions that allow for dialogue and discussion) with the users. First results of the ongoing evaluation phase are:

– Recall: Compared to the previous procedure of the trial team (reviewing new cases in the clinic information system), the CRDW shows a high recall rate. Users did not detect any patients matching a clinical study that the system had missed.
– Precision: The list of matching cases generated by the CRDW is rather unspecific. This is due to the fact that some exclusion criteria cannot be checked automatically, since the corresponding data is ambiguous or is not available in a digital format at all. Nevertheless, the system reliably sorts out cases with neurological diagnoses other than epilepsy. It also sorts out cases with types of epilepsy and/or types of seizures that are excluded by the trial protocols.
– Term extraction: The system recognizes essential synonyms and hyponyms. In almost all cases, the various types of epilepsy and epileptic seizures as well as the names of drugs and drug agents were correctly recognized.

Note that all results are still preliminary and will have to be substantiated by future experiments.

5 Conclusion

In this paper, we described a case study in using ontologies for information extraction from clinical documents. We demonstrated that we managed to build a system with a high sensitivity – a requirement for the task of patient recruitment. Improving precision, however, is still an issue. Future work will focus on the elimination of false positives by allowing logically more complex criteria.

The experimental data suggest that the process of patient identification benefits from extracting facts from structured data. We are planning to obtain more reliable results by considering more patients and trials. Moreover, the software will be tested by other departments, too.

A lesson learned in the area of ontologies is that it can be much easier to construct an ontology for a specific application instead of building or even using a general-purpose ontology. However, we also feel that the lack of German language resources hinders progress in the domain of semantic technologies suitable for German text and web resources.

Acknowledgments. The project is partially funded by TSB Technologiestiftung Berlin, Zukunftsfonds Berlin, and co-financed by the European Union – European Fund for Regional Development.

We would like to thank all computer scientists, linguists, ontologists, medical doctors, study nurses, and administrative staff who participated in the development of the software.

References

1. Bodenreider, O.: The unified medical language system (umls): integrating biomedical terminology. Nucleic Acids Research 32(Database-Issue), 267–270 (2004)
2. Browne, P.: Jboss Drools Business Rules. From technologies to solutions. Packt Publishing, Limited (2009)
3. Chapman, W.W., Hilert, D., Velupillai, S., Kvist, M., Skeppsted, M., Chapman, B.E., Conway, M., Tharp, M., Mowery, D.L., Deleger, L.: Extending the negex lexicon for multiple languages. In: Proceedings of the 14th World Congress on Medical and Health Informatics, MEDINFO 2013 (2013)
4. Cowie, J., Wilks, Y.: Information extraction. In: Handbook of Natural Language Processing, pp. 241–260 (2000)
5. Cunningham, H., Tablan, V., Roberts, A., Bontcheva, K.: Getting more out of biomedical documents with gate's full lifecycle open source text analytics
6. Dugas, M., Lange, M., Berdel, W., Müller-Tidow, C.: Workflow to improve patient recruitment for clinical trials within hospital information systems - a case-study. Trials 9(1), 2 (2008)
7. Gallaire, H., Minker, J., Nicolas, J.-M.: Logic and databases: A deductive approach. ACM Comput. Surv. 16(2), 153–185 (1984)
8. Jurafsky, D., Martin, J.H.: Speech and Language Processing, 2nd edn. Prentice Hall Series in Artificial Intelligence. Prentice Hall (May 2008)
9. Kifer, M.: Rule interchange format: The framework. In: Calvanese, D., Lausen, G. (eds.) RR 2008. LNCS, vol. 5341, pp. 1–11. Springer, Heidelberg (2008)

10. Kifer, M., Lausen, G., Wu, J.: Logical foundations of object-oriented and frame-based languages. Journal of the ACM 42(4), 741–843 (1995)
11. Lloyd, J.W.: Foundations of Logic Programming, 2nd edn. Springer (1987)
12. Müller, F.: A finite-state approach to shallow parsing and grammatical functions annotation of German. PhD thesis, University of Tübingen (2005)
13. Murphy, S.N., Mendis, M.E., Berkowitz, D.A., Kohane, I., Chueh, H.: Integration of clinical and genetic data in the i2b2 architecture. In: AMIA Annu. Symp. Proc., p. 2009 (2006)
14. Polleres, A.: From SPARQL to rules (and back). In: Williamson, C.L., Zurko, M.E., Patel-Schneider, P.F., Shenoy, P.J. (eds.) WWW, pp. 787–796. ACM (2007)
15. Reeve, L.: Survey of semantic annotation platforms. In: Proceedings of the 2005 ACM Symposium on Applied Computing, pp. 1634–1638. ACM Press (2005)
16. Rogers, F.B.: Medical subject headings. Bull. Med. Libr. Assoc. 51, 114–116 (1963)
17. Rosse, C., Mejino, J.V.L.: A reference ontology for biomedical informatics: the foundational model of anatomy. J. Biomed. Inform. 36, 478–500 (2003)
18. Ruch, P., Gobeill, J., Lovis, C., Geissbühler, A.: Automatic medical encoding with SNOMED categories. BMC Medical Informatics and Making 8, 6 (2008)
19. Russell, S.J., Norvig, P.: Artificial Intelligence: A Modern Approach, 2nd edn. Pearson Education (2003)
20. Scheitz, J.F., Mochmann, H.C., Witzenbichler, B., Fiebach, B., Audebert, H.J., Nolte, C.H.: J. Neurol. 25 (2012)
21. Scheitz, J.F., Mochmann, H.C., Nolte, C.H., Haeusler, K.G., Audebert, H.J., Heuschmann, P.U., Laufs, U., Witzenbichler, B., Schultheiss, H.P., Endres, M.: Troponin elevation in acute ischemic stroke (TRELAS) – protocol of a prospective observational trial. M. BMC Neurol. 11(98) (2011)
22. Schwaber, K., Beedle, M.: Agile Software Development with Scrum, 1st edn. Prentice Hall PTR, Upper Saddle River (2001)
23. Shvaiko, P., Euzenat, J.: Ontology matching: State of the art and future challenges. IEEE TKDE 99 (2011)
24. Staab, S., Studer, R.: Handbook on Ontologies, 2nd edn. Springer (2009)
25. Szarvas, G., Farkas, R., Busa-Fekete, R.: Research paper: State-of-the-art anonymization of medical records using an iterative machine learning framework. JAMIA 14(5), 574–580 (2007)
26. Todorov, K., Geibel, P., Kühnberger, K.-U.: Mining concept similarities for heterogeneous ontologies. In: Perner, P. (ed.) ICDM 2010. LNCS, vol. 6171, pp. 86–100. Springer, Heidelberg (2010)
27. Yu, L.: A Developers Guide the Semantic Web. Springer (2011)

Can Ontologies Systematically Help in the Design of Domain-Specific Visual Languages?

Maria das Graças da Silva Teixeira, Ricardo de Almeida Falbo, and Giancarlo Guizzardi

Ontology and Conceptual Modeling Research Group (NEMO),
Federal University of Espírito Santo, Vitoria/ES, Brazil
maria.teixeira@ufes.br, {gguizzardi,falbo}@inf.ufes.br

Abstract. There has been a growing interest in Domain-Specific Visual Modeling Languages (DSVL) and their support for domain understanding and communication. However, the quality of these languages fundamentally depends on how well their structure reflects the structure of the abstractions constituting the underlying domain conceptualization. Since a well-founded domain ontology aims at faithfully representing a domain, it can be seen the ideal input for engineering a DSVL. In this paper, we present an experiment that analyses the performance of computer students in interpreting instance models by varying the concrete syntax of the language used. We contrast a generic notation (UML-based notation for object diagrams) and a domain specific notation that was designed based on a well-founded ontology for the domain of organizational structures. The hypothesis is that the performance of participants in interpreting the models using the domain specific notation is better than those who do it through a generic notation. Performance is evaluated by taking response time and correctness of the answers into account. The results confirm, but also contradict the hypothesis initially formulated.

Keywords: Domain-Specific Visual Language, Ontology, Empirical Study, Conceptual Modeling.

1 Introduction

Conceptual Modeling is a fundamental activity to many areas in Computer Science, such as Ontology Engineering, Knowledge Representation and Software Engineering. In his seminal paper [1], Mylopoulos defines Conceptual Modeling as the activity of formally describing in diagrammatic notation aspects of the physical and social world for the purposes of understanding, communication and problem-solving. In conceptual modeling, visual notations play a key role, especially in communicating with domain experts [2] and in supporting more effective understanding and recall of domain models [3].

Recently, there has been a growing interest in the design and use of Domain-Specific Visual Modeling Languages (DSVL). According to [4], a DSVL follows the domain abstractions and semantics, allowing developers to perceive themselves as working directly with domain concepts, leading to major improvements in productivity, time-to-market responsiveness and training time.

Gurr [5] argues that the stronger the match between a conceptualization of a domain and its representing model, the easier it is to reason with the latter. The interpretation of a diagram by a person correlates precisely and uniquely with the conceptualization being represented. A direct consequence of this fact is that the more we know about the subject domain being represented, the better we can systematically exploit its properties in the design of a DSVL for that domain. To put in language engineering terms, the quality of a DSVL depends fundamentally on the quality of the representation of the subject domain that is used as input. As argued by [3]: "*cognitive effectiveness is not an intrinsic property of visual representations but something that must be designed into them. [...] There can be huge differences between effective and ineffective diagrams and ineffective diagrams can be less effective than text*".

Domain ontologies, in Computer and Information Science, are taken as explicit and formal representations of domain conceptualizations. In its original meaning, ontologies are supposed to be *reference models* of a given domain. In [6], Guizzardi proposed a language engineering framework that advocates the use of high-quality domain ontologies as the ideal representation for a subject domain and, hence, the ideal input conceptual model for the design of a DSVL. According to him, by making full use of a system of formal ontological meta-properties in the representation of the domain ontology at hand, the visual concrete syntax of a DSVL can be systematically designed to increase its cognitive and pragmatic efficiency. Thus, high-quality domain ontologies can produce better domain-specific languages, because they ensure a proper representation for their subject domain, making explicit a number of ontological meta-properties of domain concepts that can be employed in the design of a system of concrete syntax [6].

However, for a body of knowledge to be considered scientific, its truth and validity must be proven. A particular item of knowledge is considered to be scientifically valid if it has been checked against reality [7]. Thus, for evaluating the language engineering framework proposed in [6], we performed an empirical study, which is reported in this paper. The *goal* of this study is to compare a DSVL that was designed based on a well-founded ontology for the domain of organizational structures against a generic notation (UML-based notation for object diagrams). The *research hypothesis* is that the performance of the participants in interpreting instance models using the domain-specific notation is better than that made by participants interpreting instance models written in the generic notation. Performance is evaluated considering two measures: response time and correctness of the answers. The *subjects* are Computer Science students that have some experience in conceptual modeling. The results contain indications that the hypothesis is only partially valid, since we also obtained unexpected results, pointing to the need for further studies.

The remainder of this paper is organized as follows. Section 2 discusses the relationship between ontology and visual language concrete syntax. Section 3 presents a fragment of the ontology for organizational structures and discusses how this ontology and its meta-properties were exploited in the design of a DSVL. Section 4 presents the empirical study itself. Section 5 presents our final considerations.

2 Ontology and Visual Concrete Syntax

As discussed in [6], the appropriateness of an ontology can be engineered by guaranteeing an isomorphism between a concrete representation of a subject domain (a reference ontology for the domain) and a concrete representation of a language system (a language metamodel), i.e., the mapping between the ontological distinctions countenanced by the domain ontology and the modeling primitives constituting the language metamodel should be a one-to-one mapping. If this isomorphism is broken, we have cases such: (i) there are elements in the domain that cannot be expressed using the language, hurting language expressivity and domain appropriateness; (ii) there are elements in the domain that are expressed by more than one element in the language, introducing ambiguity and hurting clarity and comprehensibility appropriateness; (iii) there are elements in the language that don´t have an interpretation, hurting comprehensibility; and (iv) there are elements in the domain that are represented by more than one element in the language, hurting simplicity and interpretation.

In line with [6], this isomorphic mapping is defined only between ontology and abstract syntax (the modeling capabilities of the language). Moody [3] makes an analogous claim regarding the relation between concrete syntax and abstract syntax and, indirectly, the relationship between concrete syntax and ontology. This idea is encoded in the quality principle for visual notations that he terms ***semiotic clarity***. By discussing semiotic clarity, Moody draws from Nelson Goodman's *Theory of Symbols* when advocating *"for a notation to satisfy the requirements of a notational system, there should be a one-to-one correspondence between symbols and their referent concepts"* [7]. Here, once more, when the isomorphism is broken, the following anomalies occur [3]: (a) *Symbol redundancy* exists when multiple symbols are used to represent the same semantic construct; (b) *Symbol overload* exists when the same graphical symbol is used to represent different semantic constructs; (c) *Symbol excess* exists when graphical symbols are used but they do not represent any semantic construct; (d) *Symbol deficit* exists when semantic constructs are not represented by any graphical symbol.

Given that the suitability of a visual notation is evaluated with respect to a domain ontology, semiotic clarity of a system of visual syntax essentially depends on the characteristics of the underlying domain represented in that domain ontology. One aspect, however, which is not evident in the framework proposed by Moody, is the following. Unlike in the case of textual syntaxes, the graphical symbols that form the system of concrete syntax often fall naturally into a hierarchical typing, which informs about the semantics of what is being represented. An analogous statement can be made regarding certain relations between graphical symbols (e.g., spatial relations) that can be systematically mapped onto semantic relations with equivalent logical properties. These features of graphical symbol systems and relations are illustrated by the example in Fig. 1.

As one can notice, the models of Fig. 1.(a) and 1.(b) are isomorphic. The different concrete kinds of entities in the model (Federal Capital, State, Metropolis and Town) are represented by different kinds of geometrical objects (Square, Non-Squared Rectangle, Large Circle and Small Circle). In particular, the taxonomic structure of

Geopolitical Units is isomorphic to the one of Geometric Figures. For this reason, in a visual query one can immediately notice that Federal Capital is more similar to a State than to a City, and probably shares a common super-type with the former. Notice that, if we had produced a different taxonomic structure in the model of Fig. 1.(a), then a different choice of representing graphical symbols would have been made, possibly creating undesired implicatures for the model reader [6]. Given the difficulties experienced by modelers in the design of domain taxonomic structures [8], this illustrates the importance of having a well-designed ontology for the design of semiotic clarity in the system of visual syntax.

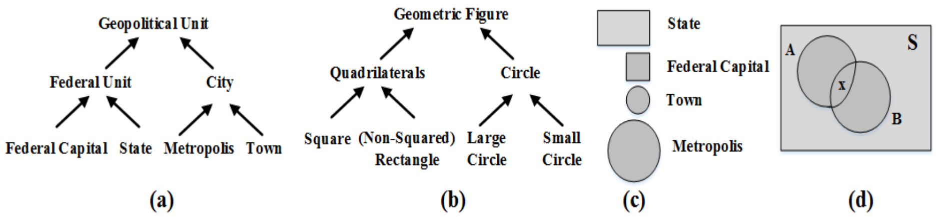


Fig. 1. (a) A fragment of a taxonomy for the geopolitical domain; (b) a taxonomy of geometric objects isomorphic to the structure in (a); (c) a system of visual symbols from (b) to represent the domain concepts in (a); (d) a representation of parthood relations between a state S, the cities A and B and one of its common part x

There is an additional point worth mentioning about this example. Although City and Metropolis/Town are examples of classifiers (classes, types), they classify their instances in very different ways with respect to modality. City is what is termed *essential* classifier, i.e., it *classifies* its instances necessarily (in the modal sense). In contrast, Metropolis/Town is a *contingent* classifier and classifies its instances only *accidently* (once more, in the modal sense). In other words, while instances of city are always instances of city and cannot cease to be so without ceasing to exist, the same city can be considered a town in a world w and a metropolis in w' while still maintaining its cross-world identity (cities change from town to metropolis given the size of their population). Likewise, in Fig. 1.(b), the color property of a geometric figure is considered one of its contingent intrinsic properties. Thus, a particular circular form is assumed to be able to change its color while maintaining a continuous visual percept. Furthermore, the intrinsic property population size that motivates the change in classification of cities is associated with a linearly ordered dimension. For this reason, we have decided to associate the intrinsic property of circles that represent this classifier variation by employing also a linearly ordered dimension (size).

In Fig. 1.(d), we have a representation of two cities A and B, which are part of a state S. By looking at this model, one can immediately infer that x, which is a part of city A (and B) is also a part of S (let us suppose x a neighborhood which is jointly managed by the two cities) . Moreover, one can immediately infer that x is a shared part of cities A and B. These inferences are termed *inferential free rides* [9], as they are considered to be costless from a cognitive point of view. These inferential free rides are present in this model because of the relation used to represent parthood, namely, the spatial inclusion in the plane. For instance, it is because spatial inclusion is a transitive relation that we can automatically derive (by transitivity) that x is part of S [6].

Although authors such as Moody and Gurr explicitly acknowledge the influence of suitable domain representations for the design of modeling languages, they do not properly elaborate on how these high-quality domain representations should be produced. Gurr simply alludes to the representation of domain models as algebraic structures, without offering any methodological and engineering support for their construction. Moody explicitly acknowledges the role of ontologies in the representation of semantic domains, but offers no discussion what so ever on how these semantic domain representations are created.

In a number of works [10][11], we have demonstrated the importance of foundational theories (i.e., domain-independent formal ontological theories) for analyzing and (re)designing domain reference ontologies. In particular, in the approach presented in [6], a philosophically and cognitively well-founded ontology representation language is used for creating domain ontologies, which, in turn, are used as input for designing domain-specific visual languages. This ontology representation language (termed OntoUML [12]) incorporates as modeling primitives, a number of formal meta-properties put forth by the Unified Foundational Ontology (UFO). UFO is a foundational theory incorporating a number of results from formal ontology in philosophy, philosophical logic, linguistics and cognitive science. For a full account of UFO, its formal characterization and empirical support, one should refer to [12][13].

A direct contribution of OntoUML to the design of *Domain-Specific Languages* is the following. Given that OntoUML incorporates in its metamodel the axiomatization of UFO, the only grammatically correct domain models that can be produced in this language are ontologically consistent ones, i.e., domain models that are consistent with UFO's basic axiomatization. Another contribution, however, is that, contrary to the merely formal structures employed in [5], the domain ontologies represented in OntoUML capture a number of ontological meta-properties that are used to further qualify the ontological status of domain concepts. We concentrate here on a fragment of OntoUML with a focus on *Object Types* and *Part-Whole* relations.

A fundamental modal meta-property used to distinguish among categories of *Object Types* is *rigidity* (and the associated notion of *anti-rigidity*). Formally, we have that [8]: a type T is rigid iff every instance of T is necessarily an instance of T (in the modal sense). In contrast, a type T' is anti-rigid iff for every instance x of T' there is a possible situation in which x is not an instance of T'. A stereotypical example that illustrates this distinction is the types Person and Student: instances of Person are necessarily so (Person is a rigid type); in opposition, instances of Student are merely contingently so (Student is an anti-rigid type).

Kinds and ***Subkinds*** are object types that are rigid [8]. These types define a taxonomy of rigid types instantiated by a given individual (kind being the unique topmost rigid type instantiated). Within the category of anti-rigid object types, we have a further distinction between ***Phases*** and ***Roles*** [8]. Both are specializations of rigid types. However, they are differentiated with respect to their *specialization conditions*. On Phases, the specialization condition is always an intrinsic one. For instance, a child is a Person whose age is in a certain range. On Roles their specialization condition is a relational one. For instance, a Student is a Person enrolled in a school.

A modal meta-property used to distinguish among the categories of Part-Whole relations is *existential dependence* [14]. An entity x is existentially dependent on another entity y iff in every situation that x exists then y must exist. Associated to existential dependence we have the notion of *generic dependence*. An entity x is generically dependent on a type Y iff in every situation where x exists an instance of Y must exist. These notions are used in UFO (among other things) to distinguish between part-whole relations that imply existential dependence and those that only imply generic dependence. A part-whole relation which implies only generic dependence from part to whole is named ***Parthood with Mandatory Wholes*** [14]. A part-whole relation which implies existential dependence from part to whole is termed ***Inseparable Parthood*** [14]. Another remark regarding part-whole relations worth mentioning is that contrary to purely formal mereological relations, part-whole relations which appear in conceptual models and material domain ontologies are non-transitive, i.e., they are transitive in certain situations and intransitive in others [15].

Finally, part-whole relations can be distinguished according to a meta-property named *shareability*. This meta-property wrongly defined in original UML specification has been refined in [12] with the following definition: (a) a (whole) type X is characterized by an exclusive (non-shareable) parthood relation with a (part) type Y iff every instance of X must have exactly one instance of Y as part; (b) a type X is characterized by a shareable parthood relation with a type Y iff instances of X can have more than one instance of Y as part.

3 From an Ontology of Organizational Structures to a Domain-Specific Visual Language for This Domain

Fig. 2 presents a small ontology of Organizational Structures. In this ontology, Employee is a role played by a Person when it is member of a Department. A Person (an abstract type) is either Man or Woman. An Employee is part of exactly one Department (represented by the non-shareable association end). Since this is a generic dependence relation, employees can change to different departments. An Employee is subordinated to at least one other employee who is its superior. Then, the types Subordinate Employee and Superior Employee are roles played by employees. As roles, an instance of Subordinate Employee can cease to be one, and for it to instantiate this role, there must exist another Employee instantiating the Superior Employee role. The same instance of Employee can simultaneously instantiate both roles.

A Department is part of exactly one Organizational Branch. Again, we have a case of a non-shareable parthood relation, but also one which implies existential dependency from part to whole (represented by the {inseparable} tag value), i.e., the Sales Department of an Organizational Branch can only exist as part of that branch. The relations between Employee and Department, and Department and Organizational Branch are cases of transitive parthood as identified in [15]. Commissions are collectives that have particular Employees as members (termed Commission Members).

Commissions can be in two different phases depending on the value of one of its intrinsic property (its amount of committed work). A Work-Overloaded Commission is a Commission such that its amount of committed work surpasses a certain threshold. A Normal Workload Commission is the complement of Commission with respect to Work-Overloaded Commission.

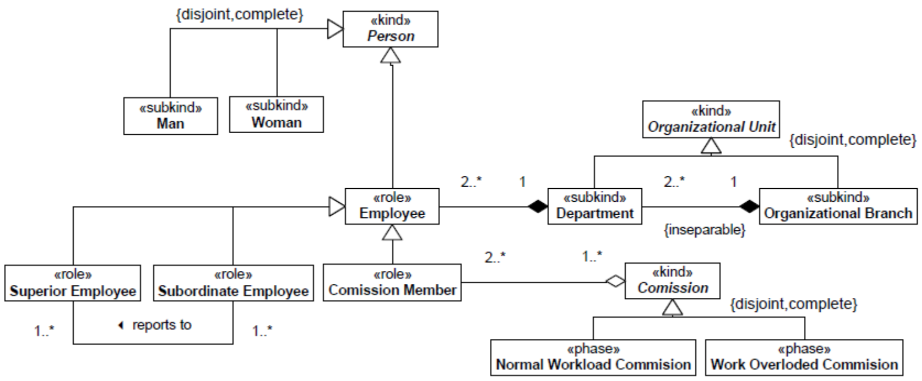


Fig. 2. A fragment of an ontology for organizational structures

Based on the ontology of Fig. 2, we have designed a domain-specific visual language aimed at representing valid instances of this ontology. This concrete syntax is the complement of the abstract syntax presented above, ideally being generated with the best that each of the theories presented in the previous section has to offer.

Table 1 presents the modeling primitives of this language via their respective concrete syntax. The table also relates these primitives with the domain concept they represent and with the ontological category of these domain concepts.

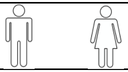
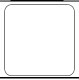
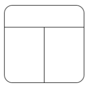
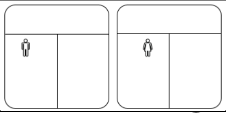
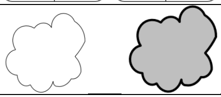
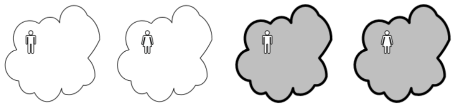
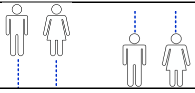
This concrete syntax presents semiotic clarity, i.e., there is an isomorphic mapping between the concepts in the domain ontology and the modeling primitives in the language. Moreover, the mapping between domain elements and elements in the visual notation takes full account of ontological categories and meta-properties of the former. Next, we elaborate on the systematic use of each of these ontological categories to derive properties of this system of concrete syntax.

Kinds and ***Subkinds***: In Fig. 2, we have both kinds and subkinds. As discussed in [16], shapes defined by closed contour are among the most basic metaphorical representations for objects. This idea is in line with a number of findings in cognitive science, including the one that shape plays a fundamental role in kind classification [17]. In the language defined in Table 1, each concrete subkind is associated with a shape. The chosen shapes are sufficiently dissimilar and are aligned with the taxonomic relations between domain types as presented in Fig. 2. For instance, the "four-sized" figures used to represent Organizational Branches and Departments are similar, considering they are organizational units. On the other hand, they are dissimilar from the blobs used to represent Commissions. These features highlight the characteristic of ***perceptual discriminability*** pointed by Moody and Hillegersberg [18].

Another aspect is the direct metaphorical resemblance between the graphical elements used and their referents. A case is the iconic representation for *Man* and

Woman. The representation of *Departments* as "pieces of an *Organizational Branch*" is adherent to the idea of "organizational divisions" associated to *Departments*. In addition, while the straight lines used in the contour of *Organizational Units* seems a more formal and rigid structure, the round boundaries of blobs representing *Commissions* are more naturally associated with a flexible informal one. The systematic use of these metaphorical resemblances brings to this notational system another important quality characteristic according to Moody, namely, ***perceptual immediacy*** [18].

Table 1. Visual concrete syntax for the organization structure ontology of Fig. 2

Domain Type	Ontological Category	Notational Element
Person, Organizational Unit, Commission	Kind	Abstract class; No direct representation
Man, Woman	Subkind	
Organizational Branch	Subkind	
Department and Department *is component of* Organizational Branch	Subkind and part-whole relation	
Employee and Employee *is component of* Department	Role and part-whole relation	
Normal Load Commission, Overloaded Commission	Phase	
Commission Member and Commission Member is part of Commission	Role and part-whole relation	
Superior and Subordinate Employees	Role	
Subordinate Employee *reports to* Superior Employee	Domain association	Combination of *is-dashed-line-connected* with the *above* relation in the plane

Phases: We used an intrinsic property of visual percept to represent different phases of a kind (the entity can change its phase but maintain its identity). In the language presented in Table 1, the changes in color of blobs used to represent *Commissions* represent different phases. We use a high-saturation color to represent the *Work-Overloaded Commission* exploring a metaphorical relation between "more quantity of color" and "more quantity of work". This feature increases its ***perceptual immediacy***. The difference in brightness of grey hue used to represent an overloaded commission and white one used to represent a regular load commission creates an efficient ***perceptual pop-out*** [18]. Finally, given that identifying overload commissions is an important task in the domain, the perceptual pop-out is increased by the increased ***perceptual discriminability*** between these two phases. This is due to the use of a different thickness of blobs boundaries. This is a case of ***redundant coding*** in [18].

<u>Relations</u>: In the ontology of Fig. 2, there are parthood relations between: (1) *Employee* and *Department*, and (2) *Department* and *Organizational Branch*. They are irreflexive and asymmetric. Moreover, transitivity holds across (1) and (2). By using the relation of spatial inclusion in the plane to represent these relations, we have a mapping to a visual relation that has exactly the same formal properties of the represented one, since spatial inclusion is also a partial order relation.

The different *Departments* that comprise an *Organizational Branch* are represented by a tessellation of the spatial region used to represent that branch. The lack of overlap between these regions allows for a ***perceptually immediate*** representation of the non-shareability metaproperty of these relations. This representation also contributes to ***perceptual immediacy*** due to the fact that, if *Departments* are represented as partitions of the region representing its associated *Organizational Branches*, this also favors the interpretation of existential dependence from part to whole.

Another parthood relation is between *Commission Member* and *Commission*. This relation is represented as a spatial containment relation between icons representing *Person* and a blob representing *Commission*. These blob forms can overlap with *Department* regions. This feature allows for direct inferential free ride on the identification of which *Department* a *Commission Member* belongs to. In addition, in line with the ***shareability*** metaproperty of this relation in the ontology, one can easily imagine overlapping blobs allowing for a certain member to be part of multiple commissions.

A third relation is the *reportsTo* relation, defined between a *Superior Employee* and its *Subordinates*. We used a combination of visual relations to represent this association (we combined the above relation in the plane with the transitive closure of the is-dashed-line-connected relation). Additionally, the different texture of this line increases the ***perceptual discriminability*** when contrasting it to the solid lines used to demarcate *Department* partitions. Finally, the spatial metaphor of using "higher in the plane" to represent "higher in the hierarchy" favors ***perceptual immediacy***.

<u>Roles</u> and ***<u>Relational Properties</u>***: Finally, we need visual representations able to highlight roles and their relational properties. The roles *Employee* and Commission Member are represented by the contained in the region relation between a person icon and a region representing a *Department* and a *Commission*, respectively The roles *Supervisor* and *Supervised by* are represented by a dotted line between two person icons and their spatial positioning.

4 The Empirical Study

In this section, we describe the empirical study performed. The *goal* of the experiment is to collect indications about the use of the concrete syntax of the domain-specific language presented previously. The *research hypothesis* is that the performance of participants in interpreting instance models using the domain-specific notation is better (according to response time and correctness of answers) than that made by participants interpreting instance models written in a generic notation. The empirical study was conducted following the guidelines presented in [19].

The experiment has qualitative and quantitative *strategies*. The experimentation *level* is in-vitro (it was conducted in a controlled environment). The research *approach* is primarily analytical, to collect early indications for further experiments. The experiment has as its *object of study* two instantiations of the conceptual model presented in Fig. 2. The instantiations were presented in two different notations, a domain-specific notation and a generic notation, giving rise to four instance models.

The *subjects* are Computer Science students, from both under-graduate and post graduate levels, which attend classes of a Conceptual Modeling course. The minimum requirement expected for participating in the experiment was having basic knowledge of UML. A questionnaire was applied to capture the participants' profile. Regarding the sample size, there were 22 participants. They were divided into two groups (A and B) randomly. Group A has 12 participants, and Group B has 10 participants. The imbalance in number of participants occurred, because, until the draw, we had 23 participants, and Group B (which had 11 participants) had one participant less for the effective execution of the activity.

The *factor* of the experiment is the concrete syntax of a visual language, and the *alternatives* are: a generic notation (UML-based notation for object diagrams) and a domain-specific notation (presented in Table 1). The *task* is the interpretation of instance models for the same instantiations, using different notations. Questions regarding two instantiations of the conceptual model presented in Fig. 2 were posed, varying the concrete syntax of the language used for representing them. The first instantiation represented using the domain-specific notation is depicted in Fig. 3. A semantically equivalent representation using the generic notation is shown in Fig. 4. Each participant had to answer two questions about this instantiation: one subjective (Q1), and other objective (Q2). Another instantiation, similar to the first one but bigger, was also used and other two questions (Q3 and Q4) were posed. Q3 is subjective, while Q4 is objective. The questions and the predefined answers (separated by fragments) about the first instantiation are presented next. Regarding the second instantiation, the Q3 has 6 answer fragments, and the Q4 has 4 answer fragments.

1. *Consider the individual Lisa. What information can be obtained about this individual from the observation of the model?* Template Answer: Lisa is part of the Marketing Department (fragment 1). She is a woman (fragment 2). She is supervised by Mary (fragment 3). She supervises Ana (fragment 4) and Otto (fragment5). She is member of the Quality Control Commission (fragment 6).
2. *Which is(are) the employee(s) with the largest number of direct subordinates*? *How many are the subordinates of this(these*) *employee(s)*? Template Answer: Peter (fragment 1) and Robert (fragment 2). They have three direct subordinates each (fragment 3).

The dependent variables are: response time and correctness of the answers. These variables are measured for each question. The way to analyze response time is trivial:

the time taken to answer each question is recorded and the smaller it is the time, the best is the related notation. Correctness is measured by comparing the fragments of the participants' answers to the corresponding fragments of the template. If they are the same, then the fragment is *correct*; otherwise the fragment is *wrong*, if the participant said something wrong about the fragment, or *missing*, if the fragment in the template is not reported by the participant. This is what we call *fragment correctness*. We have also other two types of fragments: wrong complementary fragment, which occurs when the participant included in her answer a fragment that the model does not say, related or not to the question; and extra complementary fragment, when the participant included in her answer a fragment that is not part of the template, but it can be inferred from the model. An extra complementary fragment is not a mistake, and therefore it is simply ignored.

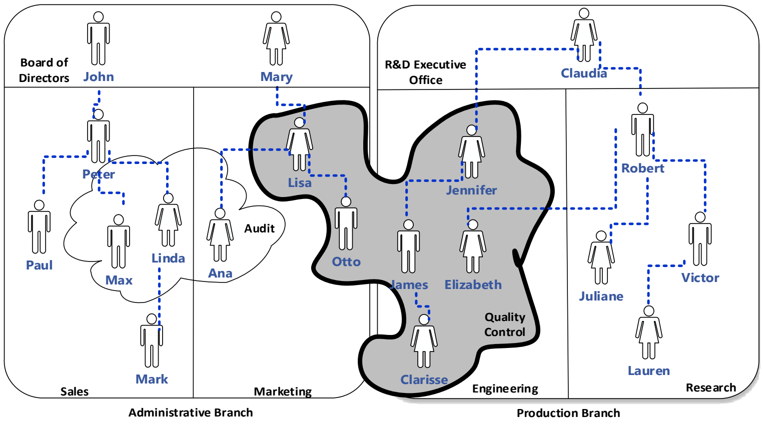


Fig. 3. An instance-model in the domain-specific language

To illustrate how we analyzed fragment correctness, consider the following answer given by a participant to question Q1: *"Lisa is part of the Board of Directors Department, and she is supervised by Mary. She is also part of the Marketing Department and she supervises Ana and Otto. She is part of the Quality Control Commission"*. According to the template, there are five correct fragments (1, 3, 4, 5 and 6), one missing (fragment 2), and one wrong complement (*Lisa is part of the Board of Directors Department*).

Besides interpreting the models, the participants filled in a questionnaire containing the following questions: (1) What is your impression (which one was easy/difficult, better/worse) among the models/concrete syntaxes presented? (2) Add any additional observation that you deem necessary.

Each group answered the same questions by interpreting each instantiation in a different notation. Group A answered the two questions (1 and 2) of the first instantiation in the domain-specific notation, while Group B answered the same questions of this instantiation using the generic notation. In the second instantiation, the situation was reversed, for answering questions 3 and 4, Group A interpreted the model written in the generic notation, while Group B interpreted the model written in the domain-specific notation.

In order to facilitate data collection, a website was developed. The site contained the instance models, the corresponding questions, and a link to the notation used in each instance. We recorded the participants' answer and the response time for each question automatically. Although we used a website, the experiment was conducted during a class in a lab, in order to ensure a stable Internet connection and to avoid distractions to participants, thereby reducing threats to the experiment.

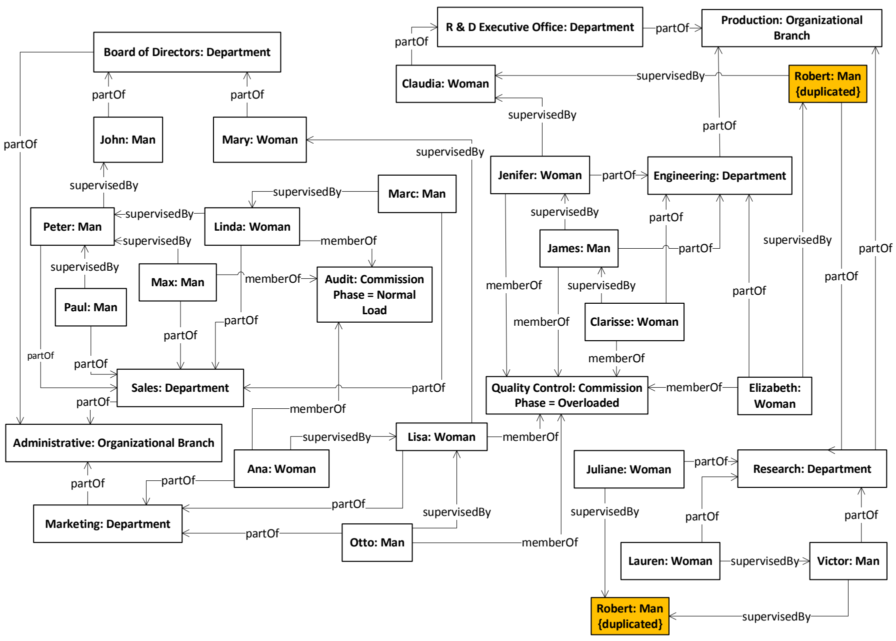


Fig. 4. An instance-model in the generic language

4.1 Collected Data

Regarding the participants' profile, we can say that: (i) The educational level (undergraduate, master and doctoral students) of the two groups were balanced; (ii) Regarding experience time in conceptual modeling, Group A has around 90% of its members with experience above 1 year, while Group B has 80% of participants in this range. We consider that the groups were balanced, even if members of Group A have a little more experience. The participants' profile was of students acting as modelers with some level of knowledge in conceptual modeling, specifically on using UML.

Table 2 presents data regarding the response time for each question. The columns present data on average, median, highest and lowest value of response time, and the percentage difference between highest and lowest averages for a question. Table 3 shows the percentage of fragment correctness for each question, and also the number of wrong complementary fragments, grouped by notation. In tables 2 and 3 we highlighted in grey the items that are consistent with our hypothesis, and in black the ones that contradict our hypothesis. Fig. 5 presents four graphs showing the response times for each question, comparing the notations. The values are ordered. Fig. 6 shows eight graphs showing the number of correct, wrong, missing, and wrong complementary fragments, two per question, comparing the notations. The values are ordered by number of correct fragments.

Table 2. Response Time (in seconds)

Question	Average (av)		Median		Highest Value		Lowest Value		(Smallest av / Largest av)
	Gr. A	Gr. B	Gr. A	Gr. B	Gr. A	Gr. B	Gr. A	Gr. B	
Q1	363,25	301,67	346,5	292	715	463	148	141	83,05%
Q2	101,92	210,22	99	226	157	324	49	79	48,48%
Q3	400,67	271,67	319,5	260	1416	453	153	117	67,80%
Q4	210,50	62,67	206,5	61	512	99	83	35	29,77%

Table 3. Percentage of correct, wrong and missing fragments, and number of wrong complementary fragments by notation

Question	% of Correct Fragments		% of Wrong Fragments		% of Missing Fragments		Number of Wrong Complementary Fragments	
	Specific	Generic	Specific	Generic	Specific	Generic	Specific	Generic
Q1	66,67%	85,00%	0,00%	3,33%	33,33%	11,671%	4	2
Q2	86,11%	90,0%	11,11%	3,33%	2,78%	6,67%	0	0
Q3	71,67%	88,89%	0,00%	0,00%	28,33%	11,11%	11	3
Q4	97,50%	75,00%	2,50%	12,50%	0,00%	12,50%	0	5

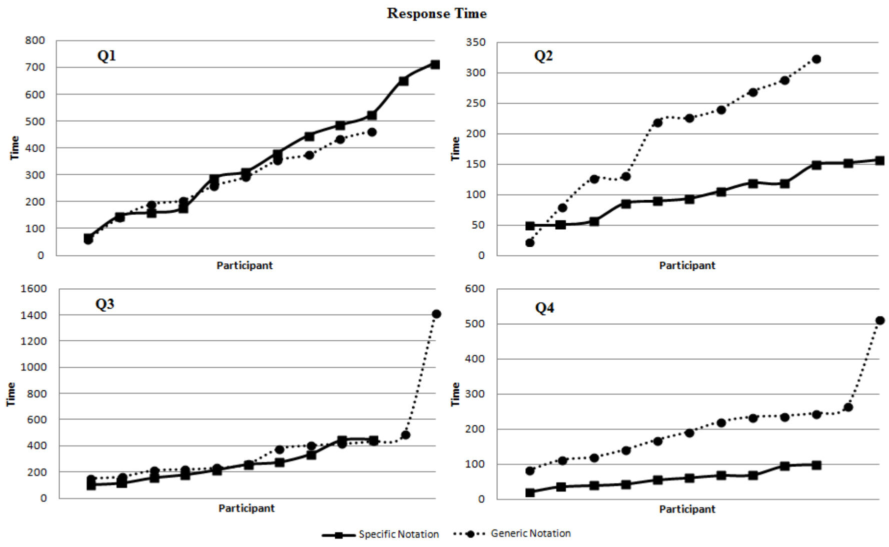


Fig. 5. Evolution of Response Time per Question for each Notation

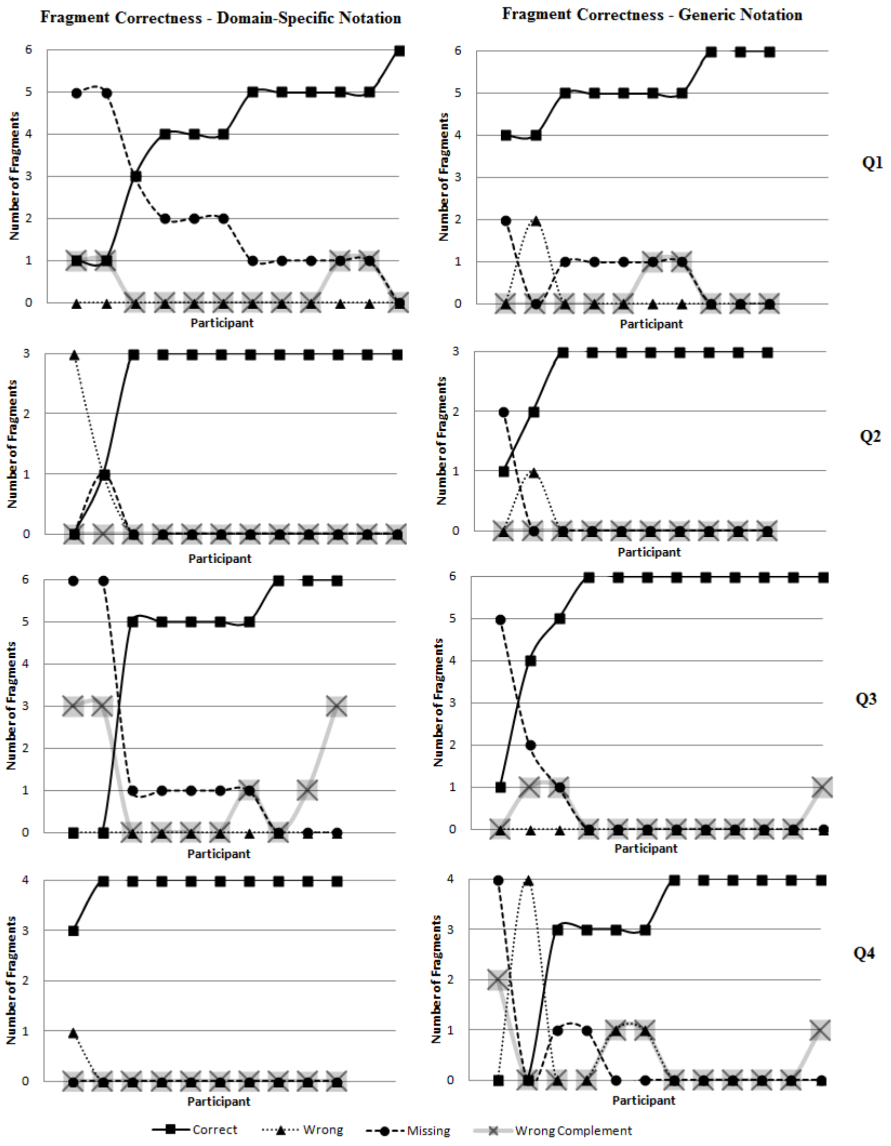


Fig. 6. Number of correct, wrong, missing and wrong complementary fragments per Question, for each Notation

4.2 Data Analysis and Discussion

According to our hypothesis, Group A should perform better on Q1 and Q2, while Group B should better perform on Q3 and Q4, as these are the times that each group works with the domain-specific notation. However, this was not always the case.

Looking at Table 2, we can say that, regarding response time, the expected results are confirmed for Q2, Q3 and Q4; however, the results contradicted our hypothesis for Q1. Moreover, the percentage differences between highest and lowest averages greatly varied. For Q1, highest and lowest averages had nearest values, while for Q4 they present the highest difference. The intention of generating such values is to observe how, on average, response times are different according to the notation. It is expected that the differences are significant, as occurred in Q2 and Q4, and even in Q3, favoring our hypothesis.

We have applied a statistical test, even having worked with a small sample. We applied the Wilcoxon-Mann-Whitney U Test [20], with significance level of 5%, for comparing response times. Considering groups A and B, U test indicated that the values are not significantly different, which is a good first indication that the groups are balanced. In Q1 and Q2, U test indicated that the values are not significantly different among groups, which was probably caused by the result of Q1. For Q3 and Q4, U test indicated that the values are significantly different among groups, what is a favorable result to our hypothesis.

However, an insulated analysis of response time is not enough. We need to check whether these results are corroborated by fragment correctness.

Looking at Table 3, which summarizes fragment correctness, we can notice that, again, we achieved results that are in favor and that contradict our hypothesis. Observing the number of correct fragments, we realize that specific notation worked better in Q4, but generic notation worked better in the other questions, with a small difference in Q2 (less than 1%). The most significant percentage of correctness occurred in Q4, and the lowest in Q1 (both from specific notation). Overall, the average correctness percentage is high (above 66%), demonstrating that participants had mostly success in the interpretation of models. On the number of wrong fragments, we had a slightly different result. In Q1 and Q4 we had fewer wrong fragments when using the specific notation, while in Q3 we had fewer wrong fragments when using the generic notation, in Q2 both had none error. Regarding missing fragments, the specific notation worked better for the objective questions (Q2 and Q4), while the generic notation worked better for the subjective questions (Q1 and Q3). Moreover, in the generic notation there is a relatively stable percentage (between 6% and 13%) of missing fragments. In the case of the specific notation, however, there is a stark difference when comparing objective and subjective questions. For objective questions, the percentages of missing fragments are very low (less than 3% for Q2, and 0% for Q4). On the other hand, for subjective questions, the percentages of missing fragments are very high (about 30% for Q1, and about 25% for Q3).

Finally, regarding the number of wrong complementary fragments, we can notice that, the specific notation worked better for the objective questions (Q4), while the generic notation worked better for the subjective questions (Q1 and Q3). We should also to highlight that, in Q2, there is not any wrong complementary fragment in both notations. Moreover, in the answers for Q3, when interpreting the model written in the specific notation, there is a high number of wrong complementary fragments.

Next, we complement our analysis presenting some detailed information for each question.

Question 1. The results obtained in this question clearly contradict our hypothesis. Q1 is a subjective question, requiring inspecting in details a model element. It requires a great attention by the participants, since it contains the greatest number of different fragments in the expected response. Response times were close, with a slight advantage for the generic notation. The missing parts stood out significantly in the domain specific notation. For instance, 10 participants did not indicate in the domain-specific notation that Lisa is a woman (83%), while only 6 participants in the generic notation did not indicate this fact (60%). Maybe it was considered obvious in the former notation, while in the case of the generic notation, perhaps the difficulty was lower, since this information is written in the model (allowing a textual perception). It is interesting to notice that, in fact, it was the number of missing fragments that caused the hypothesis contradiction, since the number of wrong fragments is favorable to the domain-specific notation. There were situations where the information was given only partially (counting as missing). For instance, there were an answer indicating that an employee supervises someone, but without indicating who is the supervised employee. Regarding the number of wrong complementary fragments, based on the answers, we suppose that some participants were not actually aware of the notation (commission treated as a kind of department or as a group).

Question 2. Q2 is an objective question. The domain-specific notation had a better performance regarding response time and the number of missing fragments. However, a better performance is achieved when using the generic notation with respect to the number of correct (a small difference) and wrong fragments. A possible explanation for this result came from the interpretation of a participant to this question: instead of answering who is the employee with the highest number of direct subordinates, she identified the employee with the greatest number of direct and indirect subordinates, giving rise to three wrong fragments. Once we worked with a small sample, this fact had a significant impact in the result. If we considered such participant an outlier, the result would have been reversed.

Question 3. Like Q1, this is a subjective question, and thus the results are similar: response times are close (with a small advantage for the domain-specific notation, as opposed to Q1), and there is a general advantage of the generic notation regarding correctness. Again, the number of missing fragments is able to change the outcome. In the specific notation, two participants indicated that there are 5 employees in the department, without indicating who are them (accounting for 10 missing fragments). In the generic notation, only one participant made this mistake. Moreover, in the specific notation, five participants did not indicate that the Marketing Department is part of the Administrative branch, while in the generic notation only one participant made this mistake. It is worthwhile to point out that the answers for this question presented the highest number of wrong complementary fragments in the experiment, highlighting the case of saying that Lisa is the leader of the Marketing Department (3 occurrences in generic notation and 5 occurrences in domain-specific notation).

Question 4. This was the question with the greatest proximity to our hypothesis. Specific notation presented better response time, higher percentage of correct fragments, and lower percentage of both wrong and missing fragments (in fact, just one error). This is a question that requires the participants realize which the members of a commission are, and then to which departments they belong. In the specific notation, this is easy to notice, since a person is within both the regions representing the commission and the department. On the other hand, in the generic notation, it is harder to follow the lines connecting the elements. Moreover, it is interesting to notice that the graph for this question when using the domain-specific notation is quite similar to the one for Q2, especially if we ignore the outlier in the latter.

In sum, the most prominent indication we noticed is that the participants using the domain-specific notation fared better on objective questions, while participants using the generic notation fared better in subjective questions. However, not all signs can be confirmed or justified. Nevertheless, we can say that the experiment fulfilled the objective of generating evidence in a qualitative way, being the starting point for subsequent experiments.

5 Final Considerations

In this paper we presented an empirical study aiming at collecting indications on the performance of participants in interpreting instance models, when using two different notations: a UML-based notation for object diagrams (said a generic notation), and a domain-specific visual language (DSVL), which was designed based on a well-founded ontology, following the approach discussed in [6]. In particular, we focus on the concrete syntax.

Some indications obtained from the results that need to be further explored: (i) Do the familiarity with the notation based on UML (generic one) may have assisted in the analysis of diagrams, closing, or even exceeding, the performance of the domain-specific notation, with which participants had the first contact? (ii) In the generic notation, all information about a given object are obtained in the same way: by navigating through links between objects, while with a DSVL, there are different ways of obtaining such information. How do this affect the results? (iii) Why in subjective questions is there a significant occurrence of missing fragments? Don't the participants perceive the information in the model, or at least do not feel the need to record the information, which can, for example, be considered "obvious"?

Future works will be directed to enhance indications identified here, trying to perform quantitative experiments, as well as deepening the knowledge regarding the unexpected data we collected. Some further experiments we foresee are: (i) to apply the experimental design discussed here in other domains; (ii) to apply the same experimental design but with participants with different profiles (novice modelers, model users). The first variation of the experiment aims at determining if the behavior is similar / different in a variety of domains. The second variation aims at identifying how the behavior is similar / different when we vary the participants' profile.

Acknowledgments. This research is funded by the Brazilian Research Funding Agencies FAPES (Process Number 52272362/11) and CNPq (Process Number #311578/2011-0) . We thank also to the participants of the experiment.

References

1. Mylopoulos, J.: Conceptual Modeling and Telos. In: Loucopoulos, P., Zicari, R. (eds.) Conceptual Modeling, Databases, and Case: An Integrated View of Information Sytems Development, pp. 49–68. Wiley (1992)
2. Avison, D.E., Fitzgerald, G.: Information Systems Development: Methodologies, Techniques and Tools. McGraw Hill, Oxford (2003)
3. Moody, D.L.: The "Physics" of Notations: Toward a Scientific Basis for Constructing Visual Notations in Software Engineering. IEEE TSE 35(6), 1–22 (2009)
4. Tolvanen, J.-P., Gray, J., Rossi, M.: Preface: A Special Issue on Domain-Specific Modeling with Visual Languages. Journal of Visual Languages and Computing (2004)
5. Gurr, C.A.: Effective Diagrammatic Communication: Syntactic, Semantic and Pragmatic Issues. Journal of Visual Languages and Computing 10(4), 317–342 (1999)
6. Guizzardi, G.: Ontology-Based Evaluation and Design of Visual Conceptual Modeling Languages. In: Reinhartz-Berger, I., Sturm, A., Clark, T., Cohen, S., Bettin, J. (eds.) Domain Engineering. Product Lines, Languages and Conceptual Models, p. 345. Springer, New York (2013)
7. Goodman, N.: Languages of Art: An Approach to a Theory of Symbols. Bobbs-Merril Co., Indianapolis (1968)
8. Guizzardi, G., Wagner, G., Guarino, N., van Sinderen, M.: An Ontologically Well-Founded Profile for UML Conceptual Models. In: Persson, A., Stirna, J. (eds.) CAiSE 2004. LNCS, vol. 3084, pp. 112–126. Springer, Heidelberg (2004)
9. Shimojima, A.: Operational Constraints in Diagrammatic Reasoning. In: Logical Reasoning with Diagrams, pp. 27–48 (1996)
10. Briguente, A.C., Falbo, R.A., Guizzardi, G.: Using a Foundational Ontology for Reengineering a Software Process Ontology. Journal of Information and Data Management (2012)
11. Gonçalves, B., Guizzardi, G., Filho, J.G.P.: Using an ECG Reference Ontology for Semantic Interoperability. Journal of Biomedical Informatics 43 (2010)
12. Guizzardi, G.: Ontological Foundations for Structural Conceptual Models. Telematica Instituut, Enschede, The Netherlands (2005)
13. Guizzardi, G., Wagner, G.: Using the Unified Foundational Ontology (UFO) as a Foundation for General Conceptual Modeling Languages. In: Theory and Application of Ontology. Springer, Berlin (2010)
14. Guizzardi, G.: Modal Aspects of Object types and Part-Whole Relations and the *de re/de dicto* distinction. In: Krogstie, J., Opdahl, A.L., Sindre, G. (eds.) CAiSE 2007. LNCS, vol. 4495, pp. 5–20. Springer, Heidelberg (2007)
15. Guizzardi, G.: The Problem of Transitivity of Part-Whole Relations in Conceptual Modeling Revisited. In: van Eck, P., Gordijn, J., Wieringa, R. (eds.) CAiSE 2009. LNCS, vol. 5565, pp. 94–109. Springer, Heidelberg (2009)
16. Ware, C.: Visual Thinking for Design. Morgan Kaufmann (2008)
17. Tversky, B., Hemenway, K.: Objects, Parts and Categories. Journal of Experimental Psychology 113, 169–193 (1984)
18. Moody, D., van Hillegersberg, J.: Evaluating the Visual Syntax of UML: An Analysis of the Cognitive Effectiveness of the UML Family of Diagrams. In: Gašević, D., Lämmel, R., Van Wyk, E. (eds.) SLE 2008. LNCS, vol. 5452, pp. 16–34. Springer, Heidelberg (2009)
19. Juristo, N., Moreno, A.M.: Basics of Software Engineering Experimentation. Springer Publishing Company, Incorporated (2001)
20. Wade, A., Koutoumanou, E.: Statistics Research & Methodology, `https://epilab.ich.ucl.ac.uk/coursematerial/statistics/index.html`

A Data Space System for the Criminal Justice Chain

Jan van Dijk[1], Sunil Choenni[1], Erik Leertouwer[1],
Marco Spruit[2], and Sjaak Brinkkemper[2]

[1] Dutch Ministry of Security & Justice, Research & Documentation Centre (WODC),
Turfmarkt 147, 2511 DP The Hague, The Netherlands
{j.j.van.dijk,r.choenni,e.c.leertouwer}@minvenj.nl
[2] Department of Information and Computing Sciences, Utrecht University
P.O. Box 80.089, 3508 TB Utrecht, The Netherlands
{m.r.spruit,s.brinkkemper}@uu.nl

Abstract. In this paper we present the concepts and implementation of a data space system for the management of data from heterogeneous sources in the criminal justice field. Our system exploits domain knowledge in the field of justice to streamline and to relate the content of different databases in the chain. In our system, domain knowledge is encoded in a space manager layer. Furthermore, in this layer it is decided which databases should be used to answer a query. This decision is taken on the basis of the encoded domain knowledge, and the content of the databases and its quality.

Keywords: data space system, criminal justice chain, domain knowledge, chain management.

1 Introduction

While in many fields data from different databases are related via automated data integration concepts and techniques – such as schema mapping or tuple mapping approaches (see amongst others [1-3]), data in the field of justice are manually or semi-manually related. The fact that data are related in a (semi-)manual manner in this specific field has to do with the complexity that is required in understanding and interpretating the data ([4-6]). In our case relating data means to bring (aggregated) data together that pertain to a same real world entity and define how the data should be interpreted in the context of other real world entities [7]. Furthermore, a concept of primary/foreign key relation between different databases is often lacking. If such a relation does exist, privacy law and regulations may not allow us to use it. Therefore, relating data from different information systems (semi-)manually is an error-prone and tedious process.

In this paper, we present the concepts and implementation of a data space system for the management of data from heterogeneous databases in the criminal justice field. The term data space system was coined in [8], as an agile form of data integration using a “pay-as-you-go” approach. Meanwhile the concepts in [8] have been elaborated in different directions ([6], [9-10]) and several prototypes have been developed

[11-15]. Kalidien et al. [6] come up with a conceptual architecture for data space systems, while Hedeler et al. [10] present a functional model for data space management systems which can be seen as a first attempt to formalize data space operators. The efforts in [11-12] combine personal information, [13-14] focus on the combination of data from domain independent sources, and [15] supports semi-automatic data integration of data sources in the domain of life science. We have built upon the concepts introduced in [8] and [6].

The main goal of our system is to give a more complete look into the flows through the criminal justice chain so that policy makers can identify potential capacity problems in an early stage. Also, an important requirement of such a system is that it should be able to combine all kinds of data in the justice field such as registered data from chain partners, forecasts, and safety survey data.

The importance of managing data between different databases is in some sense recognized in the field of federated databases and data warehouses. In a data warehouse, data from different sources that pertain to a single entity is transformed and loaded. Entities that apparently pertain to the same real-world object are linked together by means of primary and foreign keys. In [16], it has been pointed out that non-key attribute values are often inconsistent, while this is less often the case for key attribute values. In [6] it is argued that developing a data warehouse is a costly effort, and the development of a data space system is not only cheaper but may be also more effective for heterogeneous data management, e.g. in the field of justice. Although our system is tailored for the Dutch criminal system, we note the notions and concepts of data space systems (see Section 2) may be used in fields where relationships between different databases are of crucial importance.

2 Defining Data Space System Concepts

While database management systems provide an extensive set of tools to manage data within a database, tools to manage data between different databases are lacking. There is a practical need for these tools by organizations that are embedded in a chain and especially organizations that are in charge of the control of the chain. Insight into the flows through the chain is required to identify bottlenecks and improvement of the chain. For these reasons, concepts, tools and techniques need to be developed to manage heterogeneous data from the different systems in the chain. This requires an extension of the concepts and techniques of database design. In general, database design primarily focuses on those data required to serve the information needs of a selected group of users, e.g. a department in an organization. In database design it is not a common practice to establish relationships between different databases. A database is considered an independent and autonomous unit, in which the relationships with other databases are neglected and therefore not specified. It is exactly this property for which a data space system primarily differs from a database system [6]. In a data space system, we distinguish three layers, see also Fig. 1: an interface layer

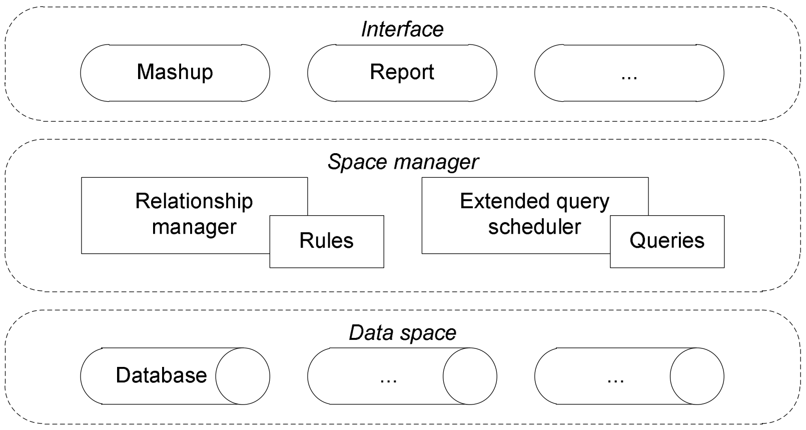


Fig. 1. Conceptual model of a data space system (Source: [6])

with querying and presentation functionality for end users; a space manager layer, containing a relationship manager with knowledge about the data in the data space and a query scheduler module; and a data space layer, consisting of a collection of databases.

In the data space layer, no constraints are put on the type of databases. The relation between different databases is encoded in the relationship manager of the space manager layer. The term "relation between databases" is defined in a broad sense. Two databases are related if they contain data about a same real-life entity or they contain data that may be used to establish a link between different entity types. For example, a criminal case brought to court leads to a verdict. Therefore, the number of cases brought to court and the number of cases with a verdict are related. So, the number of verdicts is less or equal to the criminal cases brought to court. This is a relationship between the database that records verdicts and the database that records criminal cases. The extended query scheduler module is used to determine which attributes are used to answer a query. In contrary to a query optimizer, the choice for the attributes is not based on trade-off options of (cpu and/or disk access) cost [17] but rather on the quality of the attributes [18]. The quality of the attributes is rated by domain experts and important factors that influence the rate are completeness and timeliness of the attribute data [19].

The interface layer is primarily meant to communicate with end users. Analogous to the terminology in database systems, the interface and the space manager layer may be regarded as data space management system. These layers adequately handle a query from an end user. Analogous to ANSI-SPARC architecture for the development of databases, the layers in the data space system are independent. For example, we allow making modifications at the space manager, while leaving the data space as it is. This property of independency contributes to the flexibility and extensibility of systems based on the data space system concepts.

3 Designing a Data Space System

Since the nature of the data in the criminal justice chain differs from administrative data ([5-6]), we summarize the characteristics of the data and relationships in the first section. In Sections 3.2 and 3.3, we discuss our design choices.

3.1 Data and Relationship Characteristics

Similar properties of a real-world event may be stored differently in the information systems of organizations involved in the criminal justice chain. For example, an event may be labeled by the police as a burglary with violence. However, the public prosecutor may come to the conclusion that violence will be hard to prove, therefore the same event is labeled as a burglary. Also, chain partners may use their own definition for seemingly similar data, depending on their core processes and management goals. Furthermore, chain partners register their data at different points in time, namely at the time when their involvement is required, and also at different precision levels, since detailed information relevant to one chain partner may not be relevant to another. For example, courts register the number of sentences imposed while the executing organizations register the number of sentences executed (some convicts cannot be found or die before the sentence can be executed). Courts register the type of crime because it is relevant for the sentence, but executing organizations may not record it as it has no relevance for the execution. Finally, we note that organizations at the beginning of the chain, such as the police, and at the end of the chain (the executing organizations) are primarily interested in data that pertain to individuals, while the remainder of the chain partners is rather interested in data about a case than an individual. In our system we take advantage of above-mentioned properties to relate (entities in) different databases.

To ensure the quality of the data, sets of rules are implemented in a separate relationship module. We distinguish between two types of relationships. The first type deals with missing data. If the value of an attribute is (temporarily) unavailable, for example due to technical problems, it may be replaced with the value of a similar attribute that is measured at a slightly earlier or later point in time, but more or less covers the same notion. This is called imputation. The imputed value may come from either the same or a different source. For instance, if the number of summonses is unknown, but we do know when the first court session took place, we may insert the value of the number of first sessions as an approximation for the number of summonses. In most cases these two attributes are highly correlated, because each summons eventually has to lead to a court session. Defining such relationships can help us to choose which data need to be selected when the preferred data are not available.

The second type of relationship deals with differences between seemingly similar but actually very different data. Different chain partners may label their data using a similar or even the same names, but because these data are measured on different points in time and based on their own criteria, the outcomes for these data will vary. Knowing the relationship will help us compare the data correctly. This problem is generally known as a semantic problem, and is difficult to detect [1]. To illustrate the second type of relationship, consider the number of community services agreed with the public prosecutor. These data come from two chain partners, namely the public prosecutor and the execution organization. Thus, the relationship rule for these data is that the number of community services registered by the executing organization should be lower or equal than the number registered by the public prosecutor, because suspects may die before execution or cannot be found. Historical data confirm this rule. The data are called stable when they meet this rule.

3.2 Relationship Rules

To construct relationship rules, domain knowledge is indispensable, and historical data can be used to confirm them. To apply the rules into the current data, there are two major practical challenges that need to be considered:

- The current data at hand may contain temporary missing data: this is understandable as organizations have different levels of capacity to handle data request and delivery, and thus, not all of them will be able to provide all data on time.
- The observed data may depend on other, possibly temporary factors: this is more difficult to detect especially if it pertains a change in policy. For instance, until recently the police had a so-called 'bonnenquota' (fine quota). As long as a high fine quota should be met, the police will quite easily giving out fines.

Considering these factors, it is suggested to define relationship rules in such a way that they will allow some violations. In other words, the rules should not be implemented in a strict manner (such rules are usually called soft rules).

In our application, we define the rule as a comparison between the ratio of the related data between two organizations and their average ratios. The previous data values are used to determine the average ratio. To stay as close as possible to the current situation, it seems justified not to use too much historical data. In our case, we apply the relationship rules to quarterly data and the average is calculated based on a series of quarterly data in the past two years. Violations of the rules in certain range of values will be allowed. The rules can be formally defined as follows.

$$(1-\beta)*\nabla_i \leq \frac{x_i^{(j),t}}{x_i^{(k),t}} \leq (1+\beta)*\nabla_i \text{, where} \tag{1}$$

i	= the related variable (data)
j,k	= data sources of this variable
$\frac{x_i^{(j),t}}{x_i^{(k),t}}$	= the ratio of variable *i* from source *j* and variable *i* from source *k*, at time period *t*
∇_i	= the average ratios of variable *i* from source *j* and variable *i* from source *k*, calculated at time periods *t*-1, *t*-2, ...
β	= margin for violation, $\beta \in (0,1]$

Using a margin of 10% (ß=0.1) means we allow the ratio of the present data to fall between 10% less than the average and 10% more than the average. Deciding which margin value will be reasonable requires some trial and error. Alternatively, one can observe a standard deviation calculated from historical data. In our case, most of the variables have a standard deviation around 0.1, but for some variables the standard deviation can vary up to 0.15. If we choose the largest standard deviation, it means we underestimate possible violations of the rules (or in other words: we allow the largest violation possible).

3.3 Variables

We have organized variables in hierarchical trees. For example, the database holds the following variables $v_1..v_6$: `{C-Cases-M-Verdict-Acquittal, C-Cases-M-Verdict-Conviction, C-Cases-C-Verdict-Acquittal, C-Cases-C-Verdict-Conviction, C-Cases-M-Verdict, C-Cases-C-Verdict}`. Fig. 2 depicts the tree for these variables. The variable trees are constructed bottom-up: the lowest (most specific) variables in the tree are generated directly from attributes, and upper variables are generated from lower variables.

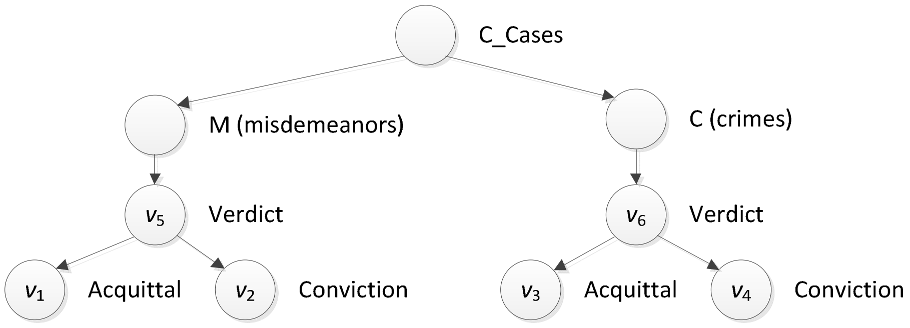


Fig. 2. Tree of variables that pertain to court cases

Variables are constructed dynamically by using naming conventions in the variable names. The name of a variable is built up from different parts, distinguished by an hyphen. Each part has its own meaning. So, the variable meaning can be constructed from its parts. The first and the second part of the variables (`C-Cases`) refer to that all variables pertain to court/trial cases. The variable v_1 pertains court cases that are misdemeanors (`M`) with 'acquittal' as verdict, while v_2 pertains to the same kind of cases, but with a 'conviction' as verdict. Analogous to misdemeanors, we have comparable variables v_3 and v_4 for crimes (`C`). The sum operator can not always be applied in a straightforward manner in a tree. This is the case whenever the attributes that are used to compute a variable are not disjunct. For instance, a verdict can have more than one 'verdict type'; an acquittal for one part of the crime, and conviction for another part. As a result, the total number of verdicts has to be extracted separately from the data space.

4 Architecture of a Data Space System

On the basis of design decisions, as discussed in Section 3, we discuss the architecture of the data space system. The architecture is presented in Fig. 3. The data space layer provides access to data sources from different organizations, which contain data of different aggregation levels. The space manager layer contains the knowledge and functionality to interpret the data, and, besides the variable database and the relationship module, consists of two auxiliary modules to create the variables. Data that are required to create a variable are extracted from the data space layer and converted to a standard format. Then, the converted data are manipulated in order to create the value of a variable and stored in the variable database. The interface layer actually presents the data to end users.

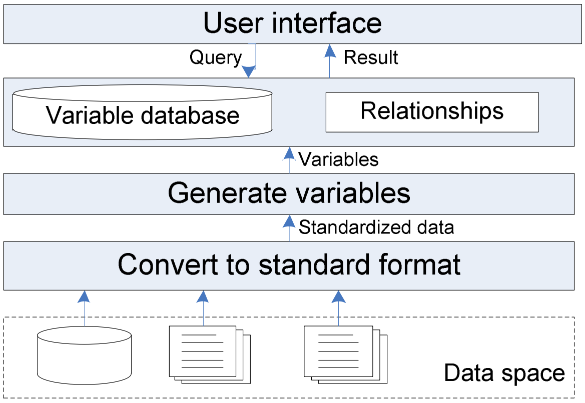


Fig. 3. Architecture of the data space system

We note that the data space layer is environment-dependent, and the interface layer is application-dependent. This means that the content of the data space is mainly determined by the environment in which the data space system is set up, whereas the way the interface is implemented depends on the requirements defined by the user. As a result, the implementation of the space manager is also application-dependent.

In this system, comparability is a key issue. Therefore, attribute data are extracted from the data space and stored in the space manager layer in a standardized format (analogous to the federated database approach [1]). On storage, attribute data is aggregated to three-month periods. The data are multi-dimensional and in this case, the dimensions are predefined: region, crime type, age, and person type (natural or legal person).

The Relationships component is a knowledge base about the variables in the data space system and their relationships. In the knowledge base the definition of variables and the relationships between variables are stored. As discussed in Section 3.3, the definition of a variable is stored in trees. Each phase in the criminal justice chain has one or more of these trees (e.g. the investigation phase has different trees for suspect and crime statistics). The knowledge base also contains the relationships rules (of the second type) between variables. Variables are generated from the attributes using the variable trees defined in the Relationships component. When an attribute is not available for one or more periods, these periods are not generated for variables that depend on this attribute. Special variables are the 'effect variables': an effect variable entails the margin of violation of a relationship rule. All variables are stored in the variable database for use in the interface.

5 Conclusions and Further Research

While database management systems provide an extensive set of tools to manage data within a database, tools to support the management of data between different databases are lacking. For several reasons, there is a need to support the management of data from heterogeneous information systems. For example, in the field of justice this might be to gain insight into the criminal justice chain by generating management information. To support the management of data from different heterogeneous systems, we have developed a data space system tailored to the Dutch criminal justice chain.

In our system we have exploited domain knowledge in the field of justice to streamline and to relate the content of the different data sources. The domain knowledge is encoded in the space manager layer. Furthermore, in this layer it is decided which databases should be used to answer a query. This decision is taken on the basis of the content of the databases and its quality. The system proves to be successful in managing data from different information systems that are involved in the criminal justice chain. Currently, the system is used by several chain partners as well as by policy makers at the Dutch Ministry of Security and Justice to gain insight into the flows through the criminal justice chain.

To our best knowledge, our data space system is the first implementation of the data space approach targeting the criminal justice chain. A topic for further research is the generalization of the data space concept by applying it in different contexts and domains. The data space concept will be further developed with a focus on data quality aspects. Another aspect of further research is the margin of violation used in the relationship rules. Learning from this case, we believe that for a good performance the margin should be made dynamically in two ways. Each variable may have its own margin of violation, and the margin of violation can change in time. Further research is necessary to determine how this can be implemented in a reliable and maintainable way.

References

1. Sheth, A.P., Larson, J.A.: Federated database systems for managing distributed, heterogeneous, and autonomous databases. ACM Computing Surveys 22(3), 183–236 (1990)
2. Cabibbo, L., Torlone, R.: Integrating heterogeneous multidimensional databases. In: Frew, J. (ed.) 17th International Conference on Scientific and Statistical Database Management, pp. 205–214. Lawrence Berkeley Laboratory, Berkeley (2005)

3. Winkler, W.E.: Matching and record linkage. In: Cox, B.G., Binder, D.A., Chinnappa, B.N. (eds.) Business Survey Methods, pp. 355–384. Wiley, New York (1995)
4. Wilkins, D., Pillaipakkamnatt, K.: The effectiveness of machine learning techniques for predicting time to case disposition. In: 6th International Conference on Artificial Intelligence and Law, pp. 106–113. ACM, New York (1997)
5. van den Braak, S., Choenni, S., Verwer, S.: Combining and Analyzing Judicial Databases. In: Custers, B., Calders, T., Schermer, B., Zarsky, T. (eds.) Discrimination & Privacy in the Information Society. SAPERE, vol. 3, pp. 191–206. Springer, Heidelberg (2013)
6. Kalidien, S.N., Choenni, R., Meijer, R.F.: Crime statistics online: potentials and challenges. In: 11th Annual International Digital Government Research Conference on Public Administration Online: Challenges and Opportunities (dg.o 2010), pp. 131–137. Digital Government Society of North America (2010)
7. Choenni, S., van Dijk, J., Leeuw, F.: Preserving privacy whilst integrating data: Applied to criminal justice. Information Polity 15(1,2), 125–138 (2010)
8. Franklin, M., Halevy, A., Maier, D.: From databases to dataspaces, a new abstraction for information management. SIGMOD Record 34(4), 27–33 (2005)
9. Hedeler, C., Belhajjame, K., Fernandes, A.A.A., Embury, S.M., Paton, N.W.: Dimensions of dataspaces. In: Sexton, A.P. (ed.) BNCOD 2009. LNCS, vol. 5588, pp. 55–66. Springer, Heidelberg (2009)
10. Hedeler, C., Fernandes, A.A.A., Belhajjame, K., Mao, L., Guo, C., Paton, N.W., Embury, S.M.: A functional model for dataspace management systems. In: Catania, B., Jain, L.C. (eds.) Advanced Query Processing. ISRL, vol. 36, pp. 305–341. Springer, Heidelberg (2012)
11. Blunschi, L., Dittrich, J.P., Girard, O.R., Karakashian, S.K., Salles, M.A.V.: A dataspace odyssey: The iMeMex personal dataspace management system. In: Proceedings of the Conference on Innovative Data Systems Research (CIDR), pp. 114–119 (2007)
12. Dong, X., Halevy, A.Y.: A platform for personal information management and integration. In: CIDR, pp. 119–130 (2005)
13. Das Sarma, A., Dong, X., Halevy, A.: Bootstrapping pay-as-you-go data integration systems. In: SIGMOD, pp. 861–874 (2008)
14. Madhavan, J., Cohen, S., Dong, X.L., Halevy, A.Y., Jeffery, S.R., Ko, D., Yu, C.: Web-scale data integration: You can afford to pay as you go. In: CIDR, pp. 342–350 (2007)
15. Leser, U., Naumann, F.: (Almost) hands-off information integration for the life sciences. In: CIDR, pp. 131–143 (2005)
16. Agarwal, S., Keller, A.M., Wiederhold, G., Saraswat, K.: Flexible Relation: An approach for integrating data from multiple, possibly inconsistent databases. In: 11th International Conference on Data Engineering, pp. 495–504. Stanford Univ., CA (1995)
17. Choenni, S., Blanken, H., Chang, T.: On the Selection of Secondary Indices in Relational Databases. Data & Knowledge Engineering 11(3), 207–233 (1995)
18. Berti-Équille, L.: Quality Awareness for Managing and Mining Data. Habilitation à diriger des recherches. L'Université de Rennes, France (2007)
19. Pipino, L.L., Lee, Y.W., Wang, R.Y.: Data quality assessment. Communications of the ACM - Supporting Community and Building Social Capital 45(4), 211–218 (2002)

Exploiting Social Tagging in Web API Search

Devis Bianchini, Valeria De Antonellis, and Michele Melchiori

Dept. of Information Engineering University of Brescia
Via Branze, 38 - 25123 Brescia Italy
{bianchin,deantone,melchior}@ing.unibs.it

Abstract. Given the huge number of available Web APIs, a web designer might take advantage of the "wisdom" or collective knowledge of the other developers who used the Web APIs for their own mashups. This knowledge may implicitly derive from the use of the Web APIs in similar mashups, or may be obtained through explicit social tagging and rating of Web APIs. In this paper, we propose to exploit this knowledge to implement advanced Web API search patterns, depending on the development scenario the web designer is acting in, namely the creation of a new mashup, the completion of an existing one or the substitution of one or more Web APIs within it.

1 Introduction

The exploitation of the "wisdom" or collective knowledge of the other developers to select the best Web APIs for developing a mashup is getting more and more relevant in the Web 2.0 [1,2]. Besides advanced Web API characterizations performed by combining categories and semantic tags [3] or technical features (e.g., protocols, data formats) [4], some researchers proposed to use the information about past mashups to let the designer of a new mashup "learn by examples" [2]. These approaches have the advantage of leveraging information from the Web API repositories. Nevertheless, the additional information brought by the so-called *social tagging* is something more: there is an implicit collective knowledge deriving from the co-occurrence of Web APIs in the same or similar mashups and an *explicit* social tagging and rating of Web APIs by developers who used them in the past.

In this paper, we exploit a new model for Web API search and ranking. The model includes a Web API characterization at different levels of complexity, namely (i) the Web API description in terms of categories and technical features, as extracted from public repositories, (ii) the aggregation of Web APIs into mashups performed by other developers (*implicit collective knowledge*), (iii) the semantic tagging and rating of Web APIs by developers who used them in their own mashups (*explicit collective knowledge*). In this paper, we are compliant with the `ProgrammableWeb` repository (`http://www.programmableweb.com/`), which offers a large, updated and highly populated catalogue of Web APIs to be selected and aggregated into mashups. Nevertheless, as we have underlined in [5], the search facilities of this repository reflect the ones of the (few) other available public registries. Semantic tagging is performed by relying on the system proposed in [3]. Specific similarity and ranking measures based on the model are described. Such metrics implement different Web API aggregation scenarios, namely the development of a new mashup, the completion of an existing mashup or the substitution of a Web API in a given mashup. With respect to the multi-perspective Web API model we proposed in [6], we stress here the distinction between the implicit and the explicit collective knowledge and we refine the measures for Web API search and ranking according to this distinction.

The strong point of our approach relies on the tuning of Web API matching and ranking according to different Web API features. In this sense, our approach is the first one that, taking into account so many aspects in Web API descriptions, combines them in a systematic way. Other approaches focus on a subset of such aspects, namely technical features [4] or collective knowledge coming from Web API use in existing web mashups [7], but they do not rely on other aspects such as ratings and designers' expertise. Exploitation of designers' ratings avoids Web API recommendation based only on the number of past mashups where it has been used, despite the inner quality of the Web API [2]. Moreover, our approach distinguishes among different Web API search targets (namely the creation of a new mashup, the completion of an existing one or the substitution of one or more Web APIs within it), performing a step forward with respect to approaches such as [8], which starts from a Web API model that is close to ours, but it is not designed to be differently tuned according to changing search targets.

The paper is organized as follows. Section 2 contains the definition of the Web API model we rely on. Metrics for Web API search and ranking are described in Section 3. Finally, Section 4 closes the paper.

2 Web API Model

The model we assume in this paper for Web API characterization is based on a representation that is commonly used for folksonomies [9], where user-based assignments of tags to resources are described. In this paper, users are Web designers, who assign both semantic tags and ratings to Web APIs as used in a given mashup. We formally define a *Web API repository* as follows.

Definition 1. *A Web API repository is a tuple* $\Re := \langle \Omega, \mathcal{M}, \mathcal{D}, \mathcal{S}, \Gamma, \Lambda, \mathcal{T}, \mu \rangle$, *where:*

- Ω, $\mathcal{M}$, $\mathcal{D}$, $\mathcal{S}$ *are finite sets, whose elements are the Web APIs, the mashups, the web designers and the semantic tags, respectively;*
- $\Gamma \subseteq \mathcal{M} \times \mathcal{D}$ *represents the set of ownerships of mashups by designers; specifically, if* $\langle m, d \rangle \in \Gamma$, *then the mashup* $m \in \mathcal{M}$ *is owned by the designer* $d \in \mathcal{D}$;
- $\Lambda \subseteq \Omega \times \mathcal{M}$ *is the set of pairs modeling the* `composedOf` *relationship between mashups and Web APIs; specifically, if* $\langle \mathcal{W}, m \rangle \in \Lambda$, *then the Web API* $\mathcal{W} \in \Omega$ *is included in the mashup* $m \in \mathcal{M}$;

– *$\mathcal{T}$ is a quaternary relation between the finite sets, that is, $\mathcal{T} \subseteq \Omega \times \mathcal{M} \times \mathcal{D} \times \mathcal{S}$, representing the assignment by designers of semantic tags to Web APIs with respect to a given mashup;*
– *μ is a function which represents a quantitative rating assigned by a designer $d \in \mathcal{D}$ to quantify how much the use of a Web API in Ω is suitable with respect to a given mashup $m \in \mathcal{M}$ which includes the Web API; the function is defined as $\mu : \Omega \times \mathcal{M} \times \mathcal{D} \mapsto [0,1]$.*

The model enables a designer to assign semantic tags and ratings also with respect to mashups owned by other designers; however, ownership of mashups by designers might be relevant. For instance, ratings assigned by the mashup owner should be considered as more reliable. Ownership of mashups is not currently exploited in our Web API search and rating metrics and will be investigated as future work.

In our model, a Web API $\mathcal{W} \in \Omega$ is described by: (i) a name $n_{\mathcal{W}}$; (ii) a unique $URI_{\mathcal{W}}$; (iii) a set of categories $C_{\mathcal{W}}$; (iv) a set $\{t_{\mathcal{W}}\}$ of semantic tags; (v) a set $\mathcal{F}_{\mathcal{W}}$ of technical features, where a feature is a pair ⟨`type`,{`values`}⟩ (e.g., ⟨`protocol`,{`SOAP`,`REST`}⟩).

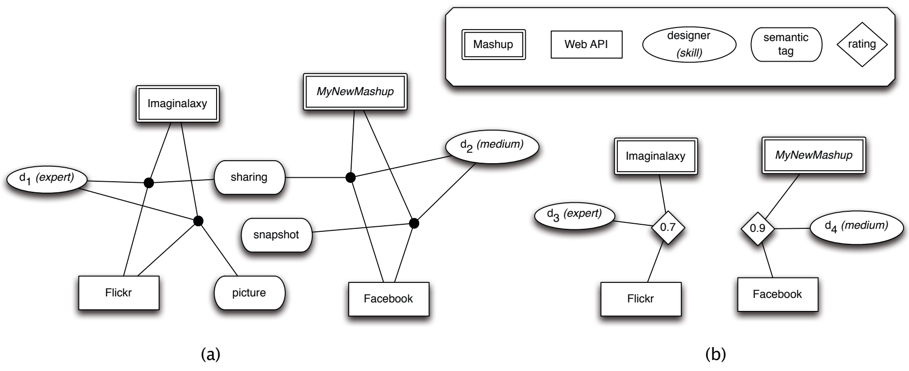
Each mashup $m \in \mathcal{M}$ is modeled through a name n_m and a Uniform Resource Identifier URI_m. No technical features are provided for mashups, since in public repositories these features are directly related to the component Web APIs.

Each designer $d_i \in \mathcal{D}$ is modeled through the skill $\sigma_i \in [0,1]$ for developing web applications. The skill is automatically assigned based on the number of mashups a designer owns (relation Γ), according to a threshold-based scale: 1.0 for `expert`, 0.8 for `high confidence`, 0.5 for `medium confidence`, 0.3 for `low confidence` and 0.0 for `unexperienced`. More sophisticated skill update functions will be investigated as future work.

A designer may assign two kinds of information about a Web API in the context of a mashup where it has been used: semantics tags and a rating score. Semantic tags are assigned with the support of the WordNet lexical system. A semantic tag in $\mathcal{S}$ is a triplet $t := \langle n_t, syn, d_t \rangle$, where: (i) n_t is the name of the tag itself; (ii) syn is the set of all the synonyms of t, extracted from WordNet; (iii) d_t is the human readable definition associated with the synset. The rating score μ is assigned by the designer according to the NHLBI 9-point Scoring System. See [3] for more details about semantic tags and rating score assignment.

Example. Let consider the mashup `Imaginalaxy`, tagged by an expert designer $d_1 \in \mathcal{D}$ and composed of the `Amazon S3`, `MySpace` and `Flickr` APIs. Now consider a fictious mashup `MyNewMashup`, developed and tagged by a medium-skilled designer $d_2 \in \mathcal{D}$, who used the `Amazon S3`, `MySpace` and `Facebook` APIs. The two mashups share the `Amazon S3` and `MySpace` APIs, while differ in the choice of the third API[1]. In Figure 1(a) we reported the semantic tags of the `Flickr` and

[1] In this paper, we adopt a toy example. In a real case scenario, all the mashups in the repository which are composed of, among the others, the `Flickr` and `Facebook` APIs should be considered for Web API search and ranking.



Tag name	Synonyms	Definition
sharing	{ }	using or enjoying something with others
picture	{photograph,exposure,photo,pic}	a representation of a person or scene in the form of a print or transparent slide or in digital format
snapshot	{snap,shot}	an informal photograph; usually made with a small hand-held camera

(c)

Fig. 1. A graphical representation of the semantic tag assignment (a) and rating of Web APIs within mashups (b) for the running example; (c) WordNet-based semantic tags definition

`Facebook` APIs assigned by d_1 and d_2, respectively. The expert designer tagged the `Flickr` API with `sharing` and `picture`, while d_2 tagged the `Facebook` API with `sharing` and `snapshot`. The definition of the semantic tags based on WordNet is given in Figure 1(c). Moreover, the expert designer assigned a rating score equal to 0.7 to the `Flickr` API used in the `Imaginalaxy` mashup, while d_2 assigned a higher score (0.9) to `Facebook` API used in `MyNewMashup`, as shown in Figure 1(b).

2.1 Web API Request and Search Targets

A Web API request is formulated depending on the search target the web designer is pursuing. A general definition of Web API request is the following:

$$\mathcal{R} = \langle C_{\mathcal{R}}, \{t_{\mathcal{R}}\}, \{\mathcal{W}_R\}, \mathcal{F}_{\mathcal{R}} \rangle \tag{1}$$

where: (i) $C_{\mathcal{R}}$ is the set of required Web API categories, to be matched against the categories $C_{\mathcal{W}}$ of each available Web API $\mathcal{W} \in \Omega$ in the repository $\Re$; (ii) $\{t_{\mathcal{R}}\}$ is a set of semantic tags specified for the Web API to search for (for each Web API $\mathcal{W} \in \Omega$, $\{t_{\mathcal{R}}\}$ is matched against the semantic tags assigned by the designers to $\mathcal{W}$ for each mashup where the Web API has been used); (iii) $\{\mathcal{W}_R\}$ is an optional set of Web APIs already included in a mashup that the designer aims at completing (if any) (for each Web API $\mathcal{W} \in \Omega$, $\{\mathcal{W}_R\}$ is matched against the Web APIs which compose the mashups where both $\{\mathcal{W}_R\}$ and $\mathcal{W}$ have been included); (iv) $\mathcal{F}_{\mathcal{R}}$ is an optional set of required technical features, among protocols, data formats and security features.

Not all the elements listed in Equation (1) are mandatory. Their specification and use within $\mathcal{R}$ depend on the *search target* pursued by the designer. We identified the following kinds of search targets:

– *single Web API selection*, when the designer is developing a new mashup and aims at finding a Web API by specifying desired Web API categories $C_{\mathcal{R}}$ and semantic tags $\{t_{\mathcal{R}}\}$; optionally, the designer may also specify the desired technical features $\mathcal{F}_{\mathcal{R}}$ for the Web API to search for;

– *mashup completion*, when a mashup is already available and the designer desires to add a new Web API; in this case, $\{\mathcal{W}_R\}$ is the set of Web APIs already included in the mashup under construction and the goal is to receive suggestions about other Web APIs $\mathcal{W}{\in}\Omega$ which have been used with $\{\mathcal{W}_R\}$ in the same mashups; moreover, if the designer does not explicitly specify $\mathcal{F}_{\mathcal{R}}$, it is automatically composed of the intersections of protocols, data formats and security features of the Web APIs in $\{\mathcal{W}_R\}$, respectively; in fact, in this specific case, the best solution is that all the Web APIs within the mashup share the same protocol, data format and security features;

– *proactive mashup completion*, it is a variant of the previous search target; in this case, the designer does not specify the categories $C_{\mathcal{R}}$ and the semantic tags $\{t_{\mathcal{R}}\}$ of the Web API to search for, but relies on suggestions of the system, which proposes candidate Web APIs based on collective knowledge coming from other existing mashups, which contain Web APIs close to $\{\mathcal{W}_{\mathcal{R}}\}$ and presenting technical features similar to $\mathcal{F}_{\mathcal{R}}$;

– *Web API substitution*, when the designer desires to substitute a Web API in an existing mashup; in this case, $C_{\mathcal{R}}$ and $\{t_{\mathcal{R}}\}$ are specified for the Web API to substitute; the construction of $\mathcal{F}_{\mathcal{R}}$ follows the same procedure of the (proactive) mashup completion.

3 Web API Search and Ranking Metrics

Web API search is performed through computation of similarity metrics. Similarity metrics are applied by combining the information available for Web APIs, namely their categories, technical features, semantic tags and mashups where the Web APIs have been included, considering the particular folksonomy-style model described in the previous section. The overall similarity measure between the request $\mathcal{R}$ and each API $\mathcal{W}{\in}\Omega$ in the repository $\Re$, denoted with $Sim(\mathcal{R},\mathcal{W})$, is computed as the following linear combination:

$$\begin{aligned} Sim(\mathcal{R},\mathcal{W}) = \omega_1 \cdot Sim_c(\mathcal{R},\mathcal{W}) \ + \ \omega_2 \cdot Sim_t(\mathcal{R},\mathcal{W}) \ + \\ \omega_3 \cdot Sim_{comp}(\mathcal{R},\mathcal{W}) \ + \ \textstyle\sum_{i=4}^{N} \omega_i \cdot \overline{Sim_i}(\mathcal{R},\mathcal{W}) \in [0,1] \end{aligned} \tag{2}$$

where $0{\le}\omega_i{\le}1$ and $\sum_{i=1}^{N} \omega_i = 1$ are weights to be set according to the search target, as summarized in Table 1, where the category similarity is weighted less than the other factors, since the category is only a coarse-grained entry point to look for Web APIs in the `ProgrammableWeb` repository. Specifically:

– $Sim_c(\mathcal{R},\mathcal{W}){\in}[0,1]$ is the similarity between the requested category and the category of $\mathcal{W}$;

Table 1. The setup of Web API similarity weights depending on the search target

Search target	**Request formulation**
Single Web API selection	$\omega_1, \omega_2, \omega_i (i = 4..N)$ equally weighted or $\omega_1 = 0.2$ and $\omega_2 = 0.8$ if $\mathcal{F}_\mathcal{R} = \emptyset$
Mashup completion	$\omega_1 = 0.1, \omega_2 = \omega_3 = \omega_i (i = 4..N)$
Proactive mashup completion	$\omega_3 = \omega_i (i = 4..N)$ or $\omega_3 = 1.0$ if $\mathcal{F}_\mathcal{R} = \emptyset$
Web API substitution	$\omega_1 = 0.1, \omega_2 = \omega_3 = \omega_i (i = 1..N)$

– $Sim_t(\mathcal{R}, \mathcal{W}) \in [0, 1]$ is the similarity according to semantic tags;
– $Sim_{comp}(\mathcal{R}, \mathcal{W}) \in [0, 1]$ is the combination of composition similarities between the mashup which contains the set of Web APIs $\{\mathcal{W}_R\}$, where the required one will be included, and each mashup which contains $\mathcal{W}$;
– $\overline{Sim_i}(\mathcal{R}, \mathcal{W}) \in [0, 1]$ is the similarity between $\mathcal{R}$ and $\mathcal{W}$ based on the i-th technical feature, such as protocols or data formats.

For what concerns $Sim_c(\cdot)$ and $Sim_t(\cdot)$ computation, we refer to the details provided in [3]. Here we will focus on the contributions of this paper on mashup composition similarity and Web API similarity based on technical features.

The mashup composition similarity between a mashup m_1 and another mashup m_2, denoted with $MashupSim(m_1, m_2)$, evaluates the number of common Web APIs in the two mashups, that is:

$$MashupSim(m_1, m_2) = \frac{2 \cdot |m_1 \cap m_2|}{|m_1| + |m_2|} \in [0, 1] \tag{3}$$

where $|m_1 \cap m_2|$ denotes the number of common Web APIs in the two mashups, while $|m_i|$ denotes the number of Web APIs in the i-th mashup. To compute the *mashup-based similarity* between the request $\mathcal{R}$ and each available Web API $\mathcal{W} \in \Omega$ in the repository $\Re$, denoted with $Sim_{comp}(\mathcal{R}, \mathcal{W})$, it should be considered that each mashup, where $\mathcal{W}$ has been included together with $\{\mathcal{W}_\mathcal{R}\}$, is developed by a designer $d_i \in \mathcal{D}$, who is characterized by a skill σ_i in web application development. Therefore, similarity with mashups developed by more expert designers should be weighted more than similarity with mashups developed by less expert designers. Therefore, the mashup-based Web API similarity is computed as follows:

$$Sim_{comp}(\mathcal{R}, \mathcal{W}) = \frac{1}{|\mathcal{D}_\mathcal{W}|} \cdot \sum_{i=1}^{|\mathcal{D}_\mathcal{W}|} \frac{\sum_{k=1}^{|M_i|} \sigma_i \cdot MashupSim(\{\mathcal{W}_\mathcal{R}\}, m_i^k)}{|M_i|} \in [0, 1] \tag{4}$$

where: (i) $\mathcal{D}_\mathcal{W} \subseteq \mathcal{D}$ is the set of designers who used the Web API $\mathcal{W}$ in one of their mashups and $|\cdot|$ denotes the set cardinality (that is, the number of designers); (ii) M_i is the set of mashups owned by a designer $d_i \in \mathcal{D}_\mathcal{W}$ and $m_i^k \in M_i$; (iii) σ_i is the skill associated to the designer $d_i \in \mathcal{D}_\mathcal{W}$.

The similarity between the request $\mathcal{R}$ and each Web API $\mathcal{W} \in \Omega$ in the repository $\Re$, based on the i-th technical feature, denoted here as $\overline{Sim_i}(\mathcal{R}, \mathcal{W})$, is defined as follows:

$$\overline{Sim_i}(\mathcal{R},\mathcal{W}) = \begin{cases} Sim_i(\mathcal{R},\mathcal{W}) & for\ single\ Web\ API\ selection \\ \\ Sim_i^{asym}(\mathcal{R},\mathcal{W}) & otherwise \end{cases} \tag{5}$$

where $Sim_i(\mathcal{R},\mathcal{W})$ is computed as the number of common elements for the i-th feature (for instance, the number of common protocols), denoted with $|\mathcal{F}^i_{\mathcal{R}} \cap \mathcal{F}^i_{\mathcal{W}}|$, with respect to the overall number of allowed values, that is:

$$Sim_i(\mathcal{R},\mathcal{W}) = \frac{2 \cdot |\mathcal{F}^i_{\mathcal{R}} \cap \mathcal{F}^i_{\mathcal{W}}|}{|\mathcal{F}^i_{\mathcal{R}}| + |\mathcal{F}^i_{\mathcal{W}}|} \in [0,1] \tag{6}$$

Note that, in this case, common values might mean also compatible values (such as RSS and Atom formats, which are both based on XML). Nevertheless, in our framework, the formula shown in Equation (6) is applied only for single Web API selection. When a (proactive) mashup completion or a Web API substitution must be performed, we provide an asymmetric variant of this formula, that assigns more relevance to the values of the i-th feature required in the request $\mathcal{R}$, that is:

$$Sim_i^{asym}(\mathcal{R},\mathcal{W}) = \frac{|\mathcal{F}^i_{\mathcal{R}} \cap \mathcal{F}^i_{\mathcal{W}}|}{|\mathcal{F}^i_{\mathcal{R}}|} \in [0,1] \tag{7}$$

This variant has been designed to take the viewpoint of the Web API request in terms of required technical features: if the Web API $\mathcal{W}$ has been designed considering all the required values for a feature, this is considered as the best situation, no matter $\mathcal{W}$ allows additional values for the same feature. This is due to the specific nature of mashup completion and Web API substitution search targets. In fact, in these cases the required feature values are the ones of other Web APIs already included in the mashup under development. Therefore, the idea is to assign more relevance to those Web APIs that share (compatible) features with the other Web APIs already in the mashup to be developed.

3.1 Ranking Metrics for Web API Search

The Web APIs returned as search results among the ones available in the repository $\Re$ (which we denote with $\{\mathcal{W}'\} \subseteq \Omega$) are those whose overall similarity is equal or greater than a threshold $\gamma \in [0,1]$ set by the web designer. Search results in $\{\mathcal{W}'\}$ are ranked according to the values of $Sim(\mathcal{R},\mathcal{W})$. Nevertheless, additional ranking criteria can be added based on the ratings assigned by web designers to Web APIs. Formally, search results in $\{\mathcal{W}'\}$ are ranked according to the following equation:

$$\rho(\mathcal{W}') = Sim(\mathcal{R},\mathcal{W}) \cdot \frac{1}{|\mathcal{D}_{\mathcal{W}'}|} \cdot \sum_{i=1}^{|\mathcal{D}_{\mathcal{W}'}|} \frac{\sum_{k=1}^{|M_i|} \sigma_i \cdot \mu(\mathcal{W}', m_i^k, d_i)}{|M_i|} \in [0,1] \tag{8}$$

where: (i) $\mathcal{D}_{\mathcal{W}'}$ is the set of the designers who rated the Web API $\mathcal{W}'$ in some mashups and $|\mathcal{D}_{\mathcal{W}'}|$ denotes the cardinality of the set (that is, the number of designers); (ii) for each designer $d_i \in \mathcal{D}_{\mathcal{W}'}$, M_i represents the set of mashups where

d_i used (and rated) the Web API $\mathcal{W}'$ and $m_i^k \in M_i$; (iii) the ratings assigned by more expert designers (that is, designers who present higher σ_i) are considered as more important.

4 Conclusions

In this paper we proposed a framework which implements a set of techniques and mechanisms to support Web API search and ranking by exploiting the "wisdom" of the other designers who used the Web APIs for their own mashups. Designers' wisdom is properly weighted using their expertise in developing web mashups and semantic tags and ratings they assigned on Web APIs. As future extensions of this work, also social relationships between developers could be used to automatically infer useful information, such as developers' reliability, while a complete implementation and in-depth evaluation of the metrics is being developed.

References

1. Gruber, T.: Collective knowledge systems: Where the Social Web meets the Semantic Web. Journal Web Semantics: Science, Services and Agents on the World Wide Web 6, 4–13 (2008)
2. Torres, R., Tapia, B., Astudillo, H.: Improving Web API Discovery by leveraging social information. In: Proceedings of the IEEE International Conference on Web Services, pp. 744–745 (2011)
3. Bianchini, D., De Antonellis, V., Melchiori, M.: Semantic Collaborative Tagging for Web APIs Sharing and Reuse. In: Brambilla, M., Tokuda, T., Tolksdorf, R. (eds.) ICWE 2012. LNCS, vol. 7387, pp. 76–90. Springer, Heidelberg (2012)
4. Gomadam, K., Ranabahu, A., Nagarajan, M., Sheth, A., Verma, K.: A Faceted Classification Based Approach to Search and Rank Web APIs. In: Proc. of International Conference on Web Services (ICWS), pp. 177–184 (2008)
5. Bianchini, D., De Antonellis, V., Melchiori, M.: A Linked Data Perspective for Effective Exploration of Web APIs Repositories. In: Daniel, F., Dolog, P., Li, Q. (eds.) ICWE 2013. LNCS, vol. 7977, pp. 506–509. Springer, Heidelberg (2013)
6. Bianchini, D., De Antonellis, V., Melchiori, M.: A Multi-perspective Framework for Web API Search in Enterprise Mashup Design. In: Salinesi, C., Norrie, M.C., Pastor, Ó. (eds.) CAiSE 2013. LNCS, vol. 7908, pp. 353–368. Springer, Heidelberg (2013)
7. Shafiq, M., Alhajj, A., Rokne, J.: On the social aspects of personalized ranking for web services. In: Proc. of 13th IEEE Int. Conference on High Performance Computing and Communications, pp. 86–93 (2011)
8. Dojchinovski, M., Kuchar, J., Vitvar, T., Zaremba, M.: Personalised Graph-Based Selection of Web APIs. In: Cudré-Mauroux, P., et al. (eds.) ISWC 2012, Part I. LNCS, vol. 7649, pp. 34–48. Springer, Heidelberg (2012)
9. Hotho, A., Jäschke, R., Schmitz, C., Stumme, G.: Information Retrieval in Folksonomies: Search and Ranking. In: Sure, Y., Domingue, J. (eds.) ESWC 2006. LNCS, vol. 4011, pp. 411–426. Springer, Heidelberg (2006)

Erratum: On the Move to Meaningful Internet Systems: OTM 2013 Conferences

Volume Editors

Robert Meersman, Free University of Brussels, Belgium
E-mail: meersman@vub.ac.be

Hervé Panetto, University of Lorraine, Vandoevre-les-Nancy, France
E-mail: herve.panetto@univ-lorraine.fr

Tharam Dillon, La Trobe University, Melbourne, VIC, Australia
E-mail: tharam.dillon7@gmail.com

Johann Eder, University of Klagenfurt, Austria
E-mail: johann.eder@aau.at

Zohra Bellahsene, University of Montpellier II, France
E-mail: bella@lirmm.fr

Norbert Ritter, University of Hamburg, Germany
E-mail: ritter@informatik.uni-hamburg.de

Pieter De Leenheer, VU University Amsterdam, The Netherlands
E-mail: pieter.de.leenheer@vu.nl

Deijing Dou, University of Oregon, Eugene, OR, USA
E-mail: dou@cs.uoregon.edu

R. Meersman et al. (Eds.): OTM 2013, LNCS 8185, 2013.

DOI 10.1007/978-3-642-41030-7_57

The name of "Dejing Dou" was incorrectly spelled as "Deijing Dou".

Affected:

Front cover page

Inner front-matter pages III, IV, IX, XXVI

Chapter page 451

Author Index page 773

The original online version for this volume can be found at
http://dx.doi.org/10.1007/978-3-642-41030-7

Author Index

Aberer, Karl 222, 241
Ali, Muhammad Intizar 588
Amziani, Mourad 21
Anisetti, Marco 434
Ardagna, Claudio A. 434
Arora, Rajeev 703

Badr, Youakim 300
Bahulkar, Ashwin 642
Becker, Anne 719
Bellahsene, Zohra 1, 711
Bellatreche, Ladjel 309
Beneventano, Domenico 616
Ben Hassine, Ahlem 542
Bezzi, Michele 434
Bianchini, Devis 764
Biennier, Frédérique 300
Biffl, Stefan 564
Binz, Tobias 130, 360
Boucelma, Omar 542
Boudjlida, Nacer 267
Breitenbücher, Uwe 130, 360
Breß, Sebastian 309
Brinkkemper, Sjaak 755
Burattin, Andrea 94

Cardoso, Diego Sá 572
Casanova, Marco Antonio 572
Choenni, Sunil 755
Cimitile, Marta 94
Colman, Alan 149
Combi, Carlo 39
Couto, Francisco M. 527
Crispo, Bruno 342
Cruz, Isabel F. 527

Damiani, Ernesto 434
da Silva Teixeira, Maria das Graças 737
De Antonellis, Valeria 764
Debruyne, Christophe 564, 677
Decat, Maarten 342
De Leenheer, Pieter 451
Deng, Tangjian 685
de Spindler, Alexandre 204
Dorn, Christoph 167
Dou, Deijing 451
Dou, Dejing 470
Dustdar, Schahram 167

Eder, Johann 1, 659
Erdur, Hebun 719

Fagan, Ollie 588
Faria, Daniel 527
Feng, Ling 685
Ferreira, Paulo 259
Folino, Francesco 275
Frick, Markus 719
Friel, Owen 588

Gal, Avigdor 241
Geibel, Peter 719
Geniet, Dominique 309
Grossmann, Georg 75
Guarascio, Massimo 275
Guizzardi, Giancarlo 737

Hadjali, Allel 519
Hagedorn, Stefan 452
Hamadache, Kahina 377
Han, Jun 149
Harispe, Sébastien 606
Haupt, Florian 360
Hipp, Markus 112
Holzgreve, Alfred 719
Hussain, Farookh Khadeer 291

Jalali, Amin 2
Janaqi, Stefan 606
Jean, Stéphane 519
Jegzentis, Kati 719
Johannesson, Paul 2
Joosen, Wouter 342

Kabir, Muhammad Ashad 149
Katz, Michael 241
Kerkad, Amira 309
Khalaf, Rania Y. 1
Köpke, Julius 659
Kopp, Oliver 130, 360

Koschmider, Agnes 57
Kovalenko, Olga 564
Kriglstein, Simone 57
Krüger, Stefan 719
Kulkarni, Sumant 703

Lagaisse, Bert 342
Lanz, Andreas 39
Leertouwer, Erik 755
Leone, Stefania 204
Levy, Eliezer 241
Leymann, Frank 130, 360
Lezoche, Mario 267
Li, Wenbin 300
Liao, Yongxin 267
Lotz, Volkmar 308
Loures, Eduardo 267
Lu, Jie 291
Lukasiewicz, Thomas 488, 501
Luong, Xuan Hoai 222

Mafazi, Shamila 75
Maggi, Fabrizio Maria 94
Mars, Ammar 519
Martinez, Maria Vanina 488, 501
Mateus, Miguel 259
Mayer, Wolfgang 75
McLeod, Dennis 204
Mehmood, Qaiser 588
Melchiori, Michele 764
Melliti, Tarek 21
Memon, Tasneem 291
Michelberger, Bernd 112
Miklós, Zoltán 222, 241
Mileo, Alessandra 588
Mitschang, Bernhard 360
Montmain, Jacky 606
Müller, Frank 719
Musen, Mark A. 470
Mutschler, Bela 112

Negrão, André Pessoa 259
Neumayr, Bernd 624
Ngo, DuyHoa 711
Nijs, Niels 677
Nolte, Christian Hans 719
Nowak, Alexander 360
Noy, Natalya F. 470

Olaru, Marius Octavian 616
Ouyang, Chun 2

Palmonari, Matteo 527
Panetto, Hervé 267
Pearson, Siani 414
Pesquita, Catia 527
Petrakis, Euripides G.M. 185
Polleres, Axel 555
Pontieri, Luigi 275
Posenato, Roberto 39

Quan, Tho Thanh 222
Quoc Viet Nguyen, Hung 222, 241

Raftopoulou, Paraskevi 185
Ranwez, Sylvie 606
Rauniyar, Rajan 291
Reddy, Sreedhar 642
Reich, Christoph 395, 403
Reichert, Manfred 39, 112
Ricardo de Almeida, Falbo 737
Ritter, Norbert 308
Rong, Chunming 327
Rübsamen, Thomas 403

Sabetta, Antonino 434
Sagi, Tomer 241
Santos, Emanuel 527
Sattler, Kai-Uwe 452
Scekic, Ognjen 167
Schaaf, Thorsten 719
Scheitz, Jan Friedrich 719
Schepers, Josef 719
Schicho, Michaela 659
Schrefl, Michael 624
Schütz, Christoph 624
Sellami, Sana 542
Serral, Estefanía 564
Setz, Jochen 719
Shafran, Victor 241
Simari, Gerardo I. 488, 501
Slaimi, Fatma 542
Sperduti, Alessandro 94
Spruit, Marco 755
Srinivasa, Srinath 703
Steyskal, Simon 555
Stumptner, Markus 75
Sulistio, Anthony 395

TaheriMonfared, Aryan 327
Takizawa, Makoto 308
Tata, Samir 21
Todorov, Konstantin 711
Tolxdorff, Thomas 719
Tragatschnig, Simon 283
Tran, Huy 283
Trautwein, Martin 719
Tryfonopoulos, Christos 185
Tudorache, Tania 470
Tütüncü, Serdar 719

Ullrich, Meike 57
Usnich, Tatiana 719

van Dijk, Jan 755
Van Landuyt, Dimitri 342
Veiga, Luís 259
Vidal, Vania Maria P. 572
Vincini, Maurizio 616

Wagner, Sebastian 360
Waizenegger, Tim 360
Wang, Hao 470
Weidlich, Matthias 241
Wettinger, Johannes 130
Wieland, Matthias 360
Wohed, Petia 2

Yu, Jian 149

Zdun, Uwe 283
Zevlis, Nikos 185
Zhao, Liang 685
Zimmermann, Lothar 719