

Barbara Pernici (Ed.)

LNCS 6051

Advanced Information Systems Engineering

22nd International Conference, CAiSE 2010
Hammamet, Tunisia, June 2010
Proceedings



CAiSE'10
Hammamet, Tunisia



Springer

Commenced Publication in 1973

Founding and Former Series Editors:

Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

David Hutchison

Lancaster University, UK

Takeo Kanade

Carnegie Mellon University, Pittsburgh, PA, USA

Josef Kittler

University of Surrey, Guildford, UK

Jon M. Kleinberg

Cornell University, Ithaca, NY, USA

Alfred Kobsa

University of California, Irvine, CA, USA

Friedemann Mattern

ETH Zurich, Switzerland

John C. Mitchell

Stanford University, CA, USA

Moni Naor

Weizmann Institute of Science, Rehovot, Israel

Oscar Nierstrasz

University of Bern, Switzerland

C. Pandu Rangan

Indian Institute of Technology, Madras, India

Bernhard Steffen

TU Dortmund University, Germany

Madhu Sudan

Microsoft Research, Cambridge, MA, USA

Demetri Terzopoulos

University of California, Los Angeles, CA, USA

Doug Tygar

University of California, Berkeley, CA, USA

Gerhard Weikum

Max-Planck Institute of Computer Science, Saarbruecken, Germany

Barbara Pernici (Ed.)

Advanced Information Systems Engineering

22nd International Conference, CAiSE 2010
Hammamet, Tunisia, June 7-9, 2010
Proceedings

Volume Editor

Barbara Pernici
Politecnico di Milano
Dipartimento di Elettronica e Informazione
Piazza Leonardo da Vinci 32, 20133 Milano, Italy
E-mail: barbara.pernici@polimi.it

Library of Congress Control Number: 2010927158

CR Subject Classification (1998): H.4, H.3, D.2, C.2, J.1, I.2

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

ISSN 0302-9743
ISBN-10 3-642-13093-3 Springer Berlin Heidelberg New York
ISBN-13 978-3-642-13093-9 Springer Berlin Heidelberg New York

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer. Violations are liable to prosecution under the German Copyright Law.

springer.com

© Springer-Verlag Berlin Heidelberg 2010
Printed in Germany

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper 06/3180

Preface

The CAiSE series of conferences, started in 1989, provides an established venue in the information systems engineering area for presenting and exchanging results of design-oriented research in information systems. The 22nd CAiSE conference, held in Hammamet, Tunisia, during June 7-11, 2010, continued this tradition. The theme of CAiSE 2010 was *Information Systems Evolution*. Modern information systems are the result of the interconnection of systems of many organizations, are running in variable contexts, and require both a lightweight approach to interoperability and the capability to actively react to changing requirements and failures. In addition, users of information systems are becoming more and more mobile and ubiquitous, requiring the system to adapt to their varying usage contexts and goals. The evolution of an information system should be a continuous process rather than a single step, and it should be inherently supported by the system itself and the design of the information system should consider evolution as an inherent property of the system. The papers selected for CAiSE 2010 focus on this theme from various perspectives, from requirement engineering and conceptual modeling, to process and services modeling and analysis, to themes such as security, quality, management and applications in information systems.

The first two days consisted of pre-conference workshops and events on business process modeling, modeling methods, requirements engineering, organizational modeling, interoperability and cooperation, ontologies, governance, empirical research in process-oriented systems, and domain engineering. In addition, in a doctoral consortium the PhD students could present and discuss their research plans and in the CAiSE Forum position papers on innovative research projects which are still at a relatively early stage and short papers describing innovative tools and prototypes were presented.

Two invited speakers discussed the role of modeling in modern search computing systems, which enable end users to perform exploratory search processes over multi-domain data sources available on the Web, and of the impact of IT systems on the use of the earth's resources, discussing design issues for IT systems themselves and for systems to manage resources, such as for instance electrical power in homes and enterprises, water delivery and distribution, and traffic management.

We thank all Program Committee and Program Board members and all additional reviewers who were essential in making this once again an excellent conference, contributing their time and effort. Each submission was reviewed by at least three reviewers. The Program Board members acted as assistant Program Committee Chairs and coordinated on-line discussions among the reviewers. The Program Board and Program Committee members met in January 2010 in Milan, Italy, to select papers based on the reviews and on-line discussions. Out of 299 submitted papers, 39 were accepted for the main conference.

We also thank everyone involved in this process. In particular, we are grateful to the General Chairs and to all Local Organizers for managing the complex coordination involved in organizing a conference, the Workshop and Tutorial Chairs, the Forum Chairs, the Doctoral Consortium Chairs, and extend our thanks to the sponsors who made the event financially feasible. Finally, we thank the participants both from academia and industry and we hope that their active participation in the CAiSE conference was inspiring for their research and a good support to industrial innovation.

June 2010

Barbara Pernici

Organization

Advisory Committee

| | |
|------------------|--|
| Arne Sølvberg | Norwegian University of Science and Technology, Norway |
| Janis Bubenko Jr | Royal Institute of Technology, Sweden |
| Colette Rolland | Université Paris 1 Panthéon Sorbonne, France |

General Co-chairs

| | |
|-------------------|--|
| Colette Rolland | Université Paris 1 Panthéon Sorbonne, France |
| Henda Ben Ghezala | ENSI, University of Manouba, Tunisia |

Program Chair

| | |
|-----------------|------------------------------|
| Barbara Pernici | Politecnico di Milano, Italy |
|-----------------|------------------------------|

Organization Chair

| | |
|----------------|--------------------------------------|
| Naoufel Kraiem | ENSI, University of Manouba, Tunisia |
|----------------|--------------------------------------|

Workshop and Tutorial Chairs

| | |
|-------------------|-----------------------------------|
| Pierluigi Plebani | Politecnico di Milano, Italy |
| Jolita Ralyté | University of Geneva, Switzerland |

Forum Chairs

| | |
|--------------|--|
| Pnina Soffer | University of Haifa, Israel |
| Erik Proper | Radboud University Nijmegen, The Netherlands |

Doctoral Consortium Chairs

| | |
|--------------------|--|
| Boualem Banatallah | University of New South Wales, Australia |
| Anne Persson | University of Skövde, Sweden |

Publicity Chairs

| | |
|---------------|--|
| Selmin Nurcan | University Paris 1 Panthéon Sorbonne, France |
| Lida Xu | Old Dominion University, USA |
| Rim Kaabi | ISI, University El-manar, Tunisia |

Publication Chairs

Cinzia Cappiello Politecnico di Milano, Italy
Motoshi Saeki Tokyo Institute of Technology, Japan

Finance Chair

Yassine Jamoussi ENSI, University of Manouba, Tunisia

Local Arrangements

Moncef Gafsi MERS, Tunisia
Malek Ghenima ENSI, University of Manouba, Tunisia
Semia Sonia Selmi ENSI, University of Manouba, Tunisia
Amina Sayeb MERS, Tunisia

Website

Jamil Dimassi ISI, University El-manar, Tunisia
Slim Mesfar ISI, University El-manar, Tunisia
Samir Azzouz ISI, University El-manar, Tunisia

Program Committee Board

Hans Akkermans, The Netherlands Antoni Olive, Spain
Sjaak Brinkkemper, The Netherlands Andreas Opdahl, Norway
Valeria De Antonellis, Italy Oscar Lopez Pastor, Spain
Eric Dubois, Luxembourg Anne Persson, Sweden
Marlon Dumas, Estonia Klaus Pohl, Germany
Pericles Loucopoulos, UK Pnina Soffer, Israel
Maira Norrie, Switzerland

Program Committee

Wil van der Aalst, The Netherlands Nacer Boudjlida, France
Pär Ågerfalk, Sweden Mokrane Bouzeghoub, France
Antonia Albani, The Netherlands Fabio Casati, Italy
Marco Bajec, Slovenia Silvana Castano, Italy
Luciano Baresi, Italy Jaelson Castro, Brazil
Zorah Bellahsene, France Corinne Cauvet, France
Boalem Benatallah, Australia João Falcão Cunha, Portugal
Giuseppe Berio, Italy Joerg Evermann, Canada
Claudio Bettini, Italy Xavier Franch, Spain
Rafik Bouaziz, Tunisia Mariagrazia Fugini, Italy

Claude Godart, France
 Mohand-Said Hacid, France
 Terry Halpin, USA
 Brian Henderson-Sellers, Australia
 Willem-Jan van den Heuvel,
 The Netherlands
 Patrick Heymans, Belgium
 Yassine Jamoussi, Tunisia
 Matthias Jarke, Germany
 Paul Johannesson, Sweden
 Marite Kirikova, Latvia
 Naoufel Kraiem, Tunisia
 John Krogstie, Norway
 Wilfried Lemahieu, Belgium
 Michel Leonard, Switzerland
 Kalle Lyytinen, USA
 Raimundas Matulevicius, Belgium
 Andrea Maurino, Italy
 Jan Mendling, Germany
 Isabelle Mirbel, France
 Haris Mouratidis, UK
 John Mylopoulos, Canada
 Selmin Nurcan, France
 Andreas Oberweis, Germany
 Jeffrey Parsons, Canada
 Mike Papazoglou, The Netherlands
 Michael Petit, Belgium
 Yves Pigneur, Switzerland
 Geert Poels, Belgium
 Erik Proper, The Netherlands

Jolita Ralyté, Switzerland
 Manfred Reichert, Germany
 Hajo Reijers, The Netherlands
 Mart Roantree, Ireland
 Michael Rosemann, Australia
 Gustavo Rossi, Argentina
 Matti Rossi, Finland
 Motoshi Saeki, Japan
 Camille Salinesi, France
 Keng Siau, USA
 Monique Snoeck, Belgium
 Janis Stirna, Sweden
 Arnon Sturm, Israel
 Stefan Tai, USA
 David Taniar, Australia
 Ernest Teniente, Spain
 Bernhard Thalheim, Germany
 Farouk Toumani, France
 Aphrodite Tsalgatidou, Greece
 Jean Vanderdonckt, Belgium
 Irene Vanderfeesten, The Netherlands
 Olegas Vasilecas, Lithuania
 Yair Wand, Canada
 Mathias Weske, Germany
 Hans Weigand, The Netherlands
 Roel Wieringa, The Netherlands
 Carson Woo, Canada
 Eric Yu, Canada
 Didar Zowghi, Australia

Additional Referees

Sofiane Abbar
 Alberto Abello
 Sudhir Agarwal
 Ghazi Al-Naymat
 Fernanda Alencar
 Bernd Amann
 David Ameller
 Ikram Amous
 Birger Andersson
 Saquib Anwar
 George Athanasopoulos
 Ahmed Awad

Marcos Baez
 Daniele Barone
 Christian Bartsch
 Moshe Chai Barukh
 Seyed-Mehdi-Reza Beheshti
 Sihem Ben Sassi
 Maria Bergholtz
 Maxime Bernaert
 Kahina Bessai
 Stefanie Betz
 Angela Bonifati
 Diogo Borges

Quentin Boucher
Lotfi Bouzguenda
Zouheir Brahmia
Stephan Buchwald
Paolo Cappellari
Mohamed Amine Chaabene
Van Munin Chhieng
Jan Claes
Andreas Classen
Bruno Claudepierre
André van Cleeff
Remi Coletta
Marco Comerio
Chad Coulin
Madalina Croitoru
Maya Daneva
Ken Decreus
Maha Driss
Pascal van Eck
Golnaz Elahi
Cécile Favre
Alfio Ferrara
Dario Freni
Frederik Gailly
Nicolas Genon
Hanene Ghorbel
Faiza Ghozzi Jedidi
Alexander Grosskopf
Adnene Guabtni
Mariem Gzara
Hakim Hacid
Ahmed Hadj Kacem
Allel Hadjali
Abdelkader Heni
Susan Hickl
Marcel Hiel
Stijn Hoppenbrouwers
Frank Hu
Arnaud Hubaux
Ela Hunt
Elena Ivankina
Wassim Jaziri
Lei Jiang
Diana Kalibatiene
Maurice van Keulen

Paul El Khoury
Bojan Klemenc
Stefan Klink
David Knuplesch
Jens Kolb
Elena Kornyshova
Agnes Koschmider
Vera Kuenzle
Aleš Kumer
Andreas Lanz
Algirdas Laukaitis
Wim Laurier
Dejan Lavbić
Evaldas Lebedys
Maria Lencastre
Pei Li
Matthias Lohrmann
Marcia Lucena
Linh Thao Ly
Dewi Mairiza
Azzam Maraee
Sergio Mascetti
Slim Mesfar
Stefano Montanelli
Joao Mourinho
Kreshnik Musaraj
Henriqueta Novoa
Nurmuliani
Martin F. O'Connor
Marc Oriol
Sietse Overbeek
Matteo Palmonari
Michael Pantazoglou
Linda Pareschi
Cristhian Parra
Olivier Pivert
Stanislav Pokraev
Artem Polyvyanyy
Michael Predeschly
Rüdiger Pryss
Ricardo Argenton Ramos
Saoussen Rekhis
Daniele Riboni
Stefanie Rinderle-Ma
Carlos Rodriguez

Lior Rokach
Seung Ryu
Ana Šaša
Semia Selmi
Oumaima Saidani
Sherif Sakr
Khalid Saleem
Germain Saval
Samir Sebahi
Khurram Shahzad
Patrícia Souza Silveira
Grega Simone
Marten van Sinderen
Sergey Smirnov
Stefano Soi
Ove Sørensen
Sebastian Speiser

Marco Spruit
Peter Stürzel
Lovro Šubelj
Thein Than Thun
Justas Trinkunas
Patrick Valduriez
Gaia Varese
Hung Vu
Romain Vuillemot
Ingo Weber
Matthias Weidlich
Moe Wynn
Huayu Zhang
Christian Zirpins
Houda Zouari
Aljaž Zrnec

Table of Contents

Keynotes

| | |
|---|---|
| Search Computing Systems | 1 |
| <i>Stefano Ceri and Marco Brambilla</i> | |
| The Influence of IT Systems on the Use of the Earth's Resources | 7 |
| <i>Jurij Paraszcak</i> | |

Session 1: Business Process Modeling

| | |
|--|----|
| Design and Verification of Instantiable Compliance Rule Graphs in Process-Aware Information Systems | 9 |
| <i>Linh Thao Ly, Stefanie Rinderle-Ma, and Peter Dadam</i> | |
| Success Factors of e-Collaboration in Business Process Modeling | 24 |
| <i>Peter Rittgen</i> | |
| Beyond Process Mining: From the Past to Present and Future | 38 |
| <i>Wil M.P. van der Aalst, Maja Pesic, and Minseok Song</i> | |

Session 2: Information Systems Quality

| | |
|--|----|
| Dependency Discovery in Data Quality | 53 |
| <i>Daniele Barone, Fabio Stella, and Carlo Batini</i> | |
| Rationality of Cross-System Data Duplication: A Case Study | 68 |
| <i>Wiebe Hordijk and Roel Wieringa</i> | |
| Probabilistic Models to Reconcile Complex Data from Inaccurate Data Sources | 83 |
| <i>Lorenzo Blanco, Valter Crescenzi, Paolo Merialdo, and Paolo Papotti</i> | |

Session 3: Service Modelling

| | |
|--|-----|
| Monitoring and Analyzing Service-Based Internet Systems through a Model-Aware Service Environment | 98 |
| <i>Ta'id Holmes, Uwe Zdun, Florian Daniel, and Schahram Dustdar</i> | |
| Modeling and Reasoning about Service-Oriented Applications via Goals and Commitments | 113 |
| <i>Amit K. Chopra, Fabiano Dalpiaz, Paolo Giorgini, and John Mylopoulos</i> | |

| | |
|---|-----|
| Conceptualizing a Bottom-Up Approach to Service Bundling | 129 |
| <i>Thomas Kohlborn, Christian Luebeck, Axel Korthaus, Erwin Fieft, Michael Rosemann, Christoph Riedl, and Helmut Krcmar</i> | |

Session 4: Security and Management

| | |
|--|-----|
| Dynamic Authorisation Policies for Event-Based Task Delegation | 135 |
| <i>Khaled Gaaloul, Ehtesham Zahoor, François Charoy, and Claude Godart</i> | |
| A New Approach for Pattern Problem Detection | 150 |
| <i>Nadia Bouassida and Hanène Ben-Abdallah</i> | |
| Comparing Safety Analysis Based on Sequence Diagrams and Textual Use Cases | 165 |
| <i>Tor Stålhane, Guttorm Sindre, and Lydie du Bousquet</i> | |

Session 5: Matching and Mining

| | |
|---|-----|
| Feature-Based Entity Matching: The FBEM Model, Implementation, Evaluation | 180 |
| <i>Heiko Stoermer, Nataliya Rassadko, and Nachiket Vaidya</i> | |
| Dealing with Matching Variability of Semantic Web Data Using Contexts | 194 |
| <i>Silvana Castano, Alfio Ferrara, and Stefano Montanelli</i> | |
| GRUVE: A Methodology for Complex Event Pattern Life Cycle Management | 209 |
| <i>Sinan Sen and Nenad Stojanovic</i> | |
| Supporting Semantic Search on Heterogeneous Semi-structured Documents | 224 |
| <i>Yassine Mrabet, Nacéra Bennacer, Nathalie Pernelle, and Mouhamadou Thiam</i> | |
| Query Ranking in Information Integration | 230 |
| <i>Rodolfo Stecher, Stefania Costache, Claudia Niederée, and Wolfgang Nejdl</i> | |

Session 6: Case Studies and Experiences

| | |
|---|-----|
| Validity of the Documentation Availability Model: Experimental Definition of Quality Interpretation | 236 |
| <i>Raimundas Matulevičius, Najji Habra, and Flora Kamseu</i> | |

| | |
|--|-----|
| Emerging Challenges in Information Systems Research for Regulatory Compliance Management | 251 |
| <i>Norris Syed Abdullah, Shazia Sadiq, and Marta Indulska</i> | |
| Experience-Based Approach for Adoption of Agile Practices in Software Development Projects | 266 |
| <i>Iva Krasteva, Sylvia Ilieva, and Alexandar Dimov</i> | |
| Coordinating Global Virtual Teams: Building Theory from a Case Study of Software Development | 281 |
| <i>Gaye Kiely, Tom Butler, and Patrick Finnegan</i> | |
| Information Systems Evolution over the Last 15 Years | 296 |
| <i>Magne Davidsen and John Krogstie</i> | |

Session 7: Conceptual Modelling

| | |
|---|-----|
| From Web Data to Entities and Back | 302 |
| <i>Zoltán Miklós, Nicolas Bonvin, Paolo Bouquet, Michele Catasta, Daniele Cordioli, Peter Fankhauser, Julien Gaugaz, Ekaterini Ioannou, Hristo Koshutanski, Antonio Maña, Claudia Niederée, Themis Palpanas, and Heiko Stoermer</i> | |
| Transformation-Based Framework for the Evaluation and Improvement of Database Schemas | 317 |
| <i>Jonathan Lemaitre and Jean-Luc Hainaut</i> | |
| Reverse Engineering User Interfaces for Interactive Database Conceptual Analysis | 332 |
| <i>Ravi Ramdoyal, Anthony Cleve, and Jean-Luc Hainaut</i> | |
| Towards Automated Inconsistency Handling in Design Models | 348 |
| <i>Marcos Aurélio Almeida da Silva, Alix Mougnot, Xavier Blanc, and Reda Bendraou</i> | |

Session 8: Adaptation

| | |
|---|-----|
| Dynamic Metamodel Extension Modules to Support Adaptive Data Management | 363 |
| <i>Michael Grossniklaus, Stefania Leone, Alexandre de Spindler, and Moira C. Norrie</i> | |
| Supporting Runtime System Evolution to Adapt to User Behaviour | 378 |
| <i>Estefanía Serral, Pedro Valderas, and Vicente Pelechano</i> | |
| Interaction-Driven Self-adaptation of Service Ensembles | 393 |
| <i>Christoph Dorn and Schahram Dustdar</i> | |

Session 9: Requirements

On the Semantics of the Extend Relationship in Use Case Models:
 Open-Closed Principle or Clairvoyance? 409
Miguel A. Laguna, José M. Marqués, and Yania Crespo

Situational Evaluation of Method Fragments: An Evidence-Based
 Goal-Oriented Approach 424
Hesam Chiniforooshan Esfahani, Eric Yu, and Jordi Cabot

Incorporating Modules into the *i** Framework 439
Xavier Franch

Ahab’s Leg: Exploring the Issues of Communicating Semi-formal
 Requirements to the Final Users 455
*Chiara Leonardi, Luca Sabatucci, Angelo Susi, and
 Massimo Zancanaro*

The Brave New World of Design Requirements: Four Key Principles 470
*Matthias Jarke, Pericles Loucopoulos, Kalle Lyytinen,
 John Mylopoulos, and William Robinson*

Session 10: Process Analysis

The ICoP Framework: Identification of Correspondences between
 Process Models 483
Matthias Weidlich, Remco Dijkman, and Jan Mendling

Process Compliance Measurement Based on Behavioural Profiles 499
*Matthias Weidlich, Artem Polyvyanyy, Nirmal Desai, and
 Jan Mendling*

Business Trend Analysis by Simulation 515
*Helen Schonenberg, Jingxian Jian, Natalia Sidorova, and
 Wil van der Aalst*

Workflow Soundness Revisited: Checking Correctness in the Presence
 of Data While Staying Conceptual 530
Natalia Sidorova, Christian Stahl, and Nikola Trčka

Panel

Octavian Panel on Intentional Perspectives on Information Systems
 Engineering 545
Arne Sølvberg

Author Index 547

Search Computing Systems

Stefano Ceri and Marco Brambilla

Politecnico di Milano, Dipartimento di Elettronica ed Informazione,
V. Ponzio 34/5, 20133 Milano, Italy
{ceri,mbrambil}@elet.polimi.it

Abstract. Search Computing defines a new class of applications, which enable *end users* to perform exploratory search processes over multi-domain data sources available on the Web. These applications exploit suitable software frameworks and models that make it possible for *expert users* to configure the data sources to be searched and the interfaces for query submission and result visualization. We describe some usage scenarios and the reference architecture for Search Computing systems.

Keywords: Search Computing, software engineering, search engine, software architectures, Web information systems.

1 Introduction

Throughout the last decade, search has become the most adopted way to access information over the Internet. Users are asking for queries that are more and more engaging for search engines, in several senses: the user search activity assumes the form of an interaction process instead of a single query; the single query itself becomes more complex (in terms of amount and extension of information the user asks for); and the information to be retrieved is often hidden in the so called "deep Web", which contains information perhaps more valuable than what can be crawled on the surface Web.

Different classes of solutions have emerged to cover this information need: general-purpose search engines; meta-search engines that query several engines and build up a unified result set; vertical search engines that aggregate domain-specific information from a fixed set of relevant sources and let users pose more sophisticated queries (e.g., finding proper combinations of flights, hotels, and car rental offers).

However, an entire class of information needs remains to be supported: the case in which a user performs a complex *search process* [2], addressing different domains, possibly covered by distinct vertical search engines. For example, a user may wish to find a place where an interesting event occurs, that has good weather in a given period, with travel and accommodation options that match given budget and quality standards, maybe with close-by trekking opportunities. This kind of search can be performed by separately looking for each individual piece of information and then mentally collating partial results, to get a combination of objects that satisfies the needs, but such procedure is cumbersome and error prone.

We define *search computing applications* [4], as the new class of applications aimed at responding to multi-domain queries, i.e., queries over multiple semantic fields of interest, by helping users (or by substituting to them) in their ability to decompose queries and manually assemble complete results from partial answers. The contribution of this paper is to provide an overview of search computing system, comprising a description of the user roles and of the system architecture and deployment strategy.

Our work is coherent with the current trend in service-oriented architectures (SOA). Proposals for service specification include WSDL, UML models, and ontologies (WSMO and OWL-S). Service registration is based on UDDI and its variants, portals like XMethods.com and RemoteMethods.com, systems like Woogoo, Wsoogoo.com, Web Service Crawler Engine [1], and others. A large set of languages and tools address service orchestration too (e.g., BPMN, WS-BPEL, XPD). The SeCo Project is focused on the integration of search services, i.e. services producing ranked and chunked output, allowing consumers to halt the, when enough results are available. Research projects with comparable goals include: Microsoft Symphony, which enables non-developers to build and deploy search-driven applications that combine their data and domain expertise with content from search engines[7]; Google Squared (www.google.com/squared) and Fusion Tables (tables.googlelabs.com), which produce tabular results of searches over web and proprietary information respectively; Kosmix (www.kosmix.com), a general-purpose topic discovery engine, which responds to keyword search by means of a topic page. All these proposals miss several significant features with respect to SeCo, including: join of results, cost awareness, definition and optimization of query plans, and others.

2 The Search Computing Framework

A Search Computing application supports users in asking multi-domain queries; for instance, “Where can I attend a scientific conference close to a beautiful beach reachable with cheap flights?”. Search Computing proposes a configurable software architecture that supports any multi-domain query. The main idea is to delegate domain-neutral computation to a generic architecture, which is then configured with domain-dependent information to obtain the desired Search Computing application.

Search computing applications have several distinguished aspects:

- Data sources are usually heterogeneous and distributed; they must be made available to the search computing application as search services that respond to input queries by returning ranked results, possibly chunked into sub-lists.
- Multi-domain queries are answered by selecting a subset of the data sources, by individually extracting their responses, and then by joining the responses thereby building combinations that collectively constitute the results of the query; this is a special case of Web service selection and orchestration, where novel algorithms are needed for joining results sets. This implies that queries can be addressed only through complex orchestration algorithms, which must scale properly.

- Combined and ranked results must be presented to the user for inspection and query refinement, with an exploratory approach that allows augmenting complex results with new findings. This, together with the desired wide adoption of the applications by the user, requires scalability of the platform with respect to the number of users and of requests per user.
- Finally, search computing applications are especially effective for improving the visibility and economical value of so-called long-tail content, which can be used to produce highly customized search solutions. This entails the need of extreme scalability of the platform with respect to the number of registered services.

Figure 1 shows an overview of the Search Computing framework, constituted by several sub-frameworks. The *service mart framework* provides the scaffolding for wrapping and registering data sources. Data sources can be heterogeneous (examples are: a Web site wrapper, a Restful data source, a WSDL web service, a YQL data source, a materialized source, etc.); registration requires a standardization of their interface, so to comply with a service invocation protocol. A service mart is defined as an abstraction (e.g., Hotel) of one or more Web service implementations (e.g., Bookings and Expedia), each capable of accepting queries and of returning results, possibly ranked and chunked into pages. Registration metadata describes the service mart signature and connection information. A connection is defined as an input-output relationship between pairs of service marts that can be exploited for joining them (e.g., the declaration that the output city of the Hotel service mart can be used as an input destination to the Trip service mart). The *user framework* provides functionality and storage for registering users, with different roles and capabilities. The *query framework* supports the management and storage of queries as first class citizens: a query can be executed, saved, modified, and published for other users to see. The *service invocation framework* masks the technical issues involved in the interaction with the service mart, e.g., the Web service protocol and data caching issues.

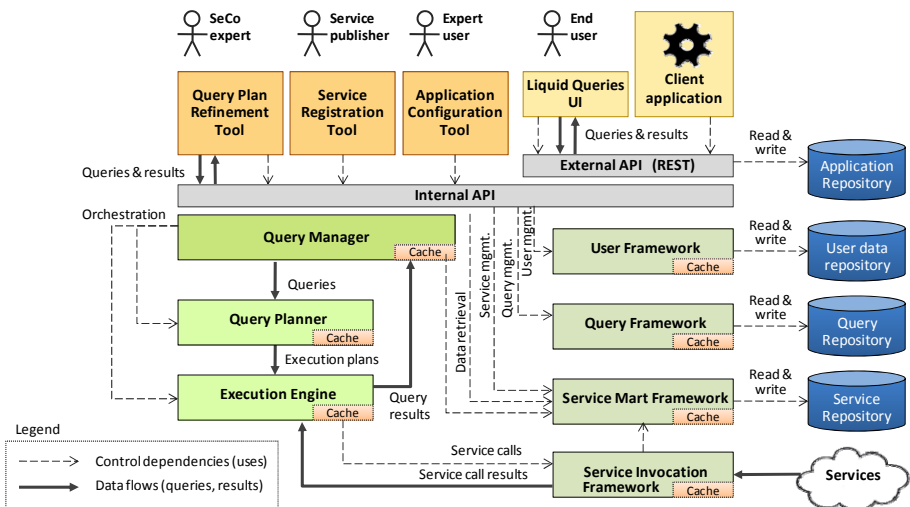


Fig. 1. Overview of the Search Computing framework

The core of the framework aims at executing multi-domain queries. The *query manager* takes care of splitting the query into sub-queries (e.g., "Where can I attend a scientific conference?"; "Which place is close to a beautiful beach?"; "Which place is reachable from my home location with cheap flights?") and bounding them to the respective relevant data sources registered in the service mart repository (in this case, conferences could be retrieved using Eventful.com, places close to a beach using Yelp.com, flights information using Expedia); starting from this mapping, the *query planner* produces an optimized query execution plan, which dictates the sequence of steps for executing the query. Finally, the execution engine actually executes the query plan, by submitting the service calls to designated services through the *service invocation framework*, building the query results by combining the outputs produced by service calls, computing the global ranking of query results, and producing the query result outputs in an order that reflects their global relevance.

To obtain a specific Search Computing application, the general-purpose architecture of Figure 1 is customized by users, supported by appropriate design tools.

3 Search Computing Users

The development of Search Computing applications involves users with different roles and expertise:

- **Service Publishers:** they are in charge of implementing mediators, wrappers, or data materialization components, so as to make data sources compatible with the service mart standard interface and expected behavior; they register service mart definitions within the service repository, and declare the connection patterns usable to join them.
- **Expert Users:** they configure Search Computing applications, by selecting the service marts of interest, by choosing a data source supporting the service mart, and by connecting them through connection patterns. They also configure the user interface, in terms of controls and configurability choices for the end user.
- **End Users:** they use Search Computing applications configured by expert users. They interact by submitting queries, inspecting results, and refining/evolving their information need according to an exploratory information seeking approach, which we call *Liquid Query* [3].

The development process steps lead to the final application accessed by the end user. The Liquid Query interface, instantiated during the application configuration phase, supports the "search as a process" paradigm, based on the continuous evolution, manipulation, and extension of queries and results; the query lifecycle consists of iterations of the steps of **query submission**, when the end user submits an initial liquid query; **query execution**, producing a result set that is displayed in the user interface; and **result browsing**, when the result can be inspected and manipulated through appropriate interaction primitives, which update either the result set (e.g., re-ranking or clustering the results) or the query (e.g., by expanding it with additional service marts or requesting for more results). This approach to development takes into account the trend towards empowerment of the user, as witnessed in the field of Web mash-ups [5]. Indeed, all the design activities from service registration on do not ask to perform low-level programming.

4 Search Computing Architecture

Given the aim of providing users with an efficient Search Computing framework and the expected amount of services, users and methods to solve multi-domain search queries, the adoption of a distributed paradigm is a natural consequence. This entails dealing with the common challenges of a distributed system, e.g. heterogeneity, scalability, concurrency, failure handling and transparency.

With this objective in mind, we have designed the architecture in Figure 2 in terms of logical layers:

- The *service layer* offers facilities to wrap and expose existing services.
- The *execution & mart layer* physically orchestrates the query execution plan by means of an engine component and provides the service mart abstraction through a repository and an invocation environment.
- The *query layer* translates the high-level query description over service marts into an optimized execution plan over service implementations. Queries and optimized plans are stored in a repository for subsequent reuse.
- The *application layer* provides a REST API to submit queries, an application repository to store application-specific data (such as UIs' definitions) and liquid query support. The latter enables a fluid user interaction thanks to client-side data management and to asynchronous communications with the SeCo back-end.
- The *tools layer* transversely provides design, management and interaction facilities to the levels of the proposed architecture.

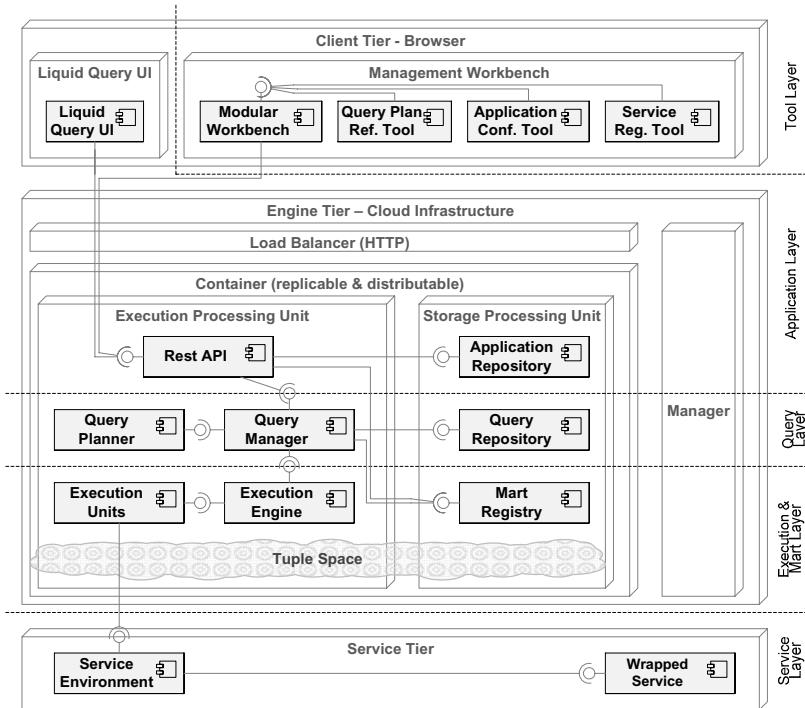


Fig. 2. Logical architecture and deployment diagram

The architecture is deployed in accordance with the scalability and performance requirements. We use a service-oriented architecture built on shared spaces within a grid computing framework (Space-Based Architecture). The power of spaces comes from not having to separate various parts of an application into discrete physical runtimes [8]. We organize the deployment in three separate tiers (Fig. 2):

- The *service tier* consists of the processing nodes providing access to registered services.
- The *client tier* consists of client machines locally running the liquid query UI, which is offered as a JavaScript component running inside Web browsers.
- The *engine tier* represents the query engine, which is invoked by clients and executes Search Computing queries over registered services.

The deployment of the *engine tier* exploits a high-end application server. In order to achieve scalability and high-performance, a load balancer distributes the execution load over a set of available processing units. A grid manager is in charge of the monitoring of the infrastructure, providing the instantiation and the recovery of grid containers according to a predefined Service Level Agreement (SLA). Finally, grid containers host processing and storage units and communicate by means of a tuple space. We adopt Gigaspaces XAP [6], a scalable, high-performance, reliable data grid implementation, supporting multiple clustering topologies to provide a clustered in-memory data repository (cache and application data), a fast message bus and a distributed platform for running the application's code.

Acknowledgements. This research is part of the Search Computing (SeCo) project, funded by the European Research Council (ERC), under the 2008 Call for "IDEAS Advanced Grants", a program dedicated to the support of frontier research. We thank all the project contributors and the advisory board members for their work.

References

- [1] Al-Masri, E., Mahmoud, Q.H.: Investigating web services on the World Wide Web. In: WWW 2008, Beijing, April 2008, pp. 795–804. ACM, New York (2008)
- [2] Baeza-Yates, R., Raghavan, P.: Next Generation Web Search. In: Ceri, S., Brambilla, M. (eds.) Search Computing. LNCS, vol. 5950, pp. 11–23. Springer, Heidelberg (2010)
- [3] Bozzon, A., Brambilla, M., Ceri, S., Fraternali, P.: Liquid Query: Multi-Domain Exploratory Search on the Web. In: WWW 2010, Raleigh, USA, April 2010. ACM, New York (2010) (in print)
- [4] Ceri, S., Brambilla, M. (eds.): Search Computing Challenges and Directions. LNCS, vol. 5950. Springer, Heidelberg (2010)
- [5] Daniel, F., Yu, J., Benatallah, B., Casati, F., Matera, M., Saint-Paul, R.: Understanding UI Integration: A Survey of Problems, Technologies, and Opportunities. IEEE Internet Computing 11(3), 59–66 (2007)
- [6] Gigaspaces XAP, <http://www.gigaspaces.com>
- [7] Shafer, J.C., Agrawal, R., Lauw, H.W.: Symphony: Enabling Search-Driven Applications. In: USETIM (Using Search Engine Tech. for Inf. Mgmt.) Workshop, VLDB Lyon (2009)
- [8] Shalom, N.: Space-Based Architecture and The End of Tier-based Computing, <http://www.gigaspaces.com/WhitePapers>

The Influence of IT Systems on the Use of the Earth's Resources

Jurij Paraszcak

IBM T.J. Watson Research Centre
Yorktown Heights, NY 10598
jurij@us.ibm.com

Information technology is so pervasive that estimates of its power usage show that data centers, locations where computing systems are densely packed consume some 2% of available power in mature economies. Other estimates show that an additional 8% or so of power is consumed by IT systems which reside in offices and homes, while assessments of power used consumer devices such as mobile phones and PDA's combined with other household appliances while charging or not deployed can add another 6-8% of total power consumption. As computing power densities have increased from a few tens of watts per square foot to hundreds of watts per square foot considerable attention has been applied to the management of power in IT systems.

Efforts to deal with these power densities includes compute and storage system re-designs to allow power to be controlled and managed, techniques to monitor and understand the distribution of thermal energy in data centers and the beginning of the integration of the grid to the IT based power demands of the data center.

Inasmuch as managing IT based power is an important issue, the same IT can be used to help manage the planet's resources such as electrical power in homes and enterprises, water delivery and distribution to agriculture and industry, flooding and sewage control, road traffic management and road charging, logistics and manufacturing processes and many other activities which use resources and generate waste.

In many situations where resources are used in cities, their usage is barely measured. For example the delivery of electrical power to homes is rarely measured more than once a month. Once the bill is submitted, the consumer has no real idea where the energy is spent and has no ability to plan how to manage that power. An even more critical issue within electrical power distribution networks is the identification of faults, where very little monitoring of the distribution networks results in irate customers being the only source of information as to the location and timing of faults. Similarly water usage by industrial, agricultural and enterprise customers and consumers is not usually monitored at any detailed level and issues such as leakage and contribute to waste and inefficiency.

Once these resource delivery systems become instrumented along with the corresponding consuming systems, the resulting data can be used to manage resources much more efficiently. This talk will address the nascent domain of resource management systems which combine the fields of data acquisition, distribution, modeling optimization and decision support and begin to allow automated systems to manage

resources within a city. Examples of the use of these integrated systems for traffic management in cities, energy management in data centers, water management in urban and suburban areas, carbon management in the distribution of goods and other situations will be shown. These will be shown to demonstrate the importance of models and their integration into the operation of these instrumented systems. The scope of resource reduction, while maintaining the same levels of service, will show the range of saving which are possible and descriptions of the IT architectures which enable this combination of data flow, models and decisions to be made will be provided.

Design and Verification of Instantiable Compliance Rule Graphs in Process-Aware Information Systems*

Linh Thao Ly¹, Stefanie Rinderle-Ma², and Peter Dadam¹

¹ Institute of Databases and Information Systems
Ulm University, Germany

² Faculty of Computer Science
University of Vienna, Austria

{thao.ly,peter.dadam}@uni-ulm.de, stefanie.rinderle-ma@univie.ac.at

Abstract. For enterprises it has become crucial to check compliance of their business processes with certain rules such as medical guidelines or financial regulations. When automating compliance checks on process models, existing approaches have mainly addressed *process-specific* compliance rules so far, i.e., rules that correspond to a particular process model. However, in practice, we will rather find *process-independent* compliance rules that are nevertheless to be checked over process models. Thus, in this paper, we present an approach that enables the *instantiation* and verification of process-independent compliance rules over process models using domain models. For this, we provide an intuitive visualization of compliance rules and compliance rule instances at user level and show how rules and instances can be formalized and verified at system level. The overall approach is validated by a pattern-based comparison to existing approaches and by means of a prototypical implementation.

1 Introduction

In many application domains, business processes are subject to compliance rules and policies that stem from domain specific requirements such as standardization or legal regulations [1]. Ensuring compliance of their business processes is crucial for enterprises nowadays, particularly since auditing and certification has become a competitive edge in many domains. Examples include certified family-friendly enterprises being more attractive to prospective employees or clinics proving a certain standard of their audited treatments to patients. Since process models are the common way to represent business processes, business process compliance can be ensured by verifying process models to be installed in process-aware information systems against imposed compliance rules. Tab. 1 summarizes quality compliance rules imposed on the software development process depicted in Fig. 1.

* This work was done within the SeaFlows project, which is funded by the German Research Foundation (DFG).

Table 1. Examples of compliance rules for software development

| |
|--|
| c_1 Goals have to be defined before starting the development. |
| c_2 Each test activity has to be documented. |
| c_3 After the development freeze, no further development activities shall take place. |
| c_4 Before carrying out a second usability test, a necessity check after the first usability test is necessary. |
| c_5 The testing has to be followed by an approval and the integration. Additionally, no changes shall take place between the approval and the integration. |

So far, existing approaches mainly focus on checking *process-specific* compliance rules that correspond to certain process models [1,2,3]. One example is rule c_1 (cf. Tab. 1) over process model P (cf. Fig. 1) referring to process activities **define goals** and **start of development phase**. However, in practice, compliance rules are often specified in a more general manner and not specifically in correspondence with a particular process model. Rule c_2 , for example, refers to a general **test** activity and not to a **functional test** as present in the software development process depicted in Fig. 1. Nevertheless, for quality assurance, it could be desired to verify c_2 over the development process. Thus, in order to support a broad spectrum of realistic compliance rules, we must enable the verification of both, process-specific and process-independent compliance rules over process models.

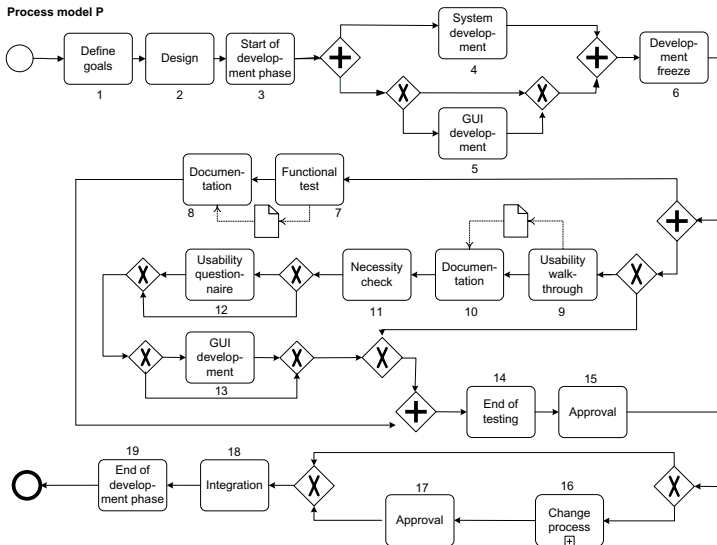


Fig. 1. Abstracted software development process (in BPMN notation)

Connected with supporting process-independent compliance rules, it is also necessary to enable the definition of compliance rules at different *design levels*. At user level an intuitive, high-level representation of compliance rules is desirable. Contrary, at system level, it must be possible to conduct automatic verification of process models against compliance rules. For this, it is necessary to *instantiate* compliance rules over a particular process model and to equip them with formal semantics for later verification.

In this paper, we present an approach for checking process-independent compliance rules over process models. This is achieved by instantiating these compliance rules over a particular process model using a domain model. The resulting compliance rule instances can be verified separated from each other. This enables individual compliance reports as well as individual solutions in case of compliance violations. Furthermore, we show how compliance rules can be defined at different design levels. At user level, compliance rules are specified using an intuitive visualization based on graph structures. At system level compliance rule graphs are equipped with formal semantics based on First Order Logic and execution traces. The latter guarantees for independence of a particular process meta model. The overall approach is validated by a pattern-based comparison to existing approaches, by a general discussion on the co-existence of compliance rules and process models, and by means of our prototypical implementation.

Sect. 2 discusses the compliance rule instantiation approach. Sect. 3 focuses on the design of instantiable compliance rule graphs. Their counterpart formalization is presented in Sect. 4. A validation of our approach is provided in Sect. 5. We close with a related work discussion in Sect. 6 and a summary in Sect. 7.

2 Compliance Rule Instantiation

Fig. 1 depicts the process model of a slightly abstracted software development process that we discovered within several practical student projects at Ulm University. As discussed in Sect. 1, several compliance rules might be imposed on the development process for quality and efficiency reasons (cf. Tab. 1). The basic challenge is now to check whether the process complies to these compliance rules or not. Additionally, in case of violations the system should yield helpful user feedback in precisely reporting the reason for the problem.

Compliance rules are generally defined at different abstraction levels ranging from rather abstract business policies to specific definitions [1]. To support different levels of abstraction for compliance rules, we enable the use of domain models. Based on these, compliance rules can be instantiated for certain processes. Note that domain models are demanded in many applications [4]. Moreover, we may also benefit from existing ontologies such as in the healthcare domain [5]. Take for example compliance rules c_2 and c_4 (cf. Tab. 1): both refer to test activities, where c_2 is general and c_4 more specific (`usability test`). Fig. 2 shows an extract of the domain model belonging to the development process. A test can be more specifically modeled as a `usability` or `functional test`, where `usability test` can additionally distinguished into `usability`

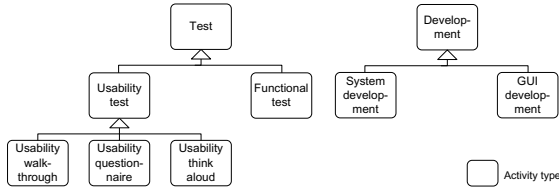


Fig. 2. An extract of the software development domain model

walkthroughs, usability questionnaires, or usability think aloud. A development might be either a system development or a GUI development.

Without any further knowledge, compliance rule c_2 cannot be evaluated over development process P since P does not contain any test activity. In fact, c_2 can only be evaluated by *instantiation* over P using the corresponding domain model as depicted in Fig. 2. Based on the domain model, the general test activity referred to by c_2 can be instantiated by three more specific testing activities contained within P (i.e., usability walkthrough, usability questionnaire, and functional test). This results in three *compliance rule instances* c_{2_1} , c_{2_2} , and c_{2_3} derived from c_2 (cf. Tab. 2). The usability walkthrough and the functional test are both documented according to the control flow of P . Thus, c_{2_1} and c_{2_2} are satisfied. However, the usability questionnaire is not documented within P what violates instance c_{2_3} .

We see that by using instantiation the verification of c_2 over P becomes possible. However, which benefits are offered by maintaining c_2 and instantiating it "on demand" instead of replacing c_2 by its instances c_{2_1} , c_{2_2} , and c_{2_3} for P and the corresponding domain model? The different advantages become evident when looking at the modeling, maintenance, and evolution of compliance rules on the one side and compliance checking for process models on the other side as depicted in Fig. 3.

First of all, for more complex domain models and processes, without instantiation, the number of compliance rules might dramatically increase resulting in huge effort for compliance rule modeling and maintenance. Moreover, supporting high-level compliance rules and compliance rule instantiation eases the evolution of compliance rules. Imagine, for example, that compliance rule c_2 has to be adapted (e.g., due to changes in the quality policies) such that each test activity not only has to be documented but also has to be approved. In this case, if the strategy to explicitly model all compliance rule instances of c_2 was

Table 2. Compliance rule instances for c_2 , cf. Tab. 1

| | |
|-----------|---|
| c_{2_1} | The functional test has to be documented. |
| c_{2_2} | The usability walkthrough has to be documented. |
| c_{2_3} | The usability questionnaire has to be documented. |

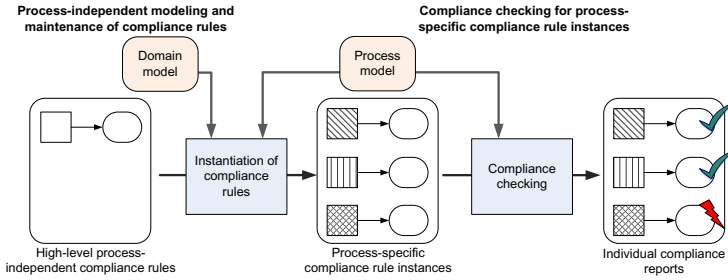


Fig. 3. The SeaFlows approach: compliance rule instantiation

applied, c_{2_1} , c_{2_2} , and c_{2_3} would have to be individually modified in order to integrate the change. By contrast, following the approach proposed in this paper, only high-level rule c_2 has to be modified resulting in modified high-level rule c'_2 . c'_2 then can be reinstated and checked for relevant process models.

Moreover, checking compliance at instance level results in fine-granule feedback on individual violations of rule instance (cf. Fig. 3). In turn, fine-granule feedback enables fine-granule treatment of compliance rule instance violations. Let us assume, for example, that c_2 is only of recommendation nature (i.e., enforcement level low). Since c_{2_3} is violated over process model P, the process designer might decide to completely ignore c_2 . However, since c_{2_1} and c_{2_2} are actually fulfilled, he might prefer to ignore c_{2_3} instead of c_2 .

Altogether, using compliance rule instantiation as proposed in this paper, we achieve minimal effort for compliance rule modeling and maintenance on the one hand, but enable individual compliance checks and corresponding reports for compliance rule instances on the other hand.

3 Instantiable Compliance Rule Graphs

The next challenge is to design instantiable compliance rules in a way that they can be easily understood by users. This task includes representation of (high-level) compliance rules as well as of compliance rule instances. We found that at both levels, graphs provide an intuitive visualization that can be equipped with formal semantics and verified at system level later on (cf. Sect. 4).

Process-independent Compliance Rules. To support process-independent (i.e., high-level) compliance rule graphs and as well as their instantiation over particular process models, a data model as depicted in Fig. 4 is needed: a process model consists of a set of nodes distinguished by their node id to which activities are assigned. An activity is assigned to activity types. An activity type may be the sub-type of another activity. For example, the activity type `usability walkthrough` is a sub-type of the activity type `usability test` (cf. domain model in Fig. 2).

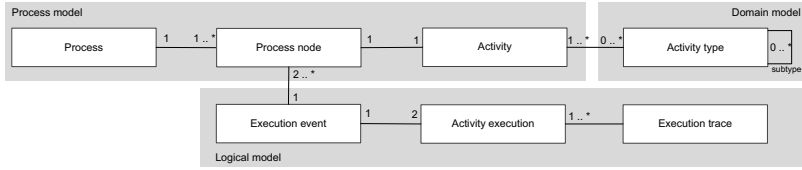


Fig. 4. The data model for instantiable compliance rules

When looking at compliance rules c_1 to c_5 (cf. Tab. 1), it can be observed that compliance rules mostly reflect certain patterns. Typically, compliance rules require the occurrence and/or the absence of certain activities (*occurrence/absence* patterns). For c_1 , for example, activity **define goals** should occur before starting the **development**. By contrast, for c_3 further **development** activities should be absent if activity **development freeze** has taken place. Furthermore, it can be observed that the occurrence or absence of certain activities is often conditional upon the occurrence or absence of other activities. Thus a compliance rule mostly comprises a triggering part (denoted as *antecedent* pattern) and a *consequence* pattern. To be able to instantiate process-independent compliance rule graphs later on, each node of a rule graph is associated with an activity type from the domain model (cf. Fig. 4). Directed edges connecting the nodes represent predecessor relations.

Based on these observations compliance rules c_1 to c_5 can be described by compliance rule graphs as depicted in Fig. 5. Compliance rule graph c_1 , for example, states that the execution of activity **start of development process** has to be directly or indirectly preceded by the execution of the activity **define goals**. Compliance rule graph c_5 states that the execution of activity **end of testing** has to be succeeded by an execution of **approval** and **integration**. Between these two executions, however, no execution of the **change process** must occur. Note that compliance rule graphs might also contain data information as data object **Test** for c_2 . How to integrate and evaluate such data information into compliance rules is subject to our future research.

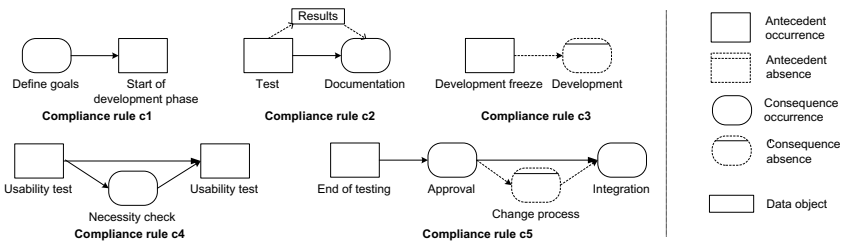


Fig. 5. Process-independent compliance rule graphs

Process-specific Instantiation of Compliance Rules . Generally, it must be possible to define compliance rule graphs independent of a particular process model. For later verification, however, it is necessary to instantiate independent compliance rule graphs over process models.

The instantiation of compliance rule graphs is accomplished by “binding” antecedent compliance rule nodes to nodes of the process model (reflected by their node ids) with the associated activity type. Instantiation of compliance rule c_2 (cf. Fig. 5) over process model P (cf. Fig. 1) using the domain model as depicted in Fig. 2, for example, results in three compliance rule instances as depicted in Fig. 6. Visualizing compliance rules that way, the user is able to locate exactly, which occurrences of a compliance rule are relevant, which are satisfied, and which are violated.

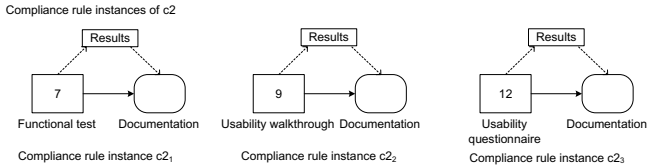


Fig. 6. Process-specific compliance rule instances

4 Formalization and Verification of Compliance Rules

The graphical compliance rule notation provides for more intuitive modeling of compliance rules since it hides formal details from the user. However, as we will show in this section, each compliance rule graph corresponds to a logical formula with defined semantics. The latter is necessary to enable the verification of processes against imposed compliance rules.

4.1 On Formalizing Compliance Rules

We opted for first-order predicate logic (FOL) to formalize compliance rules for several reasons. The use of a general and expressive logic such as FOL enables the extension of our approach in order to support further types of compliance rules not supported so far (e.g., authorization or organizational model compliance rules). Moreover, FOL allows for the definition of the antecedent-consequence-structure of compliance rules in a straight-forward and elegant manner. The nodes of compliance rule graphs are mapped to variables while properties of compliance rule graph nodes and the relations between nodes are mapped to corresponding predicates in a FOL formula. Due to lack of space we abstain from a complete definition here, but rather informally describe the structure in the following. The general structure of a compliance rule is as follows.

Structure 1 (Compliance rule). Let $A_{\mathcal{T}}$ be the set of activity types of the domain. Then, a compliance rule is of the following form:

$$\begin{aligned}
 & [true \mid antecedentOccurrencePatterns \mid antecedentAbsencePatterns \mid \\
 & antecedentOccurrencePatterns \wedge antecedentAbsencePatterns] \\
 & \rightarrow \\
 & consequence \vee consequence \vee \dots \vee consequence \\
 & consequence := \\
 & [consequenceOccurrencePatterns \mid consequenceAbsencePatterns \mid \\
 & consequenceOccurrencePatterns \wedge consequenceAbsencePatterns]
 \end{aligned}$$

In Struct. [1](#) the antecedent is either empty (i.e., the compliance rule is always activated) or consists of an *antecedent pattern*. The latter can be composed from occurrence patterns defining the occurrences of activity executions that activate the compliance rule. Compliance rule c_2 (cf. Fig. [5](#)), for example, is activated by the occurrence of an activity execution associated to the activity type **test** while compliance rule c_4 is activated by a more complex occurrence pattern (namely the sequence of two activity executions). The antecedent pattern may also consist of absence patterns defining the absence of particular activity executions. This allows for refining the occurrence pattern by putting additional conditions on the absence of activity executions (for example, to express patterns such as “if no approval takes place between the end of development and the integration”). If the antecedent of a compliance rule applies, one of the rule’s *consequence patterns* must also apply in order to satisfy the rule. Each consequence pattern, in turn, may consist of occurrence as well as absence patterns and corresponding relations. Compliance rule c_3 (cf. Fig. [5](#)), for example, has a consequence absence pattern in its consequence part while the consequence part of compliance rule c_5 is composed from both consequence occurrence and consequence absence patterns.

The formula for compliance rule c_2 is given below. It expresses that each *activity execution* associated to the activity type **test** has to be followed by an activity execution associated to the activity type **documentation** with the same **results** data object. This is a process-independent compliance rule, since it only references to activity types from the domain model (cf. Fig. [2](#)).

Compliance rule c_2

$$\forall t (ActivityType(t, \mathbf{test}) \rightarrow \exists d : (ActivityType(d, \mathbf{documentation}) \wedge Pred(t, d) \wedge results(t) = results(d)))$$

Based on the development process (cf. Fig. [11](#)), we can identify the process nodes that are associated to the activity types referenced in the formula (namely **test** and **documentation**). The formula for c_2 can be adapted accordingly by referring to activity executions associated to particular nodes in the process.

Process-specific compliance rule c_2

$$\begin{aligned}
 & \forall t (ProcessNode(t, 7) \vee ProcessNode(t, 9) \vee ProcessNode(t, 12) \rightarrow \\
 & \exists d : (ActivityType(d, \mathbf{documentation}) \wedge Pred(t, d) \wedge results(t) = results(d)))
 \end{aligned}$$

Based on this, we obtain the resulting compliance rule instances of c_2 as follows.

$$c_{2_1}: ProcessNode(t, 7) \rightarrow \exists d : (ActivityType(d, documentation) \wedge Pred(t, d) \wedge results(t) = results(d))$$

$$c_{2_2}: ProcessNode(t, 9) \rightarrow \exists d : (ActivityType(d, documentation) \wedge Pred(t, d) \wedge results(t) = results(d))$$

$$c_{2_3}: ProcessNode(t, 12) \rightarrow \exists d : (ActivityType(d, documentation) \wedge Pred(t, d) \wedge results(t) = results(d))$$

4.2 Interpretation of Compliance Rules

So far we showed that compliance rules are represented by FOL formulas. To round up the formalization of compliance rules, we also have to provide formal semantics for these formulas. The formal semantics has to be defined over an adequate logical model, that serves as interface between the compliance rule perspective and the process perspective. To support a variety of business process models, the logical model must be independent of a particular process meta-model. As discussed in our previous work in [6,3,7] *execution traces* are a suitable logical model since they are applicable to any process meta-model with formal execution semantics.

Generally, depending on their particular purpose, execution traces comprise different kinds of information. At minimum, execution traces store information on the execution of activities for a particular process instance (e.g., start, end, or start/end events for activity executions). Additional information might comprise actor assignments, input or output data, and timestamps (see, for example, the MXML execution traces used in ProM [8]). In the context of this paper, it is important to be able to instantiate compliance rules over process models. Thus, within execution traces, it should be possible to distinguish between concepts such as nodes and activities (cf. data model in Fig. 4). Thus, we define an ordered execution trace σ_P over a process model P as follows:

$\sigma_P := \langle e_1, \dots, e_k \rangle$ with

$e_i \in \{Start(activity, node, timestamp), End(activity, node, timestamp)\}$ where

- *activity* denotes the activity event e_i is associated with
- *node* denotes the process node an activity is associated with
- *timestamp* represents an abstract timestamp

For our formal interpretation of compliance rules, we need the *activity-oriented view* σ'_P of event-based execution traces σ_P . σ'_P represents the ordered activity executions in σ_P :

$\sigma'_P := \langle a_1, \dots, a_m \rangle$ with

$a_x = (activity_x, node_x, startTime_x, endTime_x)$, $x = 1, \dots, m$ with $\exists e_i, e_j \in \sigma_P$:

- $e_i = Start(activity_x, node_x, startTime_x)$,
- $e_j = End(activity_x, node_x, endTime_x)$,

- $i < j$ and $\nexists e_l \in \sigma_P : i < l < j$ and $e_l = \text{End}(\text{activity}_x, \text{node}_x, \dots)$ and
- $\forall a_r, a_p : r < p \Rightarrow \text{startTime}_{a_r} < \text{startTime}_{a_p}$

Based on the notion of execution traces and activity executions we can provide a default interpretation of compliance rules as follows. The interpretation relates the predicates in the compliance rule formula to activity executions in the execution trace. In the following, we focus on correct execution traces (e.g., each start event has a corresponding end event).

Definition 1 (Interpretation of compliance rules). *Let $A_{\mathcal{T}}$ be the set of activity types of the domain model. Let $\sigma' = \langle a_1, \dots, a_m \rangle$ be an activity-oriented view of an execution trace. Then, the interpretation over σ' over $A_{\mathcal{T}}$ is a tuple $I_{\sigma'} = \langle D_{\sigma'}, d_{\sigma'} \rangle$ with:*

$D_{\sigma'}$ is the domain of the interpretation $I_{\sigma'}$ with $D_{\sigma'} := \{a_1, \dots, a_m\}$

Let further $N_{\sigma'} = \{n | \exists a = (\text{activity}_a, \text{node}_a, \text{startTime}_a, \text{endTime}_a) \in D_{\sigma'} : n = \text{node}_a\}$ be the set of process nodes associated to activity executions in σ' .

$d_{\sigma'}$ is a function interpreting the predicates *ActivityType*, *ProcessNode*, and *Pred*¹ over σ' as follows:

- $d_{\sigma'}(\text{ActivityType}) \mapsto \{(a, A), a = (\text{activity}_a, \text{node}_a, \text{startTime}_a, \text{endTime}_a) \in D_{\sigma'}, A \in A_{\mathcal{T}} \mid \text{activity}_a = A \vee \text{activity}_a \text{ is a subtype of } A\}$
- $d_{\sigma'}(\text{ProcessNode}) \mapsto \{(a, n), a = (\text{activity}_a, \text{node}_a, \text{startTime}_a, \text{endTime}_a) \in D_{\sigma'}, n \in N_{\sigma'} \mid n = \text{node}_a\}$
- $d_{\sigma'}(\text{Pred}) \mapsto \{(a, b), a = (\text{activity}_a, \text{node}_a, \text{startTime}_a, \text{endTime}_a) \in D_{\sigma'}, b = (\text{activity}_b, \text{node}_b, \text{startTime}_b, \text{endTime}_b) \in D_{\sigma'} \mid \text{endTime}_a < \text{startTime}_b\}$

Based on Def. 1 compliance rule formulas can be interpreted over execution traces. Def. 2 provides the formal criteria necessary for compliance verification.

Definition 2 (Satisfaction of compliance rules). *Let P be a process model and let Σ'_P be the set of all activity-oriented views on execution traces of P (i.e., all traces P is able to produce). Let further $\sigma'_P \in \Sigma'_P$ be one activity-oriented view of an execution trace of P .*

We say σ'_P satisfies c (notation: $\sigma'_P \models c$) if and only if:

$$I_{\sigma'_P} \models c.$$

We say P satisfies c (notation: $P \models c$) if and only if:

$$\forall \sigma'_P \in \Sigma'_P \text{ holds } \sigma'_P \models c.$$

To illustrate the formal semantics defined above, consider again compliance rule c_5 (cf. Fig. 5) and the development process (cf. Fig. 1). Based on the execution traces that the development process can produce, Def. 1 can be applied to verify the development process against c_5 . Informally, the straight-forward way to verify compliance rules corresponds to reachability analysis as applied for checking the soundness of process models [9]. In all execution traces of the process, there is an

¹ Constants in a compliance rule formula (i.e., node identifiers in compliance rule instances) are mapped to themselves and hence, are omitted in the interpretation.

occurrence of **end of testing** (node 14). Hence, compliance rule c_5 is activated in all executions of the development process. After **end of testing**, either the upper (case 1) or the lower split branch (case 2) is executed. In the first case, the corresponding execution trace contains the execution of the **approval** (node 15), directly followed by the execution of the **integration** (node 18). Hence, this trace satisfies c_5 . In the second case, the **change process** (node 16) is executed after the **approval** (node 15). However, another **approval** activity (node 17) is executed afterwards, that is directly followed by the **integration**. According to the interpretation from Def. 1, c_5 is also satisfied in the second case.

5 Discussion and Validation

Generally, different scenarios of integrating business processes and compliance rules are conceivable. At the one side of the spectrum, all relevant compliance rules might be integrated (as far as possible) within the process model. This might be achieved by a multitude of partly nested alternative branchings. Full merging of rules into process models could be desirable for scenarios with simple process models and compliance rules that are mandatory for all process models. However, this approach has also several drawbacks. The first one is the possibly enormous complexity of the resulting process models in case of a multitude of compliance rules imposed on them. This can be compared to approaches for process configuration and variants, keeping all variants or rules within one model. As research has discussed, for many scenarios, this approach quickly leads to overly complex process models that cannot be understood by users any longer [10]. In addition, from our practical examples, we know that often not all compliance rules are mandatory, but rather have a suggesting character. Examples include medical guidelines that might be overruled by the doctor at any time. However, if all compliance rules are "hard-wired" within the process models, this optional character gets lost.

On the other side of the spectrum, we could also think of representing process models and compliance rules entirely as rules, for example within rule engines such as ILOG JRules or by using declarative approaches such as ConDec [11]. Aside from the fact that declarative process description can be covered by our approach, it cannot be neglected that in practice, most process models are described and automated using graph-based notations. Specifically, in practice, we will often find a coexistence of business process (models) and compliance rules imposed on them. Another reason for this coexistence is that compliance rules might be introduced after the process model has been designed and enacted.

For these reasons, we mainly aim at supporting the coexistence of process models and compliance rules which complies to approaches that aim at balancing flexibility and control by combining imperative and declarative process description [12]. However, since our logical model is based on execution traces (cf. Sect. 3), SeaFlows can also deal with pure process-model-based and declarative scenarios. In the latter case, execution traces are a suitable representation of process instances defined by following the imposed constraints [13].

5.1 Pattern-Based Validation

We collected recurring compliance rule patterns from literature and modeled them using the SeaFlows compliance rule formalism. Many pattern-based approaches [14,15,13] base their patterns on patterns collected by Dwyer and Corbett [16]. Simple (particularly binary) patterns, such as *global scope presence*, *global scope absence*, *after scope absence*, *before scope absence*, *response*, and *precedence* can be modeled similarly to some of the compliance rules from Fig. 5 (e.g., compliance rule c3 corresponds to *after scope absence*). Hence, we omit the illustration of these patterns due to space limitations. Three advanced patterns [14,16] are depicted in Fig. 7. The *response with absence* rule states that A must be followed by an occurrence of C without B occurring in the meantime. The *precedence with absence* rule states that C must be preceded by A such that no B occurs in between. The *after scope precedence chain* states that the sequence A . . . B must occur between S and a succeeding occurrence of C.

In contrast to approaches based on a limited set of patterns and respective combinations thereof using the logical conjunction (cf. Sect. 6), the SeaFlows approach is compositional. It allows for modeling compliance rules with an

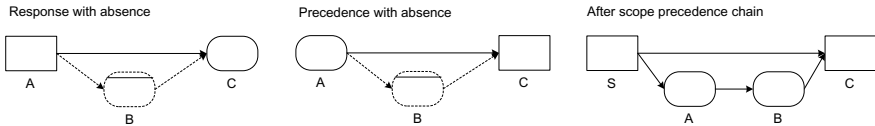


Fig. 7. Advanced compliance rule patterns [14]

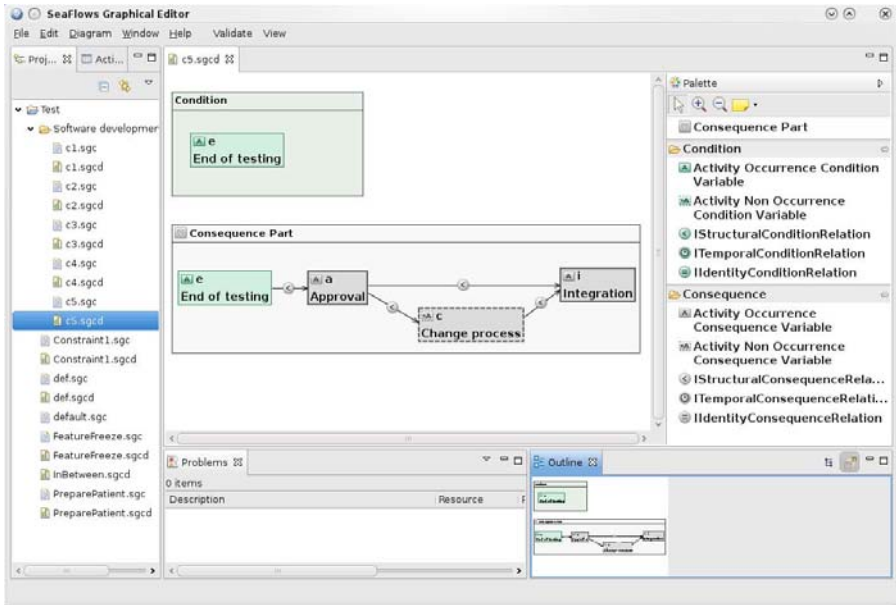


Fig. 8. The SeaFlows graphical compliance rule editor

antecedent and a consequence part. The modeled compliance rule graphs can be automatically mapped to logical formulas (cf. Sect. 4).

5.2 Technical Validation

Fig. 8 shows the SeaFlows compliance rule editor. It allows for separately modeling the antecedent and consequence patterns of compliance rules (depicted in separate boxes). The SeaFlows editor is integrated into the AristaFlow BPM Suite that is based on ADEPT [6]. Each compliance rule node can be assigned an activity type from the AristaFlow Activity Repository. The latter also provides the activity types used to model processes and serves as domain model in our implementation. For convenient compliance rule modeling, the SeaFlows editor allows for modeling parametrized recurring compliance rules patterns. If the user wants to model a compliance rule with the same structure, he may apply the parametrized compliance rule pattern.

6 Related Work

Many approaches have been proposed in literature to model compliance rules. We focus on approaches for modeling compliance rules over the occurrence and ordering relations of activity executions within a process with the goal of process verification. Existing approaches range from rather informal annotations of process models with compliance rules, over formal languages, to visual patterns.

To enable the verification of processes against imposed compliance rules, the latter have to provide formal semantics. This requirement is met by compliance rules specified in formal languages. Among the formal languages proposed for compliance rule modeling, we often come across temporal logics such as linear temporal logic (LTL) or computational tree logic (CTL) [17]. Both are fragments of first-order logic (FOL). Due to its linear time semantics, which is more suitable in the business process context [18], LTL has been clearly preferred over CTL which has branching time semantics. Albeit its expressiveness to model compliance rules with regard to the occurrence and ordering relations of activities, pure LTL has limitations when it comes to the incorporation of context conditions within compliance rules. This is due to the fact that one cannot directly address a particular state in LTL. Hence, to ensure the extendability of our approach, we opted for the formalization in FOL. The formal contract language (FCL), designed to specify business contracts, can also be applied to model compliance requirements [19,20]. However, FCL is based on a rather state-oriented than activity-oriented paradigm. In general, it requires certain skills to model compliance rules using a formal language which might be an obstacle to the practical application of corresponding approaches.

Due to the difficulties of modeling compliance rules using formal languages in practice, many approaches from literature suggest visual notations to hide the formal details from the modeler. In [18], Liu et al. propose a graphical business property specification language (BPSL) that is based on linear temporal logic.

BPSL provides visual notations for logical operators. In addition, it defines dedicated operators for recurring logical patterns. In contrast to our approach, BPSL does not support the explicit structure of antecedent and consequence patterns within a compliance rule.

Other approaches aim at establishing a set of recurring compliance rule patterns. The patterns are either given visually [21,14,11] or are organized in some kind of a pattern ontology [15,22,12]. Generally, these patterns are based on property patterns collected by Dwyer and Corbett [16]. Each pattern, in turn, can be mapped to a logical formula (e.g., in LTL). This enables the formal verification. Clearly, establishing set of rule patterns recurring in a business domain is an effective approach to facilitate compliance rule modeling. This can also be accomplished with our compliance rule language. Although the rule patterns can usually be combined using the logical conjunction, a fixed set of patterns can still be too restrictive for particular application scenarios. In these case, compositional approaches such as our approach are advantageous.

7 Summary and Outlook

We presented an approach to support the design and verification of compliance rules at different abstraction levels ranging from high-level, process independent rules to rules specified over particular process models. The main challenge was to enable the verification of process models against process-independent compliance rules. We solved this by introducing the mechanism of compliance rule instantiation using domain models and showed that instantiation results in many advantages such as easing design, maintenance, and evolution of compliance rules as well as fine-granule compliance reports. Moreover, we introduced an intuitive visualization of compliance rules and instances together with their formal semantics based on FOL and execution traces. Finally, we showed the feasibility of our approach based on a pattern-based validation and by means of our powerful prototype. In future work, we will equip compliance rule graphs with *operational semantics* to enable more efficient compliance checking. In addition, we will further investigate on runtime issues. At runtime when process instances are created from process models and executed, the compliance with imposed compliance rules may change [7]. That is why it can become necessary to monitor the compliance during process execution.

References

1. Sadiq, S., Governatori, G., Naimiri, K.: Modeling control objectives for business process compliance. In: Alonso, G., Dadam, P., Rosemann, M. (eds.) BPM 2007. LNCS, vol. 4714, pp. 149–164. Springer, Heidelberg (2007)
2. Awad, A., Decker, G., Weske, M.: Efficient compliance checking using BPMN-Q and temporal logic. In: Dumas, M., Reichert, M., Shan, M.-C. (eds.) BPM 2008. LNCS, vol. 5240, pp. 326–341. Springer, Heidelberg (2008)
3. Ly, L.T., Rinderle-Ma, S., Dadam, P.: Integration and verification of semantic constraints in adaptive process management systems. *Data and Knowledge Engineering* 64, 3–23 (2008)

4. Filipowska, A., Hepp, M., Kaczmarek, M., Markovic, I.: Organisational ontology framework for semantic business process management. In: BIS 2009. LNBIP, vol. 21, pp. 1–12. Springer, Heidelberg (2009)
5. Kumar, A., Smith, B., Pisanelli, D., Gangemi, A., Stefanelli, M.: An ontological framework for the implementation of clinical guidelines in health care organizations. *Stud. Health Technol. Inform.* 102, 95–107 (2004)
6. Rinderle, S., Reichert, M., Dadam, P.: Flexible support of team processes by adaptive workflow systems. *Distributed and Parallel Databases* 16, 91–116 (2004)
7. Ly, L.T., Rinderle-Ma, S., Göser, K., Dadam, P.: On enabling integrated process compliance with semantic constraints in process management systems. *Information Systems Frontiers* (2009) (accepted for publication)
8. van der Aalst, W., et al.: Prom 4.0: Comprehensive support for real process analysis. In: Kleijn, J., Yakovlev, A. (eds.) ICATPN 2007. LNCS, vol. 4546, pp. 484–494. Springer, Heidelberg (2007)
9. van der Aalst, W.: Verification of workflow nets. In: Int'l Conf. on Application and Theory of Petri Nets, pp. 407–426 (1997)
10. Hallerbach, A., Bauer, T., Reichert, M.: Managing process variants in the process lifecycle. In: Proc. ICEIS 2008, pp. 154–161 (2008)
11. Pesic, M., Schonenberg, M., Sidorova, N., van der Aalst, W.: Constraint-based workflow models: Change made easy. In: Meersman, R., Tari, Z. (eds.) OTM 2007, Part I. LNCS, vol. 4803, pp. 77–94. Springer, Heidelberg (2007)
12. Sadiq, S., Orlowska, M., Sadiq, W.: Specification and validation of process constraints for flexible workflows. *Inf. Syst.* 30, 349–378 (2005)
13. Pesic, M.: Constraint-Based Workflow Management Systems: Shifting Control to Users. PhD thesis, Eindhoven University of Technology (2008)
14. Awad, A., Weske, M.: Visualization of compliance violation in business process models. In: Proc. BPI 2009 (2009)
15. Yu, J., Manh, T.P., Hand, J., Jin, Y.: Pattern-based property specification and verification for service composition. CeCSES Report SUT. CeCSES-TR010, Swinburne University of Technology (2006)
16. Dwyer, M.B., Avrunin, G.S., Corbett, J.C.: Patterns in property specifications for finite-state verification. In: Proc. ICSE 1999, pp. 411–420 (1999)
17. Ghose, A., Koliadis, G.: Auditing business process compliance. In: Krämer, B.J., Lin, K.-J., Narasimhan, P. (eds.) ICSOC 2007. LNCS, vol. 4749, pp. 169–180. Springer, Heidelberg (2007)
18. Liu, Y., Müller, S., Xu, K.: A static compliance-checking framework for business process models. *IBM Systems Journal* 46, 335–361 (2007)
19. Governatori, G., Hoffmann, J., Sadiq, S., Weber, I.: Detecting regulatory compliance for business process models through semantic annotations. In: Proc. BPD 2008 (2008)
20. Lu, R., Sadiq, S., Governatori, G.: Compliance aware process design. In: ter Hofstede, A.H.M., Benatallah, B., Paik, H.-Y. (eds.) BPM Workshops 2007. LNCS, vol. 4928, pp. 120–131. Springer, Heidelberg (2008)
21. van der Aalst, W., Pesic, M.: DecSerFlow: Towards a truly declarative service flow language. In: Bravetti, M., Núñez, M., Zavattaro, G. (eds.) WS-FM 2006. LNCS, vol. 4184, pp. 1–23. Springer, Heidelberg (2006)
22. Namiri, K., Stojanovic, N.: Pattern-based design and validation of business process compliance. In: Meersman, R., Tari, Z. (eds.) OTM 2007, Part I. LNCS, vol. 4803, pp. 59–76. Springer, Heidelberg (2007)

Success Factors of e-Collaboration in Business Process Modeling

Peter Rittgen

Vlerick Leuven Gent Management School, Reep 1, 9000 Gent, Belgium &
University of Borås, Allégatan 1, 50190 Borås, Sweden
peter.rittgen@vlerick.com

Abstract. We identify the success factors of collaborative modeling of business processes by a qualitative analysis of the experiences of participants in group modeling sessions. The factors and their relations form a preliminary theoretical model of collaboration in modeling that extends existing models. The insights from this guided the improvement of a group modeling method and tool support which are in turn relevant outcomes of the design part of this study. We show in field experiments that the new method outperforms the conventional one.

1 Introduction

The purpose of this paper is to identify factors of a modeling method that have an impact on the quality of the model and the modeling process. For this purpose we draw on the existing literature on electronic collaboration and extend and refine the factor models that can be found there. To do this, we study collaborative modeling sessions and interview the participants to elicit possible factors. As we assume that participants can make valid assertions about modeling we have to show whether these factors are indeed relevant. We did so by incorporating them into a modeling method and accompanying tool to see whether using the factors really improves collaborative modeling. In detail we proceed as follows.

We describe the results from 4 case studies and eight controlled field experiments conducted at four organizations. The aim of the case studies is to find factors that determine the success of business process modeling sessions. This is done *ex post* with the help of semi-structured interviews with the participants of such sessions. The details of that procedure and its results are described in section 4. We identify eleven factors and their relations in the areas facilitation, motivation, group & team, and support. The factor model is provided in section 5.

The sessions were done to improve a process modeling method and a supporting tool. Method development was driven by the success factors (6.2). Section 6 also shows the lessons we learnt with respect to the necessity of a co-evolution of method and tool and the current status of the process modeling method.

The field experiments aim to validate the tool-supported method by comparing it to the conventional method used in many organizations. This is done by a quantitative analysis of questionnaire data. The results are reported in section 7. The factor model itself is not validated but as the factors controlled method development, the validation

of the latter is an indication of the usefulness of the factor model. In the following two sections we describe related research and the research approach that we have taken.

2 Related Research

One relevant area is that of electronic meeting systems (EMS). They are computer systems designed to support electronic meetings (also called group support systems). Some collaboration factors were identified in the Focus Theory of Group Productivity [1] that refers to cognitive processes in group members. EMS's have been found to decrease the demand for **attention** and increase **productivity**.

The proponents of electronic meeting support claim that EMS's have a positive effect on the outcome of meetings. There is general consent to that but there is some debate as to the conditions for successful EMS use. Most researchers think that an EMS has a moderating impact on the process, which then improves the output [2]. We agree and put the focus of our research on the modeling process.

There are two mechanisms by which EMS's can make a difference over face-to-face meetings: **anonymity** and **simultaneity**. The former means that utterances cannot be attributed to their originator. It is assumed that this will lead to additional ideas that people would otherwise not have felt comfortable to share. Simultaneity means the possibility of all team members to utter their ideas at the same time. In conventional meetings only one person can speak at a time which leads to air time fragmentation and thereby to a loss of ideas.

Both anonymity and simultaneity are supposed to moderate group process gains (such as more objective idea evaluation, learning from others, etc.) and process losses (such as air time fragmentation, production blocking, conformance pressure, free riding, evaluation apprehension, etc.). These process characteristics are assumed to determine outcome factors such as efficiency, effectiveness, satisfaction, consensus and so on [3].

Experimental results were inconclusive [4] but case and field studies showed the contributions of EMS's in reducing meeting time and increasing meeting productivity [5]. The application of EMS's to group modeling has also been studied [6-8] with similar results.

The problem with EMS's is that they only support divergent processes but process modeling also requires convergent processes such as **filtering** and **integrating** alternatives. These aspects are addressed in the COMA approach (Collaborative Modeling Architecture).

Another closely related research area that can contribute success factors is that of model or modeling quality. E.g. [9] suggests that modeling is a mapping procedure where specific choices for the representation of reality have to be made along the way. We consider modeling rather as a social construction process, though, which offers many more choices. [10] proposes to the use of cognitive mapping techniques as a quality factor. So the method itself can also be a factor. We have restricted our attention on the factors within a method. This way we can engineer in the long term an optimal method instead of making a choice between arbitrary given methods.

3 Research Methodology

Process modeling (or modeling, for short) is a collaborative activity that involves a number of stakeholders possessing the required knowledge about the process or goals of the organization. Although modeling is in principle always collaborative we call it collaborative modeling because we want to stress the importance of a true and close collaboration of the participants in group modeling (we come back to this later). In this section we present the theories that have an impact on collaborative modeling and the approach we have taken to conduct the research.

3.1 Relevant Theories

As modeling is a collaborative activity it is natural to first take a look at the literature that deals with collaboration in general. In the section on related research we have already pointed out the major works that deal with this topic. But here we consider only such literature that provides some theoretic framework for explaining the factors that determine the quality of collaboration.

An early milestone of collaboration was the work of Osborn on brainstorming [11]. He claimed that the critical factors for successful idea creation are **quantity of ideas** (divergent production), **absence of criticism** (to stimulate ideas), encouragement of **unusual ideas** (out of the box), and **idea combination**. Although these factors seem logical enough, subsequent research has not found clear evidence that brainstorming provides better ideas than other techniques. This ‘contradiction’ was solved by [12, 13] who found that the advantages were thwarted by **production blocking**, **evaluation apprehension**, and **social loafing**.

Production blocking means that a person that explains his idea distracts other group members from producing own ideas. Evaluation apprehension [14] means that participants are reluctant to share their ideas because they fear social punishment. Social loafing refers to a group member spending little effort relying on others.

One of the most elaborated theories on collaboration is Focus Theory by Briggs [1]. It studies the **individual effort** a participant spends on collaboration assuming that it can be spent on communication, deliberation, and information access, but not at the same time. Productivity in one area therefore limits the effort that can be spent on the others. The overall productivity depends on high productivity in all three areas which constitutes a kind of a vicious circle. Briggs did intensive behavioral research to concretize the factors that have an impact on individual effort.

In short his model relates the factors perceived difficulty of task, desired certainty of success, perceived effort required, perceived effort available, self efficacy, desire for goal, individual effort and group productivity. Briggs mentions some motivational factors that are important for successful collaboration: desired certainty of success and desire for goal. These intrinsic output-oriented factors are not sufficient for successful process modeling, though, as the output (model) is not desirable for the participants.

We have therefore turned our attention to the literature in psychology dealing with collaboration. There a distinction is made between intrinsic and extrinsic motivation, i.e. motivation that originates in us and motivation that is imposed on us by people in our environment. In our study we consider both of them. In our study we have also elicited further factors and classified them.

Another source of knowledge that for our research is Consensus Building Theory [15]. It specifies a few basic rules for collaboration that should lead to consensus, which is of particular importance in process modeling. Among them are the use of a professional and neutral facilitator, the presence of representatives of all stakeholders and the existence of ground rules to guide participants (i.e. a method).

3.2 Research Approach

Our research mainly follows the principles of design science and combines it with action research [16, 17] to study the collaborative creation of process models. Design science is a framework that emphasizes the development and improvement of artifacts in cycles, in our case a method and a tool.

The development of the artifacts was based on business needs derived from an empirical studies and applicable knowledge from theory [18, 19]. Validation and further improvement of the artifacts was studied in [20] in the form of a comparative experiment.

The purpose of this paper is not so much the relevance cycle which is documented in [18, 20], but the rigor cycle, i.e. the contributions to the knowledge base. These contributions take on the form of important factors and the way they are related to each other. The general factors in e-collaboration were derived from relevant theories as outlined in 3.1. Special factors for e-collaboration in modeling were derived from a qualitative analysis of an interview study (see 3.3, 4) and confirmed in quantitative comparative experiments (see 7). The result is a qualitative model of these factors.

3.3 Data Gathering

The data was collected in four case studies that were carried out in a sequential order so that the lessons learned in one could be used to adapt the methodology and/or tool before going into the next one (design cycle). The cases were done (in this order): at a large psychiatric hospital (PH), a large insurance company (IC), a medium-sized bio-engineering laboratory (BL), and a large public administration (PA). In total we did 5 modeling sessions lasting between 0.5 and 1.5 days. We did a to-be model for all cases except IC, and an as-is model for PH and IC. The teams had between 7 and 12 participants that usually worked in groups of two.

Data was gathered in the form of ex-post qualitative interviews with participants that were normally performed directly after the respective session but not later than on the morning after. The interviews were semi-structured but we allowed interviewees to elaborate on a point or to mention other issues. We also asked questions that came to our mind while listening to the respondents. All 37 interviews were conducted by the same researcher. The major questions were:

- Which factors are relevant to achieve a good model?
- Which factors are relevant for making modeling worthwhile?
- Which factors are relevant for making the output more desirable?
- What are the advantages of COMA over conventional modeling?
- What are the advantages of conventional modeling over COMA?

The answers from the respondents were coded and specific factors were taken up in the list of relevant factors when they were mentioned by at least half of the group

members in a least 3 of the 5 sessions. We grouped the factors into facilitation, motivation, group & team, and support factors. For further details see section 4.

4 Data Analysis

The interview data was analyzed as described above. The resulting factors have been categorized into 4 categories: facilitation factors, motivation factors, group & team factors, and support factors. The following sections are devoted to each category and discuss the results from the interview in the light of relevant theories.

4.1 Facilitation Factors

Consensus building theory mentions that the facilitator has a significant impact on the collaboration output. He should therefore be professional and neutral. Professionalism can easily be ensured by hiring a consultant who does that work on a daily basis. But that does not automatically ensure that he is neutral, even if he has been recruited externally, i.e. not within the targeted organization. **Facilitator bias** is always present because the facilitator has to translate the input from the participants into a process model. His perceptions will therefore influence the way in which the model is built.

With conventional techniques such as brown paper this can hardly be avoided as they leave the overall responsibility for the model to the facilitator who consequently plays a role that is too dominant. Reducing facilitator dominance can be achieved by **participant involvement** in model building. This also frees facilitator resources that to be used elsewhere and removes the facilitator bottleneck that prolongs modeling sessions. But according to conventional wisdom this is impossible because modeling requires a highly skilled modeling expert.

Our experience in the cases has shown the contrary to be true. Unskilled people can develop complex business process models after having played a modeling game for about an hour. These models are not always 100 % perfect but they usually require very little re-working, mostly to fix poorly structured layouts.

From this we conclude that facilitation has to be seen from a different perspective: the facilitator should not elicit knowledge and transform it into a model, but he should rather support people in modeling themselves and help them with the **integration** of the different views. This approach was tested in many cases and was rated by the participants as being both a better way of modeling and delivering a better result (see section 7).

4.2 Motivation Factors

A fundamental problem with collaborative modeling is the fact that participants of such an exercise have no intrinsic motivation for the result itself. Most people are not interested in the model and do not see a need for it. But extrinsic motivation implies the risk of shirking (i.e. underperforming when not noticed). Consequently intrinsic motivation seems more promising, but instead of on the model we have to focus on the modeling process. **Motivation for modeling** is therefore a key factor. A powerful intrinsic motivation is that of gaming. The use of gaming for modeling has already been suggested by [21].

People gladly do a tough job if it comes in the disguise of a game. Think of PC gamers who spend days and nights without monetary reward just to reach the next level. In terms of the modeling process we therefore need to increase the motivation for modeling with the help of gaming elements. We have done so by introducing a **competition** that consists of scoring models followed by the nomination of a winner (i.e. the best model). In practice this means that models are not only developed by participants but the “players” also score one another.

The latter can also be interpreted as a form of extrinsic motivation. In psychology this is called social comparison. If I know that I will be judged by my peers I will put much more effort into model development because nobody wants to be a loser.

4.3 Group and Team factors

For practical reasons and to improve group productivity, the whole group is split into teams of 2. For the whole group it is paramount that members have complementary knowledge. The obvious reason for this is that it increases the overall unique process knowledge of the team and hence the richness of knowledge they can contribute to the model. Together the group members should possess all required knowledge.

But there is also a more subtle mechanism at work that requires the division into smaller, **complimentary** teams. Team members with complimentary, but overlapping knowledge are likely to have some conflicting views on the process. Solving these conflicts with the help of a constructive dialogue between two team members is much easier than in the larger group where the air time is much more fragmented, which prolongs conflict resolution. There is also a higher risk for the conflict to escalate when even more different opinions collide.

In short, minor conflicts are solved more effectively and efficiently in a small team and will then not escalate to the group level. Fundamental problems will surface to the group level in the consolidation phase, which is the forum where problems of this kind need to be addressed. This facilitates **consensus** building.

But dividing a group into small teams is not only a way of introducing two conflict handling layers to speed up the modeling process. Discussions in a small team also stimulate the team’s creativity (four eyes see more than two) and thereby the richness and quality of their proposal and of the integrated model.

Another important factor is the **degree of participation**. It indicates the relative number of group members that are actually active in a session. A higher degree raises model quality by making models richer but lowers consensus by adding views.

4.4 Support Factors

When it comes to the support we mainly think of some computerized tool that can increase the motivation for modeling and the model quality. We have found four areas in which support is needed. They are discussed in the following sub-sections.

Modeling support

First of all we have to take a look at modeling itself. If participants are supposed to model we obviously need some computerized tool support for them as a number of paper versions could hardly be integrated in a structured way. This means that the tool, called COMA, has to provide a model editor that is simple enough to allow unskilled modelers to draw their business processes without the need for major education.

We have used a reduced version of the activity diagrams of UML that provides only the most basic elements: activities, simple flows, decision points, parallel processes, actors (as swim lanes), and notes (for comments and issues). Any other modeling language (e.g. the Business Process Modeling Notation, BPMN) would work just as well as long as the participants are not drowned in too many features of the language.

In a simple modeling game that goes through all the steps of the modeling procedure but uses a simple process known to everybody (getting cash from a teller machine) we make the participants acquainted with the language, the tool and the procedure. They learn them as a side-effect while trying to win the prize for the best model (in some cases we actually handed out a small prize, in others the winners were just applauded).

Competition support

Just telling the participants that a winner will be nominated is not enough as an incentive. If, for example, a jury would determine the winner, or perhaps even the facilitator himself, the decision process is not transparent for the participants and they might think that other factors than model quality would influence the decision. This easily leads to a situation of discouragement.

A jury would also be hard to find because the experts in that domain already sit around the table. The facilitator is the worst judge because he typically possesses no knowledge whatsoever about the targeted process (which otherwise is a plus because it makes him more neutral).

We have therefore decided to introduce a scoring of each model proposal by the other modeling teams. After the complete scoring round of all proposals the facilitator shows the whole group the average scores of all teams as bars of different sizes and numbers. This is an exciting moment for the teams as they get to know their own scores and how they relate to the others.

Being judged by their own peers (often colleagues) makes them put as much effort into the modeling as they can, and that is precisely what we want to achieve: the best possible effort by all group members. Nobody can hide behind more active group members as everybody has to deliver a model and every model is scored.

The result of the scoring round, which usually takes ten minutes, is not only a winning team but also a winning model. This will be the basis for all further development as the highest overall score clearly indicates that this model has the strongest support and therefore the best chance of creating consensus. It cannot be taken as the final version, though, as some details might still be missing or misrepresented. This needs to be settled in a consolidation step.

Communication support

Verbal communication is essential in modeling [22] as the process of creating a semi-formal representation is embedded in a natural language conversation about this representation. If the modeling session is organized as a workshop with all participants being in the same room at the same time, most of the conversation can take place in the usual face-to-face manner.

If the group members are physically distributed but still working synchronously, the face-to-face talk can be replaced by a video or teleconference. But many organizations

prefer the modeling work to be done off-line (asynchronously) so that the involved employees can do the work at a time that is convenient for them. If the participants are managers it is also very difficult to find empty time slots that coincide in everybody's agenda. This delays modeling work unnecessarily.

There is thus a need for asynchronous communication support. Such technologies do exist, of course, in the form of email or even voicemail but the conversation is related to a specific part of a particular model so the communication support has to be linked to the modeling support.

Information access support

According to focus theory information access is one of the three building blocks of collaboration. Collaborative modeling therefore also needs to give participants access to vital information. The most important information is of course stored in the model itself, but open issues expressed in natural language are also relevant information.

Both kinds of information should be accessible via a collaborative modeling tool to facilitate distributed and asynchronous work. This means that model proposals of one team can be opened by other teams to look at but also to reuse elements from other models in your own. When it comes to open issues the collaborative modeling tool has to provide a function that allows participants to log written comments with respect to another team's proposal they are currently reviewing.

This is further elaborated in section 6.2.

5 An Initial Model of Factors in Collaborative Modeling

If we summarize the findings from the empirical study we arrive at the factor model that is shown in Fig. 1.

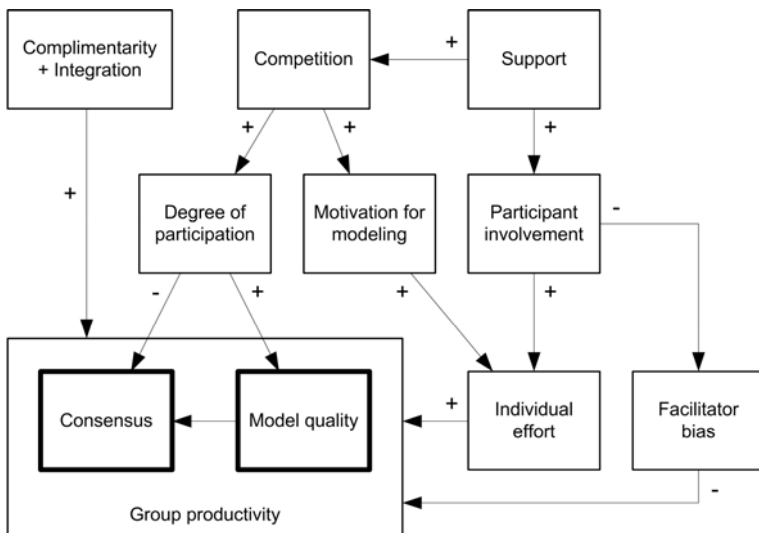


Fig. 1. Our model of collaboration factors in modeling

Please note that the model is not a general collaboration model but restricted to the specific case of collaborative modeling. It is an extension of the existing models that are connected via the variables *Individual effort* and *Group productivity*, which are present in all of them. In our case the latter consists of model quality and consensus, which are both important products of the modeling process.

It should be noted that the factor model is the result of qualitative research. The model needs to be validated in a quantitative study that also determines the strength of the links between the factors. Nevertheless, the results from the qualitative study have allowed us to draw some interesting conclusions and to develop a modeling method and tool that make use of the identified factors in order to improve the modeling process as well as its result, the model.

6 Lessons Learned

The lessons that we learned from the case studies and interviews about the relevant success factors provided valuable input for the development of a modeling method that incorporates these factors. If the elicited factors do have an impact on the quality of modeling and the model then this should lead to a more successful method also. The following sub-sections describe how the method and tool were developed and section 7 compares it to the most common existing method.

6.1 Co-evolution of Method and Tool

The first lesson was that the modeling method strongly depends on modeling steps that some tool can support. But the development of the tool is of course also driven by the requirements of the method. The latter is the normal way of developing methods and tools. But the former needs explanation. Why should the tool drive the method? The reason is that certain method steps cannot be performed (or not efficiently) with conventional tools but require sophisticated tool support that can only be provided by a computerized tool. They might therefore be excluded from the method.

But it is precisely these additional or modified steps that make the method superior to others as seen in a study comparing the COMA method to the conventional method based on brown paper. This is shown in section 7.

The conclusion we can draw from these experiences is that the development of method and tool are so closely related that you cannot separate the one from the other. Instead the design of both artifacts is done together. We call this co-design or co-evolution of method and tool.

We see the tool as an inseparable part of the method which means that an empirical evaluation of only one is not possible. We have therefore set up an experiment to compare the COMA method + tool against the conventional method and tool (brown paper). This is described in section 7.

6.2 Towards a Collaborative Process Modeling Method

Another important lesson told us that the new factors we have identified make it necessary to develop a method that exploits these factors in order to improve both the modeling process itself (in terms of participant satisfaction) and its result (the model).

The method has been developed in a design cycle where we started with the conventional method and an existing modeling tool (UML Pad). We added, modified or extended steps in the method that we thought would implement a certain factor. At the same time we introduced the corresponding functionality in the tool. To check this we have used the modified method and tool in action research to test whether it works and to get feedback on the implemented factor, e.g. Did the motivation for modeling increase? This procedure was iterated factor by factor and so far we have reached a modeling method that contains 7 steps. Modeling is done in teams of 2 people. Each team has a computer with the COMA tool (*factor support*). This tool basically offers three tabs: One for editing your own model; one with the latest version of the group model; and one that allows you to open models by other teams. The facilitator is equipped with the same tool running in facilitator mode and being connected to a beamer that projects the screen content on a blank wall that is visible to everybody.

Step 0:

Introduce participants to the COMA method and tool by playing a simple game of 1-2 hours. Instructions for the game can be found on the Internet, both for the participants and the facilitator (anonymized). The facilitator divides the whole group into complimentary teams of two supported by the responsible process or change manager that knows the group members (*factor complimentary teams*).

Step 1:

Collect activities (as-is) or brainstorm for them (to-be) (*similar to the conventional method*). Each team of two participants enters into the model editor an activity box for each activity labeled with the name of the activity using the usual form *verb + noun*. The time for this step is set in advance by the facilitator who also explains the scope (where does the process start and end) and the granularity (how detailed should the model be, e.g. in terms of a rough upper limit for the number of activities). At the end of the time the teams post their models, i.e. they make a proposal. The facilitator checks that all teams have made their proposals (*factor degree of participation*).

Step 2:

Score the activity model (*factor competition*). Each team opens the proposals of each other team, one by one and gives a score on a scale from 0 to 10. The facilitator has to specify the scoring criterion: E.g. compare the other model with your own model w.r.t. completeness assuming a score of 5 for your own. At the end the facilitator will show the average scores of all teams on the big screen. The model with the highest score is chosen and becomes the new version of the group model.

Step 3:

Consolidate the best proposal (*factor integration*). In this step the best proposal is adapted according to input from the participants. This is similar to model elicitation in conventional sessions but the facilitator here has already an almost finished model available and can concentrate on fine-tuning it. The model is visible on the big screen (and also the facilitator's changes) and participants can discuss in the large group to resolve remaining issues. In most cases there is little work left to do.

Step 4:

Structure the activities (*similar to the conventional method*). The teams copy the final set of activities into their editors and add the control flow and swim lanes (for actors). The facilitator has to set an appropriate time when the teams post their proposals.

Step 5:

Score the process models (*factor competition*). This step proceeds exactly as step 2 but the scoring criterion might be different depending on the particular situation. For an as-is model completeness and correctness are appropriate even here.

Step 6:

Consolidate the best process model (*factor integration*). This step proceeds as step 3. The facilitator helps solve critical issues and improving the layout.

We have now described the method and which of the factors led to which of the steps being affected. What remains to be shown is whether this new method, although it worked very well in the design cases, does really outperform the conventional way of producing models. This is done in the next section.

7 Comparative Experiments

We have carried out comparative experiments at IC that involved 8 groups with a total of 83 members. In each experiment we had the same group model equally sized parts of the same process. The morning session was performed using COMA; in the afternoon the conventional approach was used. This order was chosen to avoid that better scores for COMA could be attributed to the learning curve. Immediately after the afternoon session each participant had to fill out a questionnaire that asked for a ranking of COMA and conventional method with respect to 12 quality categories concerning both the model and the modeling process.

The COMA sessions at IC were performed as described above.

The conventional sessions at IC proceeded as follows. The major tools used are a big brown paper attached to the wall and a number of differently colored post-it notes. In the beginning the facilitator asked the participants for the roles that are involved in the business process and made a swim lane for each of them on the brown paper.

He then asked the group members to name the steps (activities) in the process and wrote each one down on a post-it note that was attached to an empty space on the brown paper, i.e. not in a particular swim lane. When all activities had been named the facilitator would take one note after the other and ask the participants in which swim lane it belongs. It was fixed there in an arbitrary position.

After having placed all notes the facilitator elicited the order of activities starting from the first step. He drew an arrow from an activity to its successor and he also introduced decision points where necessary. As the notes were not ordered within a swim lane the resulting diagram became messy and consolidation of the model proved often to be a difficult and time-consuming task.

After the session and off-line, the facilitator entered the diagram into a professional modeling tool and sorted the activities as well as the arrows to get a readable layout.

The quality categories for our study are based on the ones used in a similar study [23]. They have been adapted to reflect the differences between system dynamics models and business process models.

The respondents had to check the approach that they experienced as superior in each of the quality categories. The conventional method was coded as 0, COMA as 1. We performed the non-parametric χ^2 -test to find out whether there is a significant surplus of 0's or 1's in the respective category.

The χ^2 -statistic measures the fitness of an observed frequency with a theoretical distribution, in our case a uniform distribution with 2 outcomes (50% 0's and 50% 1's) if the outcome is random. For a degree of freedom of 1 (2 outcomes) and a significance level of 5% the critical value is 3.84. If χ^2 is lower than that the outcome is random, otherwise it is significant.

The result is shown in Table 1 and Table 2. BP means 'business process'.

Table 1. χ^2 for the first six of twelve quality categories

| | More Insight into BP | Quicker Insight into BP | Better Commu- nication | Better Shared View | Quicker Shared View | More Commitment |
|----------|----------------------------|-------------------------------|---------------------------|--------------------------|---------------------------|--------------------|
| χ^2 | 6.373 | 0.301 | 0.301 | 24.398 | 0.976 | 0.590 |

Table 2. χ^2 for the second six of twelve quality categories

| | Quicker Commitment | Stronger Influence | More Ownership | Better Result | Better Way Of Working | More Progress |
|----------|-----------------------|-----------------------|-------------------|------------------|--------------------------|------------------|
| χ^2 | 0.108 | 41.940 | 18.325 | 22.277 | 28.928 | 11.578 |

Significant results are highlighted in bold. Some them also reach a significance of 1%. Out of the 12 quality categories COMA achieves a significant improvement in 7.

The improved categories are:

- More insight into the business process
- A better shared view
- A stronger influence of individuals on the result
- A stronger feeling of ownership of the model
- A better result (model)
- A better way of working
- More progress in modeling

In the other categories COMA scored equally well as the conventional method.

8 Conclusions

We set out to identify the relevant factors for successful modeling in the area of business process modeling. We did so by performing four case studies and collecting

qualitative data on the factors by means of semi-structured interviews. While these factors and their relations also form important insights in themselves, the major reason for looking for them was for the purpose of guiding the design of artifacts that support group modeling.

In a design cycle we therefore developed a business process modeling method and a supporting tool. Both were employed in action research in many cases, amongst them the ones of this study, to ensure continuous improvement. To see whether the current artifacts really represent an improvement over existing methods we performed field experiments. In each we used the same modeling group and the same business process to ensure that the modeling method is indeed the only different factor. To exclude a learning curve bias we always conducted the COMA session first.

In seven out of twelve quality categories COMA significantly outperformed the conventional method, while there were no differences in the other five. This also lends credence to the factor model, which we used in building the artifacts. Nevertheless, the factor model cannot be considered validated knowledge yet as a quantitative study has to prove the strength of the identified links.

References

- [1] Briggs, R.O.: *The Focus Theory of Group Productivity and its Application to the Design, Development, and Testing of Electronic Group Support Technology*. Management Information Systems Department, PhD. University of Arizona, Tucson (1994)
- [2] Reinig, B.A., Shin, B.: The Dynamic Effects of Group Systems Support on Group Meetings. *Journal of Management Information Systems* 19, 303–325 (2002)
- [3] Fjermestad, J., Hiltz, S.R.: Group Support Systems: A Descriptive Evaluation of Case and Field Studies. *Journal of Management Information Systems* 17, 115–159 (2001)
- [4] Fjermestad, J., Hiltz, S.R.: An assessment of group support systems experimental research: Methodology and results. *Journal of Management Information Systems* 15, 7–149 (1999)
- [5] Nunamaker, J.F.J., Briggs, R.O., Mittleman, D., Vogel, D.R.: Lessons from a dozen years of group support systems research: a discussion of lab and field findings. *Journal of Management Information Systems* 13, 163–207 (1997)
- [6] Dean, D., Orwig, R., Lee, J., Vogel, D.: Modeling with a group modeling tool: group support, model quality, and validation. In: *Proceedings of the Twenty-Seventh Hawaii International Conference on System Sciences*. Information Systems: Collaboration Technology Organizational Systems and Technology, January 4-7, vol. 4, pp. 214–223. IEEE Computer Society Press, Los Alamitos (1994)
- [7] Dean, D.L., Lee, J.D., Orwig, R.E., Vogel, D.R.: Technological Support for Group Process Modeling. *Journal of Management Information Systems* 11, 43–63 (1994)
- [8] Dean, D.L., Orwig, R.E., Vogel, D.R.: Facilitation Methods for Collaborative Modeling Tools. *Group Decision and Negotiation* 9, 109–127 (2000)
- [9] Mendling, J., Recker, J.: Extending the Discussion of Model Quality: Why Clarity and Completeness may not always be enough. In: *19th International Conference on Advanced Information Systems Engineering (CAiSE 2007)*, pp. 109–121 (2007)
- [10] Siau, K., Tan, X.: Improving the quality of conceptual modeling using cognitive mapping techniques. *Data & Knowledge Engineering* 55, 343–365 (2005)
- [11] Osborn, A.F.: *Applied imagination: Principles and procedures of creative problem solving*. Charles Scribner's Sons, New York (1963)

- [12] Diehl, M., Stroebe, W.: Productivity loss in brainstorming groups: toward the solution of a riddle. *Journal of Personality and Social Psychology* 53, 497–509 (1987)
- [13] Diehl, M., Stroebe, W.: Productivity loss in idea-generating groups: tracking down the blocking effect. *Journal of Personality and Social Psychology* 61, 392–403 (1991)
- [14] Cottrell, N.B.: Social Facilitation. In: McClintock, C. (ed.) *Experimental Social Psychology*, pp. 185–236. Holt, Rinehart & Winston (1972)
- [15] Susskind, L.: Arguing, Bargaining and Getting Agreement. In: Rein, M., Goodin, R., Moran, M. (eds.) *Oxford Handbook of Public Policy*. Oxford University Press, London (2006)
- [16] Hevner, A.R., March, S.T., Park, J., Ram, S.: Design Science in Information Systems Research. *MIS Quarterly* 28, 75–105 (2004)
- [17] March, S.T., Smith, G.: Design and Natural Science Research on Information Technology. *Decision Support Systems* 15, 251–266 (1995)
- [18] Rittgen, P.: Collaborative Modeling - A Design Science Approach. In: 42nd Hawaii International Conference on System Sciences (HICSS-42), Waikoloa, Big Island, Hawaii, USA, January 5-8, p. 10. IEEE Computer Society, Los Alamitos (2009)
- [19] Rittgen, P.: Negotiating Models. In: Krogstie, J., Opdahl, A.L., Sindre, G. (eds.) *CAiSE 2007 and WES 2007*. LNCS, vol. 4495, pp. 561–573. Springer, Heidelberg (2007)
- [20] Rittgen, P.: Collaborative Modeling of Business Processes - A Comparative Case Study. In: 24th Annual ACM Symposium on Applied Computing, March 9-12, pp. 225–230 (2009)
- [21] Hoppenbrouwers, S.J.B.A.: Community-based ICT Development as a Multi-Player Game. In: *What is an Organization? Materiality, Agency and Discourse*. International Communication Association (2008)
- [22] Frederiks, P.J.M., van der Weide, T.P.: Information Modeling: the process and the required competencies of its participants. *Data & Knowledge Engineering* 58, 4–20 (2006)
- [23] Rouwette, E.A.J.A., Vennix, J.A.M., Mullekom, T.v.: Group model building effectiveness: a review of assessment studies. *System Dynamics Review* 18, 5–45 (2002)

Beyond Process Mining: From the Past to Present and Future

Wil M.P. van der Aalst¹, Maja Pesic¹, and Minseok Song²

¹ Department of Mathematics and Computer Science,
Eindhoven University of Technology,
P.O. Box 513, NL-5600 MB, The Netherlands

w.m.p.v.d.aalst, m.pesic@tue.nl

² School of Technology Management

Ulsan National University of Science and Technology,
100 Banyeon-ri, Ulju-gun, Ulsan Metropolitan City, 689-798, South Korea
minseok.song@gmail.com

Abstract. Traditionally, process mining has been used to extract models from event logs and to check or extend existing models. This has shown to be useful for improving processes and their IT support. Process mining techniques analyze historic information hidden in event logs to provide surprising insights for managers, system developers, auditors, and end users. However, thus far, *process mining is mainly used in an off-line fashion and not for operational decision support*. While existing process mining techniques focus on the process as a whole, this paper focuses on individual process instances (cases) that have not yet completed. For these running cases, process mining can be used to *check conformance, predict the future, and recommend appropriate actions*. This paper presents a framework for operational support using process mining and details a coherent set of approaches that focuses on time information. *Time-based operational support* can be used to detect deadline violations, predict the remaining processing time, and recommend activities that minimize flow times. All of this has been implemented in *ProM* and initial experiences using this toolset are reported in this paper.

1 Introduction

Processes are everywhere. Organizations have business processes to manufacture products, provide services, purchase goods, handle applications, etc. Also in our daily lives we are involved in a variety of processes, for example when we use our car or when we book a trip via the Internet. More and more information about these processes is captured in the form of *event logs*. Contemporary systems, ranging from copiers and medical devices to enterprise information systems and cloud infrastructures, record massive amounts of events. These events can be used to make processes visible. Using *process mining* techniques it is possible to discover processes [2,5]. Moreover, event logs can be checked to assess conformance/compliance with respect to defined processes and process models can be modified and extended using process mining techniques. This provides the

insights necessary to manage, control, and improve processes. Process mining has been successfully applied in a variety of domains ranging from healthcare and e-business to high-tech systems and auditing.

Despite the success of process mining, a limitation is that existing techniques are rarely used in an operational setting. Process mining is mainly used in an offline setting where historical information is analyzed without considering the *running cases*, i.e., instances of the process that have not completed yet. The goal of this paper is to demonstrate that process mining techniques can be used for *operational decision support*. Based on process models, either discovered through process mining or (partly) made by hand, we can (a) *check*, (b) *predict*, and (c) *recommend*. We can “replay” a running case on the process model and check whether the observed behavior fits. The moment the case deviates, an appropriate actor can be alerted. The process model based on historic data can also be used to make predictions for running cases, e.g., it is possible to estimate the remaining processing time and the probability of a particular outcome. Similarly, this information can be used to provide recommendations, e.g., proposing the activity that will minimize the expected costs and time.

This paper presents a general framework for operational decision support. It shows that process mining is not limited to the “Past” but can also be used for the “Present” and “Future”. To make this concrete, we present a new set of approaches for *time-based operational support* implemented in our process mining tool *ProM* [1]. These approaches center around an annotated transition system that contains time information extracted from event logs. The *annotated transition system* can be used to *check* (time) conformance while cases are being executed, *predict* the remaining processing time of incomplete cases, and *recommend* appropriate activities to end users working on these cases.

In the remainder, we first present our framework for operational decision support. Then we describe a concrete application of the framework aiming at time-based operational support, its implementation in ProM, and some initial experiences. Finally, we discuss related work and conclude the paper.

2 Framework for Operational Support

To position the results presented in this paper, we first introduce the classical form of process mining typically done offline. Starting point for process mining is the so-called *event log*. An event log consists of a set of *traces*. Each trace is a sequence of *events* corresponding to a particular *case*. Note that a case represents one process instance, i.e., one run of the process/system. Each event refers to a task and typically also has a timestamp. Moreover, additional data elements, information about resources, event types, etc. may be attached to events. For example the trace $\langle A_{John}^{10}, B_{Mary}^{15}, C_{Mary}^{25}, D_{Pete}^{33} \rangle$ could represent a case for which four tasks are executed *A*, *B*, *C*, and *D*. Each event also has a timestamp and a reference to a resource. For example A_{John}^{10} refers to the execution of *A* by *John* at time 10. An example of a log consisting of three cases would be: $L = \{ \langle A_{John}^{10}, B_{Mary}^{15}, C_{Mary}^{25}, D_{Pete}^{33} \rangle, \langle A_{Ivan}^{12}, C_{Joe}^{16}, C_{Mary}^{24}, B_{John}^{31} \rangle, \langle A_{John}^{14}, E_{Chris}^{18}, D_{Joe}^{44} \rangle \}$.

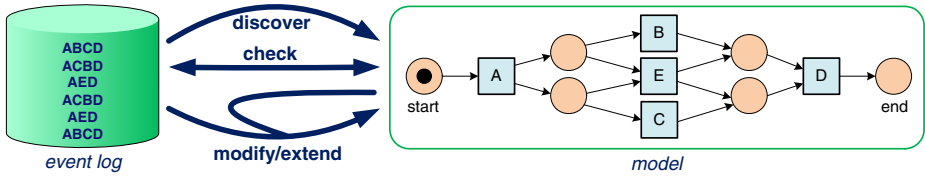


Fig. 1. Overview of classical forms of process mining: *discover*, *check*, *modify*, and *extend*

| focus | traces/log | | | model | | | |
|---|------------|---------|-----------|----------|-------|--------|--------|
| action | check | predict | recommend | discover | check | modify | extend |
| active ("now") online, partial traces | ✓ | ✓ | ✓ | | | | |
| passive ("history") offline, full traces | ✓ | | | ✓ | ✓ | ✓ | ✓ |

... costs time

Fig. 2. Overview of the process mining spectrum distinguishing between the active use of partial traces and passive use of completed traces

As Figure 1 shows there are three types of process mining. First of all, there are various techniques for *process discovery* [2,5]. These techniques aim at extracting models (for example Petri nets, EPCs, UML ADs, BPMN models) from event logs [1]. Secondly, there are techniques for *conformance checking* [16]. These techniques compare the log and the model, measure the “fit”, and highlight deviations in the models. In a model with a fitness of 0.88, 88% of the events in traces can be “replayed” using the model while 12% of the events can not be “replayed” using the model [2]. In the model, it could be highlighted that a particular task happened 343 times without being enabled according to the model. Finally, there are techniques to *modify* or *extend* the model. Based on an analysis of the event log there could be suggestions to change the model, e.g., to make it better fitting. Moreover, an existing model just describing the control-flow could be extended with temporal aspects extracted from the event log. This way bottlenecks are highlighted and the extended model can be used for simulation purposes [17].

The techniques just mentioned have in common that they focus on offline analysis. This means that only *full traces* are being considered, i.e., completed cases that were handled in the past are used. Moreover, process mining is used only in a *passive* manner not directly influencing the running cases. As Figure 2 shows, one can also use process mining in an online setting. Now the focus is on *partial*

¹ Note that the Petri net model shown in Figure 1 was obtained by applying the α -algorithm [5] to the event log shown on the left-hand side of the figure. This is for illustration purposes only; in this paper we present a generic approach and do not favor a particular representation or discovery technique.

² There are various definitions of fitness [16], but this conveys the basic idea.

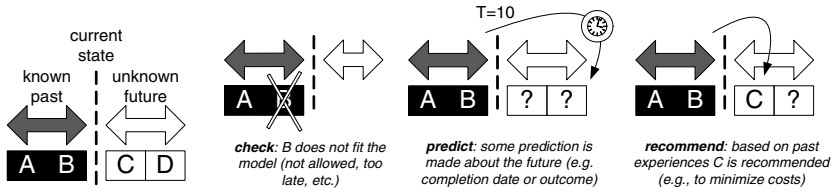


Fig. 3. Overview of operational support

traces, i.e., cases that are still running and did not yet complete. For these cases, the *active* use of process mining is interesting, e.g., to check the last step executed, to predict the future of a case, and to recommend the next task to be executed. The right-hand side of Figure 2 shows the classical forms of process mining (*discover*, *check*, *modify*, and *extend*) already mentioned in Figure 1. These focus on the model rather than the traces or the log, i.e., the result is reported as a (partially) new model or annotations of an existing model. The left-hand side of Figure 2 shows other types of analysis focusing on the traces in the log rather than the model. The third dimension shown in Figure 2 shows the aspect the analysis is focusing on, e.g., time, costs, logic (routing), quality, etc.

The most likely combinations are highlighted using a ✓. Note that most of the passive forms of process mining focus on the model rather than traces. There is one exception (see the ✓ in the bottom-left cell). When doing conformance checking one compares a model and an event log. The deviations can be highlighted in the model as discussed before. However, the deviations can also be shown in the event log, i.e., parts of completed traces that do not fit into the model are highlighted in the log.

In this paper, we focus on the active use of process mining involving partial traces corresponding to cases that did not complete yet. As shown in Figure 2, we identify three types of actions related to such running cases: (a) *check*, (b) *predict*, and (c) *recommend*. We refer to these actions as *operational support* as they aim at influencing the process while it is running.

Figure 3 illustrates the three types of operational support. Starting point is some model and a partial trace. Note that the model is typically learned using classical process mining techniques. The partial trace refers to a case that is running. The left-hand side of Figure 3 shows a partial trace $\langle A, B \rangle$. Note that we abstract from timestamps, resources, data, etc. For this case, we know that A and B occurred, but we do not know its future. Suppose now that the partial trace $\langle A, B \rangle$ is not possible according to the model. In this case, the operational support system would generate an alert. Another possibility would be that B took place three weeks after A while this should happen within one week. In such a case another notification could be sent to the responsible case manager. Such scenarios correspond to the *check* action mentioned before. Figure 3 also illustrates the goal of *predictions*. Given the current state of a case, the model is used to make some kind of prediction. For example, given the $\langle A, B \rangle$ trace it could be predicted that the remaining processing time is ten days. This prediction

Table 1. A fragment of an event log

| case id | task | trans. | resource | timestamp |
|---------|------------------|----------|----------|---------------------|
| 1 | <i>check</i> | complete | admin | 2009-01-01 11:55:25 |
| | <i>advertise</i> | complete | admin | 2009-01-15 14:03:18 |
| | <i>inspect</i> | complete | admin | 2009-01-28 16:56:53 |
| | <i>decide</i> | complete | admin | 2009-02-02 09:08:03 |
| 2 | <i>check</i> | complete | admin | 2009-01-01 09:36:21 |
| | <i>process</i> | complete | admin | 2009-01-15 14:19:59 |
| | <i>decide</i> | complete | admin | 2009-01-20 17:47:13 |
| ... | ... | ... | ... | ... |

would be based on historic information both in the partial trace and in the event log used to learn the model. Predictions are not restricted to time, but can also refer to costs, probability of a particular outcome, resource availability, etc. Closely related to predictions are *recommendations*. The main difference is that recommendations suggest the next action based on possible continuations of the case. Based on the model, one can try all possible actions and see which one would lead to the best (predicted) performance. Note that recommendations are not only used for determining the next task, but also for allocating resources to work-items or for timing a particular action.

The process mining framework *ProM* aims to support the whole spectrum shown in Figure 2. Earlier versions of ProM focused mainly on passive forms of process mining [1]. In the new version of ProM, we aim to also *support operational decision making in a generic manner*. The basic idea is that some operational system, e.g., a workflow management system, business process management system, or other Process-Aware Information System (PAIS), sends partial traces to ProM as shown in Figure 3. ProM then does the appropriate checks, generates predictions, or sends recommendations while using models derived from event logs (or alternatively use manually created models).

3 Application of the Framework to Time-Based Operational Support

To illustrate that process mining is not limited to passive/offline analysis, we will use a small event log of a process for handling requests of citizens for building permits. Using this example, we present new process mining techniques that cover the whole process mining spectrum. Our example process contains five tasks: (1) *check* for checking whether the requested building permit is compliant to the regulations, (2) *advertise* for advertising the requested permit in the local newspaper for a period of 30 days, (3) *inspect* for inspecting the construction site, (4) *process* for handling requests that are not compliant with the regulations, and (5) *decide* for deciding whether to issue or decline the permit. Table 1 shows a fragment of the log. Each line in the event log of this process corresponds to an event related to one of the five mentioned tasks. For each event, information

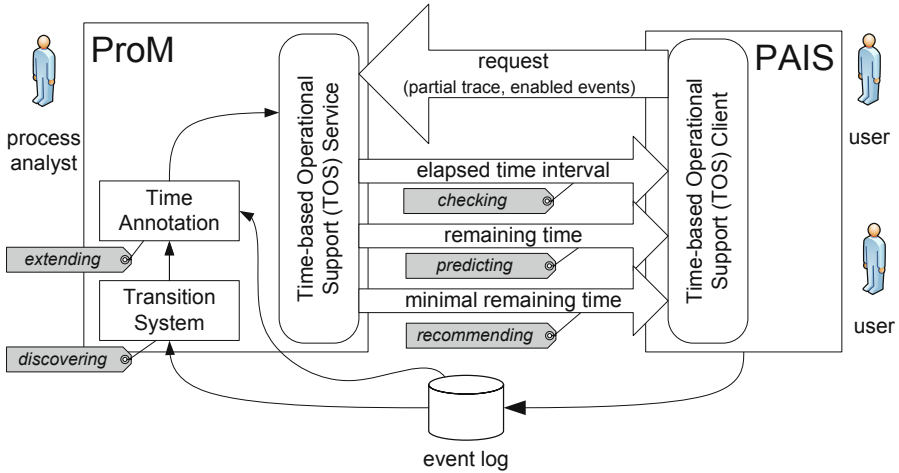


Fig. 4. Architecture of our system to support users based on time information in logs

about the task name, event type, resource that triggered the event and the timestamp is available. Moreover, each event is associated to one case, i.e., a single permit request. For example, Table 1 shows events of two cases containing four and three events, respectively. Note that, for the purpose of simplicity, in the remainder of this paper we will use only the task name to refer to one event.

In the remainder of this section we will use this log to show how process mining techniques can be used for discovering, extending, checking, predicting and recommending in the context of execution times of processes. Figure 4 shows this example and the functionalities of the ProM tool that we will use in this paper. We start with describing existing procedures (i.e., ProM plugins) for *discovering* a transition system from an event log [3] and for *extending* a transition system with time information from an event log (i.e., time annotations) [4] in sections 3.1 and 3.2, respectively. The generated transition system and time annotations can be used to provide useful information about active processes. For this purpose, we have implemented the *Time-based Operational Support* (TOS) Client and Service. The TOS Client can be used by any PAIS to request temporal information about active processes. The TOS Service uses the transition system and its time annotations to generate information about active processes and sends them to the TOS Client. The TOS Client sends the partial trace (i.e., all events executed until the moment of request) and currently enabled tasks of the running case when requesting the information from the TOS Service. The TOS Service generates three types of information about the current case. First, the TOS Service *checks* whether the elapsed time of the current case is within certain temporal boundaries calculated based on elapsed times of earlier completed cases visiting the same state (cf. Section 3.3). Second, Section 3.4 describes how the TOS Service can *predict* the remaining execution time based

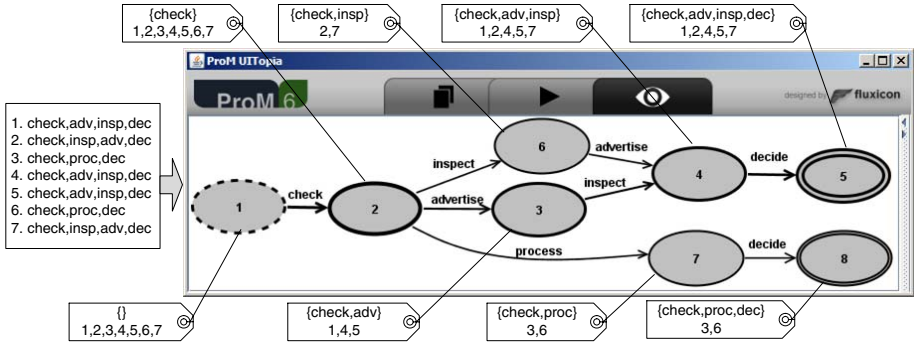


Fig. 5. A transition system constructed from an event log with seven traces

on the past processes. Finally, in Section 3.5 the possibility to *recommend* the enabled events that, based on historic information, are most likely to lead to minimal execution times is described.

3.1 Discovering a Transition System from History

An approach that uses various abstractions for discovering a transition system from an event log is described in 3. The advantage of this process mining technique is that it is very flexible as it allows for a wide range of abstractions, i.e., the discovered model can be tailored towards the needs of the analyst. A transition system is a triplet (S, E, T) where S is the set of states, E is the set of event (transition) labels, and $T \subseteq S \times E \times S$ is the transition relation describing how the system can move from one state to another. For example, Figure 5 shows a ProM screen of a transition system mined from our event log with seven traces containing events referring to tasks *check*, *advertise*, *inspect*, *process* and *decide*. The transition system has eight states ($S = \{s_1, s_2, \dots, s_8\}$), five event labels ($E = \{check, advertise, inspect, decide, process\}$) and eight transitions ($T = \{(s_1, check, s_2), (s_2, advertise, s_3), (s_2, inspect, s_6), (s_2, process, s_7), (s_3, inspect, s_4), (s_6, advertise, s_4), (s_7, decide, s_8), (s_4, decide, s_5)\}$).

The transition system in Figure 5 is mined from the event log using two types of abstractions 3. First, an event abstraction is used when considering which event information is relevant. For example, the transition system shown in Figure 5 is mined using the event abstraction that considers only the task name and ignores the event type, resource and timestamp. Second, a state abstraction is used when it comes to how a sequence of events is ‘replayed’ on the transition system. For example, the transition system shown in Figure 5 is mined using the ‘set state abstraction’ that considers only which tasks were executed and ignores the execution order and frequency. The tags connected to states in Figure 5 show two types of state-related information. First, the set abstraction for the state is shown in the upper line. For example, state s_4 refers to a trace prefix that contains tasks *check*, *advertise* and *inspect* in any order. Second,

the bottom line shows which traces are replayed in which state. For example, traces 1, 2, 4, 5 and 7 all visit state s_4 : traces 1, 4 and 5 after executing sequence $\langle check, advertise, inspect \rangle$ and traces 2 and 7 after executing sequence $\langle check, inspect, advertise \rangle$. It is important to note that this state considers all traces where these three tasks were executed, regardless of the order.

3.2 Extending the Transition System with Time Information

Event logs can also be used to enrich models with information about past executions. An approach for annotating a transition system with time information from an event log is described in [4]. This procedure starts by replaying each trace of the event log on the transition system, and collecting three types of time information extracted from the trace for each visited state. First, the *time elapsed* from the beginning of the trace is assigned to the state as the difference between the timestamp of the current event and the timestamp of the first event in the trace. Second, the *remaining time* until the end of the trace is assigned to the state as the difference between the timestamp of the last event in the trace and the timestamp of the current event. Finally, the *sojourn time*, the time that the trace spent in this state is assigned to the state as the difference between the timestamp of the next event in the trace and the timestamp of the current event.

Figure 6 shows how elapsed, remaining and sojourn times are collected from the building permits event log and transition system in Figure 5. Note that the actual time data extracted from the event log refers to milliseconds, but for the reasons of simplicity displayed time data is rounded to days. Because s_1 is the initial state, elapsed and sojourn times for all traces are zero and remaining times are equal to the total execution times at s_1 . The elapsed (remaining) times in the initial state s_1 correspond to remaining (elapsed) times in the two final states s_5 and s_8 . This is expected, because the remaining time in the initial state must be equal to the elapsed time in the final state for each trace. For example, trace 1 has a total duration time of 68 days. The elapsed, remaining and sojourn times for this trace are shown as the first elements for the states that this trace visits: s_1 , s_2 , s_3 , s_4 and s_5 . While the *remaining time* value decreases from 68 in state s_1 to zero in state s_5 , the elapsed time increases from zero in s_1 to 68 in state s_5 . Note that, in each of these states, the sum of elapsed and remaining time is equal to the trace's total execution time. For example, the sum of elapsed and remaining times for trace 1 in each of the visited states is 68 days. The sum of sojourn times in all states one trace visits is equal to the total duration of that trace. For example, the sum of sojourn times for trace 1 is $0 + 6 + 39 + 23 + 0 = 68$ days.

The collected time information can be used to annotate each state with statistical data for elapsed, remaining and sojourn times: average, standard deviation, etc. In this paper we focus on elapsed and remaining time annotations. We have used time information in our example event log to annotate the transition system mined from this log (cf. Figure 5). Figure 7 shows the ProM screen with elapsed and remaining times in days and hours. For example, the average elapsed time in state s_3 is 9 days and 15 hours and average remaining time in state s_2 is 39 days and 1 hour.

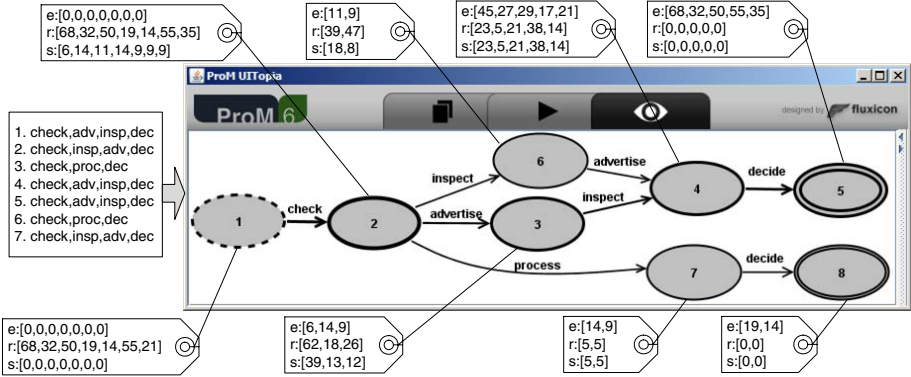


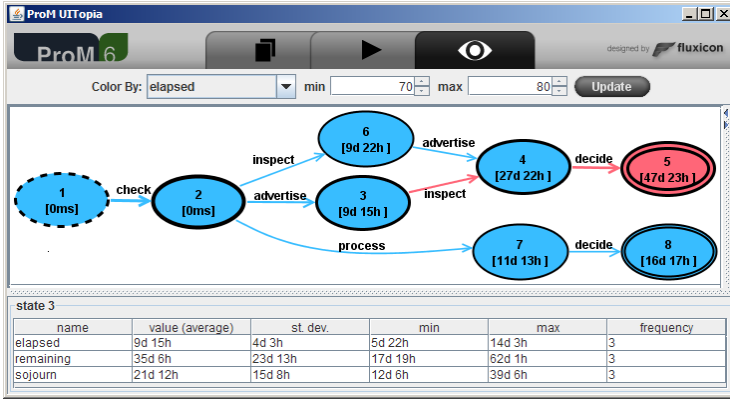
Fig. 6. Collecting elapsed (e), remaining (r) and sojourn (s) times for the transition system from Figure 5

3.3 Checking Running Cases

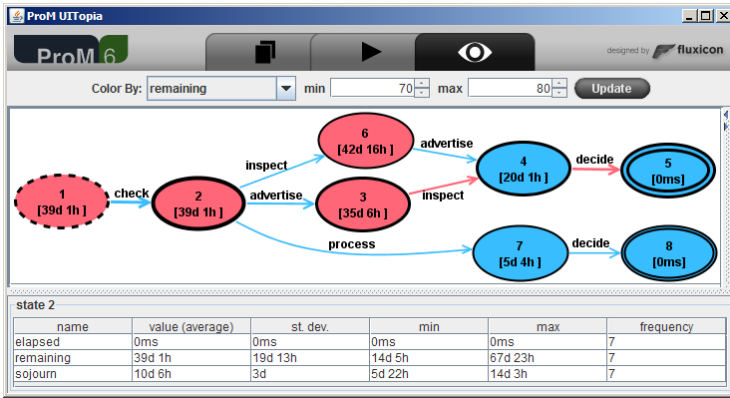
The elapsed time annotations can be used to check how fast currently running cases are being executed when compared to past cases. The procedure for checking a particular running case is as follows:

1. *Replay the partial trace* of the case under consideration on the transition system and identify the current state of the case under consideration.
2. *Calculate the a time interval* for the elapsed time in the current state. The time interval sets upper and lower bounds. There are many ways to define such a time interval. Here, three simple approaches are considered:
 - all elapsed times seen in the past: $[min, max]$,
 - a pre-defined deviation from the average elapsed time: $[\mu - C, \mu + C]$, where μ is the average elapsed time and C is a constant, and
 - standard deviation from the average elapsed time: $[\mu - C * \sigma, \mu + C * \sigma]$, where μ is the average elapsed time, C is a constant and σ is the standard deviation.
3. *Calculate the elapsed time* of the case under consideration as the difference between timestamps of the first and the last event in its partial trace.
4. *Check the execution speed* of the running case: if the elapsed time of the case is under, within or above the time interval, then the case is considered to be slower, as fast as, or faster than processes executed in the past, respectively.
5. *Alert interested parties* (e.g., employees who work on the running case, the manager, etc) if the case is too fast or too slow.

Consider, for example, a situation where inspectors processing permit requests want to be alerted by their TOS Clients if it takes them too long to process a particular request. Assume that the time bounds are set by the time interval $[\mu - 2 * \sigma, \mu + 2 * \sigma]$. Further, assume that an inspector is currently working on



(a) elapsed times



(b) remaining times

Fig. 7. Time annotations based on Figure 5 in ProM

a request (i.e, running case) with the partial trace $\langle check, advertise \rangle$ where tasks *check* and *advertise* were executed on 26/10/2009 and 26/11/2009, respectively. The procedure for checking the elapsed time based on the annotated transition system shown in Figure 6 is as follows:

1. Replaying this partial trace on the transition system leads to state s_3 .
2. The time interval $[\mu - 2 * \sigma, \mu + 2 * \sigma]$ is calculated for the elapsed time in state s_3 . As Figure 7(a) shows, average elapsed time in this state is $\mu = 9$ days and 15 hours and standard deviation is $\sigma = 4$ days and 3 hours. Therefore, the time interval for elapsed times in this state is $[1d9h, 17d21h]$.
3. The elapsed time of the current case is 31 days (time between execution of tasks *check* and *advertise*).
4. The elapsed time of the active process is above the upper bound set by the time interval.

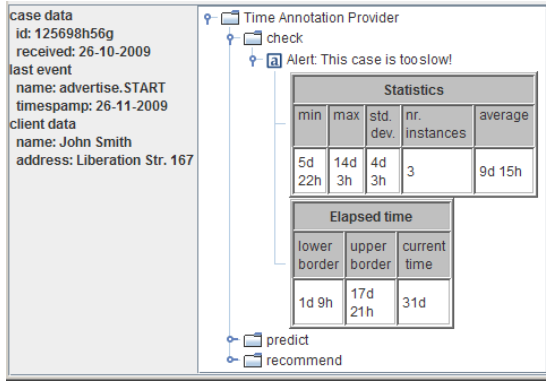


Fig. 8. Checking the elapsed time of a running case with partial trace $\langle check, advertise \rangle$

- Figure 8 shows the alert popping up in the TOS Client. The considered case is substantially slower than cases from the past. In addition to the alert itself, additional information about the running case and the used statistics are included to interpret the warning.

3.4 Predicting the Future of Running Cases

The remaining time annotations created from past cases can also be used to *predict* the remaining execution time of the running cases. The prediction procedure for one running case is simple:

1. *Replay the partial trace* of the case under consideration on the transition system and identify the current state of the case under consideration.
2. *Take the average or median value of the remaining time annotations in the current state* as the prediction for the remaining execution time of the case under consideration.
3. *Notify* interested parties about the predicted remaining time.

Assume, for example, that the inspectors working on building permits want to be informed by their TOS Clients regarding the expected remaining processing time. Consider, for example, a situation where a client who submitted a building permit request (for which the partial trace is $\langle check \rangle$) is interested how much longer it will take to get the final decision. By using the remaining time annotations shown in Figure 7(b), the prediction for this running case can be generated in the following way:

1. Replaying this partial trace on the transition system leads to state s_2 .
2. The predicted remaining execution time based on the average of the remaining times in state s_2 is 39 days and 1 hour.
3. Figure 9 shows how the result presented by the TOS Client.

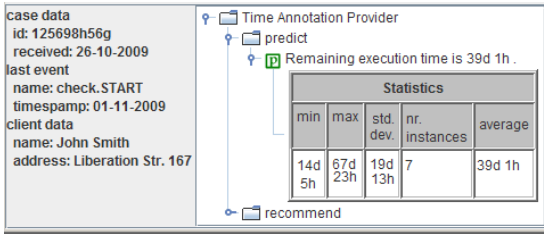


Fig. 9. Predicting the remaining time of a running case with partial trace (*check*)

3.5 Recommending the Next Step for Running Cases

The remaining time annotations of a transition system can also be used to recommend steps that lead to shortest execution times in past cases. In addition to the partial trace of the running case, this procedure also uses the set of enabled events in the case to recommend which one of them should be executed:

1. For each enabled event, identify the state in which the transition system would be if this enabled event would indeed be executed in the running case using the following two steps: (a) *Create a new trace* by extending the partial trace of the running case with the enabled event under consideration; and (b) *Replay the new trace* in the transition system to identify the state to be assigned to this event.
2. *Create an ordered list of recommendations* by sorting enabled events in the increasing order of average remaining times annotated to assigned states: the state assigned to the first recommended event has a shorter (or equal) predicted remaining time than the state assigned to the second recommended event, etc.
3. *Inform* interested parties about the recommendations.

Consider, for example, a situation when inspectors working on a building permit request (i.e., running case) with partial trace (*check*) would like to get the recommendation whether to execute events *advertise.start* or *inspect.start* next (i.e., enabled tasks) in order to process this request as quickly as possible. Based on the remaining time annotations shown in Figure 7(b), the recommendation is generated in the following way:

1. Transition system states are assigned to enabled events *advertise.start* and *inspect.start* by extending the partial trace of the running case: (a) State s_3 is assigned to *advertise.start* because replaying trace $\langle \text{check}, \text{advertise} \rangle$ on the transition system leads to state s_3 ; and (b) State s_6 is assigned to *inspect.start* because replaying trace $\langle \text{check}, \text{inspect} \rangle$ on the transition system leads to state s_6 .
2. The list of recommended events contains *advertise.start* or *inspect.start*, where *advertise.start* has higher priority than *advertise.start* or *inspect.start*, because the state s_3 has a shorter predicted remaining time (i.e., 35 days and 6 hours) than the state s_6 (i.e., 42 days and 16 hours).
3. Figure 10 shows how the recommendations are shown by the TOS Client.

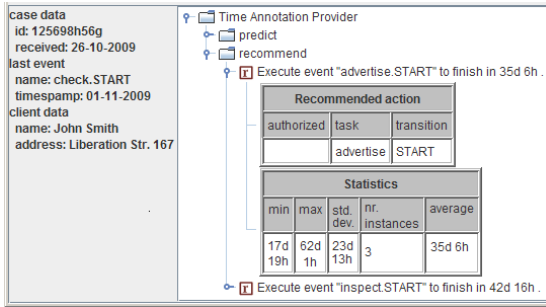


Fig. 10. Recommending the next step for a running case with partial trace $\langle check \rangle$

4 Related Work

Lion's share of process mining research has been focusing on passive forms of process mining such as process discovery [2,5,3,6,8,9,12]. These serve as a basis for learning good models, but are not the focus of this paper. Conformance checking is typically also done in an off-line fashion [16] while the extension of models into full-fledged simulation models is also not used in an operational sense [17]. See www.processmining.org for more pointers to process mining literature.

There have been some initial attempts to support operational decision making using process mining techniques or simulation. In [18] both techniques are combined in the context of YAWL and in [10] non-parametric regression is used to predict completion times. A recommendation service that uses historic information for guiding the user to select the next work item has been implemented in ProM [19] and it is related to case-based reasoning [21]. A recommender for execution of business processes based on the Product Data Model (PDM) is presented in [20].

The framework has been tested using a set of plug-ins related to time-based operational support. This approach is most related to the flexible mining approach in [3] and the prediction approach in [4]. However, in this paper we present an overarching approach, and a generic implementation that does not just support prediction, but also time-based conformance checking and time-based recommendations.

There are various approaches to run time support in the context of world wide web. Some examples are monitoring based on business rules [13], BPEL [7], event calculus [14], etc. Other examples are the various types of recommender systems that support users in their decision-making [15,22]. These systems generate recommendations based on the user's preferences/behavior and are becoming an essential part of e-commerce and information seeking activities. Similar questions are also studied in the context of intrusion detection [11].

The main contribution of this paper is that it provides a framework for positioning the various types of process mining (cf. Figure 2 on page 40) and details the aspect of operational support for running processes in a generic manner. This view is supported in the new version of ProM.

5 Conclusions

In this paper, we focus on the application of process mining to operational decision making. We presented a generic framework and described a set of ProM plug-ins for time-based operational support. The approaches are based on transition systems annotated with time information. These are used to check the timely execution of cases, predict the completion time of cases, and recommend the best steps to minimize the overall flow time. This serves as an example for a much larger set of possible techniques for operational support. In the future, we would like to add more techniques (not only related to time, but also costs, quality, compliance, etc.) and apply them actively in selected domains (most likely hospitals and municipalities). Note that the application of new techniques requires a tight integration with existing information systems.

References

1. van der Aalst, W.M.P., van Dongen, B.F., Günther, C.W., Mans, R.S., de Medeiros, A.K.A., Rozinat, A., Rubin, V., Song, M., Verbeek, H.M.W(E.), Weijters, A.J.M.M.T.: ProM 4.0: Comprehensive Support for Real Process Analysis. In: Kleijn, J., Yakovlev, A. (eds.) ICATPN 2007. LNCS, vol. 4546, pp. 484–494. Springer, Heidelberg (2007)
2. van der Aalst, W.M.P., Reijers, H.A., Weijters, A.J.M.M., van Dongen, B.F., Alves de Medeiros, A.K., Song, M., Verbeek, H.M.W.: Business Process Mining: An Industrial Application. *Information Systems* 32(5), 713–732 (2007)
3. van der Aalst, W.M.P., Rubin, V., van Dongen, B.F., Kindler, E., Günther, C.W.: Process Mining: A Two-Step Approach to Balance Between Underfitting and Overfitting. *Software and Systems Modeling* 9(1), 87–111 (2010)
4. van der Aalst, W.M.P., Schonenberg, M.H., Song, M.: Time Prediction Based on Process Mining. BPM Center Report BPM-09-04, BPMcenter.org (2009)
5. van der Aalst, W.M.P., Weijters, A.J.M.M., Maruster, L.: Workflow Mining: Discovering Process Models from Event Logs. *IEEE Transactions on Knowledge and Data Engineering* 16(9), 1128–1142 (2004)
6. Agrawal, R., Gunopulos, D., Leymann, F.: Mining Process Models from Workflow Logs. In: Schek, H.-J., Saltor, F., Ramos, I., Alonso, G. (eds.) EDBT 1998. LNCS, vol. 1377, pp. 469–483. Springer, Heidelberg (1998)
7. Baresi, L., Ghezzi, C., Guinea, S.: Smart Monitors for Composed Services. In: ICSSOC 2004: Proceedings of the 2nd International Conference on Service Oriented Computing, pp. 193–202. ACM Press, New York (2004)
8. Cook, J.E., Wolf, A.L.: Discovering Models of Software Processes from Event-Based Data. *ACM Transactions on Software Engineering and Methodology* 7(3), 215–249 (1998)
9. Datta, A.: Automating the Discovery of As-Is Business Process Models: Probabilistic and Algorithmic Approaches. *Information Systems Research* 9(3), 275–301 (1998)
10. van Dongen, B.F., Crooy, R.A., van der Aalst, W.M.P.: Cycle Time Prediction: When Will This Case Finally Be Finished? In: Meersman, R., Tari, Z. (eds.) OTM 2008, Part I. LNCS, vol. 5331, pp. 319–336. Springer, Berlin (2008)

11. Feng, L., Guan, X., Guo, S., Gao, Y., Liu, P.: Predicting the Intrusion Intentions by Observing System Call Sequences. *Computers and Security* 23(3), 241–252 (2004)
12. Ferreira, D.R., Gillblad, D.: Discovering Process Models from Unlabelled Event Logs. In: Dayal, U., Eder, J., Koehler, J., Reijers, H.A. (eds.) *BPM 2009*. LNCS, vol. 5701, pp. 143–158. Springer, Heidelberg (2009)
13. Lazovik, A., Aiello, M., Papazoglou, M.: Associating Assertions with Business Processes and Monitoring their Execution. In: *ICSOC 2004*, pp. 94–104. ACM Press, New York (2004)
14. Mahbub, K., Spanoudakis, G.: A Framework for Requirements Monitoring of Service Based Systems. In: *ICSOC 2004*, pp. 84–93. ACM Press, New York (2004)
15. Resnick, P., Varian, H.R.: Recommender Systems. *Communications of the ACM* 40(3), 56–58 (1997)
16. Rozinat, A., van der Aalst, W.M.P.: Conformance Checking of Processes Based on Monitoring Real Behavior. *Information Systems* 33(1), 64–95 (2008)
17. Rozinat, A., Mans, R.S., Song, M., van der Aalst, W.M.P.: Discovering Simulation Models. *Information Systems* 34(3), 305–327 (2009)
18. Rozinat, A., Wynn, M., van der Aalst, W.M.P., ter Hofstede, A.H.M., Fidge, C.: Workflow Simulation for Operational Decision Support. *Data and Knowledge Engineering* 68(9), 834–850 (2009)
19. Schonenberg, H., Weber, B., van Dongen, B.F., van der Aalst, W.M.P.: Supporting Flexible Processes Through Recommendations Based on History. In: Dumas, M., Reichert, M., Shan, M.-C. (eds.) *BPM 2008*. LNCS, vol. 5240, pp. 51–66. Springer, Heidelberg (2008)
20. Vanderfeesten, I.T.P., Reijers, H.A., van der Aalst, W.M.P.: Product Based Workflow Support: Dynamic Workflow Execution. In: Bellahsene, Z., Léonard, M. (eds.) *CAiSE 2008*. LNCS, vol. 5074, pp. 571–574. Springer, Heidelberg (2008)
21. Weber, B., Wild, W., Breu, R.: CBRFlow: Enabling Adaptive Workflow Management Through Conversational Case-Based Reasoning. In: Funk, P., González Calero, P.A. (eds.) *ECCBR 2004*. LNCS (LNAI), vol. 3155, pp. 434–448. Springer, Heidelberg (2004)
22. Zhou, B., Hui, S.C., Chang, K.: An Intelligent Recommender System Using Sequential Web Access Patterns. In: *IEEE Conference on Cybernetics and Intelligent Systems*, pp. 393–398 (2004)

Dependency Discovery in Data Quality

Daniele Barone¹, Fabio Stella², and Carlo Batini²

¹ Department of Computer Science, University of Toronto, Toronto, ON, Canada
barone@cs.toronto.edu

² Department of Informatics, Systems and Communication,
University of Milano-Bicocca, Milano, Italy
{stella,batini}@disco.unimib.it

Abstract. A conceptual framework for the automatic discovery of dependencies between data quality dimensions is described. Dependency discovery consists in recovering the dependency structure for a set of data quality dimensions measured on attributes of a database. This task is accomplished through the data mining methodology, by learning a Bayesian Network from a database. The Bayesian Network is used to analyze dependency between data quality dimensions associated with different attributes. The proposed framework is instantiated on a real world database. The task of dependency discovery is presented in the case when the following data quality dimensions are considered; accuracy, completeness, and consistency. The Bayesian Network model shows how data quality can be improved while satisfying budget constraints.

Keywords: Data quality, Bayesian networks, Data mining.

1 Introduction

In the last two decades, research and practical efforts to improve data quality did not recognize, with the exception of a few works (e.g., *logical interdependence analysis* [1], *tradeoff analysis* [2,3,4,5,6] and *data dependency analysis* [7]), the relevance of studying and analyzing potential dependencies among data quality dimensions, namely, correlations and reciprocal influences among them. To give an example, data that are up-to-date (thus having a low currency) have a high chance to be incorrect too.

Nowadays, the issue of data quality and of dependencies among quality dimensions is gaining more and more importance in the life of organizations. For example, concerning those Information Systems used for decision-making and problem solving in business, i.e., *Decision Support Systems* (DSSs) and *Management Information Systems* (MISs), it is well known [8] that their effectiveness is strictly related to the quality of information resources involved in the decision making process. Making correct decisions is clearly dependent on the quality of data used [9]; however, complete knowledge regarding the quality of data cannot be gained without knowing what the existing relationships are among data quality dimensions. In fact, as shown in the following examples, dimensions

can be strictly related to each other and dependencies among them can play an important role in a decision making process:

- *Accuracy and Timeliness*: often the information becomes better over time, i.e., more *accurate*. However, it is also possible that for a given context, the information becomes less relevant and critical over time [2]. Consider an air traffic control center which receives data from several controller stations. To regulate air traffic, the traffic control center has to cope with uncertain data. Thus, the decision process must balance the delay in receiving more accurate data of airplane positions and the critical period of time in which an “effective” decision must be made to regulate traffic;
- *Consistency vs Completeness*: often the information is based on incomplete but consistent data or on complete but less consistent data [4]. A classic situation concerns human resource data in which different evaluators (i.e., data sources), provide information regarding a particular employee. Those evaluators could have used different attributes to characterize an employee and could have evaluated him over a potentially extended time period. In a scenario where the goal is to promote one out of several employees to a position in senior management, the decision process can use different strategies; it can use all available data even though some items are inconsistent, or only use recent data obtained by a common evaluator.

Therefore, taking into account data quality dependencies is a basic ingredient for rationale decision making and activity planning. Moreover, the knowledge of data quality dependencies can also be extremely important for improvement activities; in fact, it contributes to: i) *diagnose* which is the most probable cause of the bad quality for the considered data quality dimensions and thus helps to identify error sources in an Information System; ii) *select* the most effective data quality improvement strategy, i.e., the one which maximizes data quality when we are subject to budget constraints. Finally, since the quality of data quickly degenerates over time¹, a complete “knowledge” of quality issues represents a fundamental set of *quality requirements* in those (Evolution) Information Systems [11] in which the capability to actively react to organization changes must also take into account data quality problems².

This paper presents the Dependency Discovery in Data Quality (D³Q) framework which extends, from bivariate to multivariate, the data-driven analysis in [7]. As shown in Fig. 1, the D³Q framework is a flexible component that can be added as an extra layer to whatever data quality assessment solution is available. It provides a “comprehensive” data quality knowledge for supporting improvement activities. The results provided by the assessment activities (*Assessment Knowledge* (AK)), are exported into the *Dependency Discovery Model* (DDM)

¹ Experts say that 2% of the records in a customer file become obsolete in a month because customers die, divorce, marry and move [10].

² Data entry errors, systems migrations, and changes to source systems, generate bucket loads of errors. Thus, the knowledge of the cause-effect structure helps to diagnose what the error sources are.

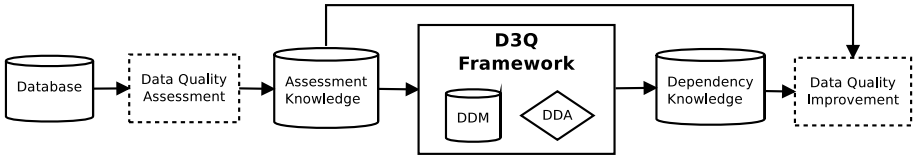


Fig. 1. The D³Q framework

which feeds the *Dependency Discovery Algorithm* (DDA) implemented by a data mining algorithm to provide the *Dependency Knowledge* (DK). The assessment activity plays a critical role for an “effective” discovery of dependencies, since the benefit of the proposed approach is directly influenced by the results obtained during the assessment phase. This paper focuses on the data quality dependency discovery, while assessment that received much attention from the data quality community is not discussed. The interested reader can refer to [12] for a rich and comprehensive survey on data quality assessment methodologies and techniques. Bayesian Networks have been used to implement the DDA component. The D³Q framework has been tested on a real world database, namely the Italian Social Security Contributors’ Anagraph database, which is assessed along the three most important data quality dimensions, i.e., syntactic accuracy, completeness, consistency, and the most adopted metrics³. However, it is worthwhile to mention that it can be applied to any given number and type of data quality dimensions.

The rest of the paper is organized as follows. Section 2 describes the D³Q framework together with its main components, i.e., the DDM and the DDA. Section 3 reports on the instantiation of D³Q framework to the selected database, namely the Italian Contributors’ Anagraph database. This section describes the main features of the Italian contributors’ anagraph database, the related assessment activities and the results of a rich set of numerical experiments concerning the task of D³Q. The last section is devoted to conclusions and directions for further research.

2 The D³Q Framework

The proposed framework aims to discover the dependency structure of the assessed data quality dimensions for a set of attributes belonging to a target database. In this paper the authors focus their attention on the case where the data quality dimensions are described by means of binary variables, i.e., (true,

³ In [12], about 70% of the reviewed methodologies use syntactic accuracy, and about 60% adopt the NULL value as a completeness measure. Moreover, these dimensions and the associated metrics are also part of the ISO/IEC 25012:2008 - SQuaRE - Data Quality Model [13], in which they appear in the first three positions of the list of dimensions belonging to the model.

false). However, it is possible to extend the proposed approach to include the case where the data quality dimensions are represented through discrete multi-value and/or continuous variables.

2.1 The Dependency Discovery Model

The Dependency Discovery Model receives the results of the data quality assessment component in input in order to build the corresponding learning dataset LD^{DDM} , i.e., the dataset used to discover the dependency structure of the assessed data quality dimensions. To clarify how the DDM is used to solve the D³Q problem and how the LD^{DDM} is obtained, we need to introduce several quantities. Let $\mathbf{X} = \{x_i, i = 1, \dots, N\}$ be the set of instances of a database consisting of M attributes $A = \{A_1, \dots, A_M\}$ on which K data quality dimensions $D = \{d_1, \dots, d_K\}$ have been assessed. Furthermore, let the data quality dimension d_j for the attribute A_l be described by means of a binary variable $Y_{(l,j)}$. Then, for the instance x_i we have that $Y_{(l,j)}^{(i)} = true$ in the case where the value of the attribute A_l for the instance x_i , $A_l(x_i)$, is correct w.r.t the data quality dimension d_j . The $DDM(\mathbf{X}) = \{Y_{(l,j)}^i(\mathbf{X}) \mid i = 1, \dots, N, l = 1, \dots, M, j = 1, \dots, K\}$ model consists of N realizations of $M \cdot K$ binary variables $Y_{(l,j)}$. The learning dataset $LD^{DDM}(\mathbf{X})$ (hereafter $LD(\mathbf{X})$) is obtained exploiting the results provided by the assessment component.

2.2 The Dependency Discovery Algorithm

The Dependency Discovery Algorithm is implemented through a Bayesian Network (BN) which exploits the information stored using the DDM, i.e., the learning dataset LD^{DDM} , to discover the dependency structure. Bayesian Networks (BNs) implement an optimal trade-off between complexity and interpretability when we have to cope with highly dimensional domains. BNs are an alternative to rule based models. BNs differ from rule based models in three ways: i) instead of modeling the knowledge of the domain expert, they model the domain; ii) instead of using a non-coherent uncertainty calculus tailored for rules, they use classical probability calculus and decision theory; iii) instead of replacing the expert, they support her/him. The main problem of rule based models is coherence. Indeed, rule based models do not deal properly with uncertainty which naturally arises for the following reasons; observations may be uncertain, the information may be incomplete, the relations in the domain may be of a non-deterministic type. A way to incorporate uncertainty in rule based models is to extend the production rule to include the concept of certainty for both the left and the right part of the rule. The rule based model must be extended with new inference rules, which shall ensure a coherent reasoning under uncertainty. However, it is not possible to capture reasoning under uncertainty with inference rules. Inference rules are context free; while coherent reasoning under uncertainty is sensitive to the context in which the certainties have been established [14].

A BN \mathbf{B} consists of n discrete random variables X_1, \dots, X_n and an underlying Directed Acyclic Graph (DAG) $G = (V, E)$, where V is the set of vertices while

E is the set of directed links. Each random variable is uniquely associated with a vertex of the DAG. The BN model \mathbf{B} is fully specified by means of the DAG G , together with a set of conditional probability tables $P(X_i|pa[X_i])$, $i = 1, \dots, n$, where $pa[X_i]$ denotes the parents of node X_i , i.e., the set of variables which directly influence the random variable X_i . The main characteristic of the BN model \mathbf{B} is that the joint probability distribution for the random vector (X_1, \dots, X_n) can be represented through the following factorization:

$$P(X_1, \dots, X_n) = \prod_{i=1}^n P(X_i|pa[X_i]). \quad (1)$$

In the case where the random variable X_i has no parents (no directed links oriented towards the node associated with X_i), $P(X_i|pa[X_i])$ is simply its marginal probability $P(X_i)$. One of the interesting features of BNs is that one can infer conditional variable dependencies by visually inspecting the DAG and exploiting the concept of conditional independence [15]. Another interesting aspect is that many algorithms are available for learning their structure from data [16]. The main structural learning algorithms, namely PC and NPC [17], are based on making dependence tests that calculate a test statistic which is asymptotically chi-squared distributed when assuming (conditional) independence. If the test statistic is large for a given independence hypothesis, the hypothesis is rejected; otherwise, it is accepted. The probability of rejecting a true independence hypothesis is given by the *level of significance*. Heckerman [17] provides a full discussion of how the Bayesian estimation approach can be exploited to construct BNs from data. It is worthwhile to mention that inference and learning algorithms are available for multi-valued and continuous random variables. However, for multi-valued random variables the complexity grows exponentially with the number of parents for each node. While for continuous random variables, inference and learning are restricted to gaussian and conditionally gaussian distributions.

3 Dependency Structure Discovery

This section is devoted to present the Italian Social Security Contributors' List database together with the corresponding BN model, learnt from the available learning dataset. The BN model is queried to show how it can be used for inference on data quality dimensions dependency. A simple example illustrating how the BN model could be exploited for data quality improvement is given.

3.1 Italian Social Security Contributors' List

The *Italian Social Security Contributors' List* (ISSCL) database, maintained by the Italian social security administration, contains data related to Italian contributors on social security for retirement funds. The ISSCL database consists of the following six attributes; *Social Security Number/Value Added Tax Number, Juridical/Physical Person Name, Street, ZipCode, Country, and Region*. A

portion of the ISSCL database, containing 1,483,712 records, is used; while for privacy reasons, only the following three attributes ($M = 3$): *ZipCode*, *Country*, and *Region*, have been included in the dataset \mathbf{X} . Three data quality dimensions ($K = 3$) have been considered: namely *accuracy*, *completeness* and *consistency*. The (*syntactic*) *accuracy* has been assessed by using a *comparison function*, which computes the distance between the value v of a given attribute and the true values belonging to a domain look-up table. A relevant example of comparison function is offered by the Edit distance [18]. Through the DDM we addressed only exact matching by means of binary variables. However, it is worthwhile to mention that it is possible to use a threshold value to deal with approximate matching. The assessment of the *completeness* dimension is based on i) the close world assumption [19], which states that only the values actually present in a relation, and no other values, represent true facts of the real world, and ii) the relational data model with *NULL* values. The *consistency*, which is defined as the logical coherence of different information [20], has been assessed by finding the violations of semantic rules. A semantic rule is defined over data items (a set of), where the items can be either tuples (e.g., x_i) of relational tables or records belonging to a given file. In regard to \mathbf{X} , the following business rule is applied:

$$\mathbf{IF} \text{ ZipCode}(x_i) = z \mathbf{THEN} (\text{Country}(x_i) = u \text{ AND } \text{Region}(x_i) = v), \quad (2)$$

which means that if, for a given tuple x_i , the attribute *Zip* equals z , then the values of *Country* (u) and *Region* (v) are univocally determined by means of a domain look-up table.

3.2 Bayesian Network Learning and Inference

The meanings of the binary random variables: *CO-acc*, *CO-com*, *RE-acc*, *RE-com*, *ZIP-acc*, *ZIP-com* and *Cons*, belonging to the learning dataset $LD(\mathbf{X})$, are described in Table 1. It is worthwhile to mention that the variable *Cons* refers to the triplet of attributes (*Country*, *Region*, *ZipCode*) and measures the consistency of the entire tuple by using the business rule (2). The BN modeling tasks, structural learning and inference, have been accomplished by using HUGIN Ver. 6.3. The structural learning tasks, performed by using PC and NPC with the value of the *Level of Significance* parameter set to 0.01, resulted in the same BN model (Fig. 2). The BN model represents the dependency structure of

Table 1. Variables for the ISSCL dataset

| Name | Meaning |
|----------------|-----------------------|
| <i>CO-acc</i> | Country accuracy |
| <i>CO-com</i> | Country completeness |
| <i>RE-acc</i> | Region accuracy |
| <i>RE-com</i> | Region completeness |
| <i>ZIP-acc</i> | ZIP code accuracy |
| <i>ZIP-com</i> | ZIP code completeness |
| <i>Cons</i> | tuple consistency |

the assessed data quality dimensions for the considered subset of attributes in a compact and efficient way. The model summarizes **qualitative** and **quantitative** knowledge extracted from the considered database. These two kinds of knowledge are the basic elements of the DK component in Fig. II. The qualitative knowledge is associated with the graph component of the BN model, i.e., the DAG; while the quantitative knowledge is obtained through the inferential process over the nodes of the BN model.

Qualitative Knowledge (QLK) is read from the DAG by exploiting the property of the *Markov blanket* of a node, i.e., the set of its parents, children and nodes with which it shares children. Indeed, the *Markov blanket property* states that a node is independent from the rest of the BN's nodes when the state of its *Markov blanket* is known.

Example 1. An example of qualitative knowledge extracted from the analyzed database is as follows: the variable *Cons* is conditionally independent from the variables *ZIP-com*, *CO-com* and *RE-com*, given the variables *ZIP-acc*, *CO-acc* and *RE-acc*. This means that whenever the states of the variables *ZIP-acc*, *CO-acc*, and *RE-acc* are known, the knowledge about the state of one or more of the following variables *ZIP-com*, *CO-com* and *RE-com* brings no information about the variable *Cons*. Conditional independence is symmetric. Thus, the variables *ZIP-com*, *CO-com* and *RE-com* are conditionally independent from the variable *Cons*, given the variables *ZIP-acc*, *CO-acc* and *RE-acc*.

Comments. The above example makes explicit what is known from theory, i.e., the completeness property is a necessary condition for the accuracy property. In fact, we can evaluate the syntactic accuracy of a value only in the case where we

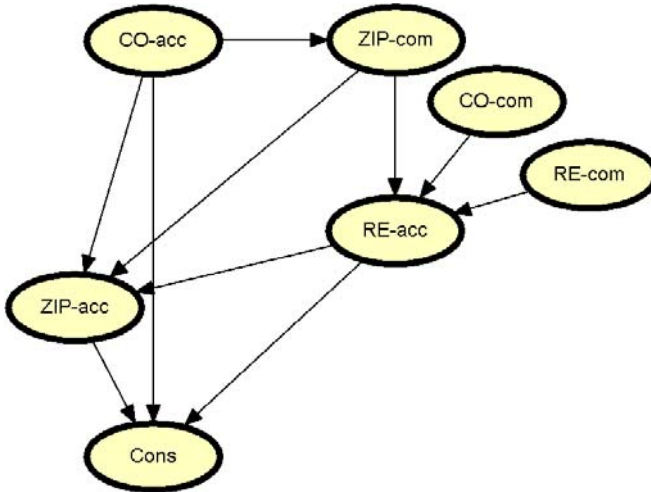


Fig. 2. BN model of the ISSCL dataset

have some information to assess. Therefore, it is also true that a value which is not complete is also not accurate. Notice that, a missing value does not always represent an error of completeness. For instance, in a table of employees that has a column for the name of the employees's manager, that value will be missing for the president of the company. In this case, no value is applicable for the president and therefore, no information can be assessed along the accuracy dimension. However, in this scenario, the NULL value can be seen as complete information (in referring to the semantic expressed by the employees table) but also accurate, though we have no information to assess.

Example 2. A second example of qualitative knowledge, which refines what presented through Example 1, is as follows. *Cons* is conditionally independent from *CO-com* and *RE-com* given the variables *ZIP-com* and *RE-acc*, while it is not conditionally independent from *CO-acc* and *ZIP-acc*.

Comments. The above example enforces the statement we provided about the completeness property as a necessary condition for the accuracy property. In fact, *Cons* becomes conditionally independent from *ZIP-com*, *CO-com* and *RE-com*, if we already have some accuracy values for *ZIP-acc*, *CO-acc* and *RE-acc*, i.e., we had some values (complete information) on which to perform activity assessment along the accuracy dimension.

Quantitative Knowledge (QTK) is accessed through the inferential process applied to the BN model. The inferential process allows to quantify multivariate relationships between data quality dimensions, i.e., to quantify how data quality dimensions relate to each other and impact on each other. The inferential process includes the computation of the most probable configuration of the BN model, the computation of the posterior distribution for a set of nodes when some evidence is available, as well as the computation of the most probable cause for the available evidence. This last type of inferential instance is known as BN diagnosis and is particularly useful for selecting the correct inspection policy when dealing with decision making under uncertainty. By inspection policy we mean the process through which we acquire information on the BN variables to discover what the cause of the available evidence is.

Example 3. The first information extracted from the BN model depicted in Fig. 2 concerns the most probable joint assignment of the states for the variables, i.e., the most probable configuration of a database record. This joint assignment is (*CO-acc=true*, *CO-com=true*, *RE-acc=true*, *RE-com=true*, *ZIP-acc=true*, *ZIP-com=true*, *Cons=true*), and it occurs with probability 0.743995. Thus, the probability that a tuple belonging to the considered database is affected by at least one data quality problem equals 0.256005.

Comments. The above example shows how the QTK easily answers simple but important questions, such as: “What is the probability that a tuple belonging to the database is wrong?”. From a business point of view, if we consider the following example, the relevance of the answer becomes clear. In fact, referring

Table 2. Posterior probability $P(RE - acc, RE - com|CO - acc, CO - com)$

| (CO-acc, CO-com) | (RE-acc, RE-com) | $P(RE - acc, RE - com CO - acc, CO - com)$ |
|------------------|------------------|--|
| (true,true) | (true,true) | 0.964357 |
| | (true,false) | 0.000000 |
| | (false,true) | 0.035640 |
| | (false,false) | 0.000003 |
| (false,true) | (true,true) | 0.963785 |
| | (true,false) | 0.000000 |
| | (false,true) | 0.036213 |
| | (false,false) | 0.000002 |
| (false,false) | (true,true) | 0.399998 |
| | (true,false) | 0.000000 |
| | (false,true) | 0.600000 |
| | (false,false) | 0.000002 |

Table 3. Posterior probability $P(CO - acc, CO - com|RE - acc, RE - com)$

| (RE-acc, RE-com) | (CO-acc, CO-com) | $P(CO - acc, CO - com RE - acc, RE - com)$ |
|------------------|------------------|--|
| (true,true) | (true,true) | 0.962532 |
| | (true,false) | 0.000000 |
| | (false,true) | 0.037464 |
| | (false,false) | 0.000004 |
| (false,true) | (true,true) | 0.961879 |
| | (true,false) | 0.000000 |
| | (false,true) | 0.038064 |
| | (false,false) | 0.000057 |
| (false,false) | (true,true) | 0.962522 |
| | (true,false) | 0.000000 |
| | (false,true) | 0.037476 |
| | (false,false) | 0.000002 |

to a generic selling process which uses customer information to perform transactions, it is possible to estimate the probability of process failure exploiting the probability of a (customer) tuple being wrong.

Example 4. Table 2 and Table 3 report two examples of the computation of the posterior distribution for a set of nodes when some evidence is available. Indeed, Table 2 (Table 3) reports the posterior probability distribution of accuracy and completeness for the attribute Region (Country) depending on the accuracy and completeness for the attribute Country (Region). Therefore, from Table 2 we discover that the probability for Region to be jointly accurate and complete ($RE-acc=true, RE-com=true$) decreases from 0.964357, when Country is jointly accurate and complete ($CO-acc=true, CO-com=true$), to 0.399998, when Country is neither accurate nor complete ($CO-acc=false, CO-com=false$). Furthermore, from Table 3 we discover that the probability for Country to be complete but not

accurate (CO-acc=false, CO-com=true) increases from 0.037464, when Region is jointly accurate and complete (RE-acc=true, RE-com=true), to 0.038064, when Region is complete but not accurate (RE-acc=false, RE-com=true).

Comments. The QTK allows the following considerations: i) if *Country* is complete and accurate then *Region* tends to be accurate; ii) if *Country* is not complete and therefore not accurate, the quality of *Region* significantly decreases; and iii) the probability for *Country* being complete but not accurate is slightly affected by the accuracy of *Country*, given that *Country* is complete. In fact, a possible case for i) is to suppose that a sales employee is registering information about a new customer who lives in “Castelfiorentino” in the province of “Florence” and that he/she is able to correctly type “Castelfiorentino” since he/she knows this place; then there is a high probability that he/she correctly inputs the associated *Region* “Florence”. Analogous cases can be found for ii) and iii).

Example 5. The last instance of the inferential process, i.e., BN diagnosis, is probably the most interesting one. Suppose we are concerned with the most probable cause of the inaccuracy for the attribute *Region*, i.e., we want to discover which data quality dimension/s for which attribute/s is associated with the inaccuracy of attribute *Region*. While the prior marginal probability for each variable, is reported in Table 4, the BN model is queried with the following evidence $RE-acc=false$ to compute the posterior probability for the remaining variables $CO-acc$, $CO-com$, $RE-com$, $ZIP-acc$, $ZIP-com$ and $Cons$ (Table 5). The posterior probability of the variable $Cons$ equals 1. This is obvious when recalling the definition of tuple consistency implemented through the business rule (2). The most probable single cause for the inaccuracy of the attribute *Region* is the inaccuracy for the attribute *Country* ($P(CO-acc=false|RE-acc=false) = 0.038065$), the second most probable single cause is the inaccuracy for the attribute *ZipCode* ($P(ZIP-acc=false|RE-acc=false) = 0.028000$) while the third most probable single cause is the incompleteness for the attribute *ZipCode* ($P(ZIP-com=false|RE-acc=false) = 0.000624$) and so on. However, according to posterior probability values listed in Table 5, $CO-acc=false$ is about one and a half times more probable than $ZIP-acc=false$, which in turn is about fifty times more probable than $ZIP-com=false$. Therefore, it is very likely that by inspecting the attribute *Country*, we find it to be accurate, i.e., $CO-acc=true$. Then, we can think that the next two data quality dimensions to inspect are the accuracy and the completeness for the attribute *ZipCode*. However, this is not the correct database inspection policy. Indeed, we must introduce the new evidence into the BN model to compute the new posterior probability for the non evidenced variables $Cons$, $ZIP-acc$, $ZIP-com$, $RE-com$ and $CO-com$ (Table 6). Under the new evidence ($RE-acc=false, CO-acc=true$), we deduce that the attribute *ZipCode* is complete, i.e., $P(ZIP-com=false|RE-acc=false, CO-acc=true) = 0$.

Comments. The BN diagnosis process suggests what the right inspection policy is to discover the cause of the observed evidence, lack of data quality. From a data quality analyst point of view, this can be a useful instrument for discovering cause-effect patterns in order to identify the most relevant sources of errors.

Table 4. Prior probability

| VARIABLE | $P(\text{Variable} = \text{false})$ |
|----------|-------------------------------------|
| CO-acc | 0.037487 |
| CO-com | 0.000003 |
| RE-acc | 0.035663 |
| RE-com | 0.000004 |
| ZIP-acc | 0.168496 |
| ZIP-com | 0.000022 |
| Cons | 0.256005 |

Table 5. Posterior probability for the evidence [$RE - acc = false$]

| VARIABLE | $P(\text{Variable} = \text{false} RE - acc = \text{false})$ |
|----------|---|
| Cons | 1.000000 |
| CO-acc | 0.038065 |
| ZIP-acc | 0.028000 |
| ZIP-com | 0.000624 |
| RE-com | 0.000113 |
| CO-com | 0.000057 |

Table 6. Posterior probability for the evidence [$RE - acc = false, CO - acc = true$]

| VARIABLE | $P(\text{Variable} = \text{false} RE - acc = \text{false}, CO - acc = \text{true})$ |
|----------|---|
| Cons | 1.000000 |
| ZIP-acc | 0.013515 |
| RE-com | 0.000113 |
| CO-com | 0.000057 |
| ZIP-com | 0.000000 |

3.3 Data Quality Improvement

The DK component

- assists the data quality expert in discovering which the most probable *sources of non quality* are. An example is described in [7], in which a positive association between the *timeliness* of the Standard & Poor’s (S&P’s) and Moody’s data sources has been discovered. This association was caused by an inappropriate loading process for the considered data sources; the loading process, responsible for feeding the internal database by using the S&P’s and Moody’s data sources, was not updated with the same frequency as that of the external bond rating data provider;
- allows the selection of the most effective improvement activity. If the syntactic accuracy of Y depends on the syntactic accuracy of X , then it could be the case that improving the quality of Y will result in a quality improvement for X , while improving the quality of X does not necessarily bring an improvement of the quality for Y ;
- implements the correct process for decision making; it avoids redundant activities to be performed and thus minimizes the improvement costs.

To clarify how the DK can be used to improve data quality lets us present the following simple example. Assume we want to maximize the probability of *Consistency* for the ISSCL database. Suppose, for any attribute belonging to the database, we can force the accuracy to hold true. However, to force accuracy to be true an amount of money has to be paid, where the amount of money depends on the attribute to act on. Furthermore, it is usual to impose an upper bound on the budget to maximize the probability of *Consistency*. The amount of money required to force accuracy on the three attributes of the ISSCL database is reported in Table 7, while the budget for each tuple of the database is set to 2.20€. If the accuracies for the considered attributes are assumed to be independent, then the probability of *Consistency* could be maximized by means of a simple ranking procedure. Indeed, we can compute the conditional probability of *Cons* to be true when the accuracy for the considered attribute is set to true. These conditional probability values are ranked in Table 8. Information in Table 7 and in Table 8 allow to conclude that the solution which maximizes the probability of *Consistency* consists of forcing *ZIP-acc=true* and *CO-acc=true*. This solution costs 1.50€+ 0.60€= 2.10€ < 2.20€, which satisfies the budget constraint, while enforcing also *RE-acc=true* further maximizes the probability of *Consistency* but violates the budget constraint (1.50€+ 0.60€+ 0.40€= 2.50€ > 2.20€). However, this solution (*ZIP-acc=true, CO-acc=true*) results in a probability of *Consistency* equal to 0.926090 ($P(Cons = true | ZIP - acc = true, CO - acc = true) = 0.926090$), as computed through inference on the BN model. Thus, it is not the optimal solution which is obtained from the BN model to be (*ZIP-acc=true, RE-acc=true*). This solution is both feasible (1.50€+ 0.40€= 1.90€ < 2.20€) and optimal ($P(Cons = true | ZIP - acc = true, RE - acc = true) = 0.933681$).

The above example emphasizes the importance of knowing the structure of dependency between data quality dimensions when concerned with decision making under uncertainty. The knowledge of the dependency structure between the *ZIP-acc*, *ZIP-com*, *RE-acc*, *RE-com*, *CO-acc*, *CO-com* and *Cons* data quality dimensions allows to efficiently and effectively improve the data quality of the considered database.

Table 7. Costs to enforce accuracy for the ISSCL database

| ATTRIBUTE | COST |
|-----------|-------|
| ZipCode | 1.50€ |
| Country | 0.60€ |
| Region | 0.40€ |

Table 8. Sorted conditional probabilities $P(Cons = true | Variable = true)$

| VARIABLE | $P(Cons = true Variable = true)$ |
|----------|------------------------------------|
| ZIP-acc | 0.894757 |
| CO-acc | 0.772971 |
| RE-acc | 0.771509 |

4 Related Work

The analysis of dependencies between data quality dimensions has been mainly investigated in terms of tradeoff. An example of such an approach is presented in [2], where the authors investigate how the improvement of the timeliness dimension negatively affects the accuracy dimension. The paper presents a theoretical framework to select the most appropriate changes to data managed in information systems. An accuracy-timeliness utility function figures prominently in the analysis, while in the case when it is not possible to determine the utility function, the authors describe how to identify a suitable approximation. A similar framework, that allows the systematic exploration of the tradeoff between completeness and consistency is presented in [4]. The relative weight (importance) of completeness and consistency to the decision maker is an input to the analysis. In order to examine the tradeoff the authors explore various facets of the two dimensions that produce analytical expressions for the measurement activity. The utility of various combinations of completeness and consistency, for fixed and variable budgets, provides guidance to evaluate the appropriate tradeoff of these factors for specific decision contexts. The main difference between the frameworks presented in [24] and our approach, is that D^3Q does not limit to a tradeoff between two dimensions but addresses multi-variate dependencies. Moreover, in [24], the user must provide: i) a *weight*, that represents the importance of a dimension versus another dimension, and ii) a *functional relationship*, that defines the binding between the dimensions involved⁴. Instead, the D^3Q is able to learn such information (probabilities) directly from the data. In [5], a practical case of tradeoff among timeliness and other generic data quality dimensions in the context of the Bureau of Labor Statistics Covered Employment and Wages is described. The goal is to determine if data quality decreases as a result of receiving data earlier than the current due date. No significant quality deterioration is detected. Although this work describes a real case where dependencies play an important role, a general framework is not provided. In [21] a rigorous and pragmatic methodology for information quality assessment, called AIMQ, is presented. The AIMQ methodology consists of three components: i) a model representing what information quality means to information consumers and managers; ii) a questionnaire for measuring information quality along the dimensions; iii) analysis techniques for interpreting the assessments gathered by the questionnaire. The results of the proposed methodology highlight the fact that information quality is a single phenomenon where dimensions are not inherently independent. The table of correlations among dimensions, calculated on the basis of answers, is reported in the paper. The AIMQ methodology shows how knowledge on dependencies can be exploited to offer a comprehensive quality solution; however, the approach we presented here is more powerful than the one based on *correlation* [21]. In [1], within the context of business scenarios,

⁴ When the user cannot provide such functional relationship, the framework suggests the use of generic families of functions to approximate the required true functions; but still, functional parameters must be provided by the user.

a set of logical interdependencies among dimensions is presented. The authors define a taxonomy for data quality dimensions composed of *direct* and *indirect* attributes. Direct attributes represent the main dimensions which directly influence the results of business operations when a change in their values occurs. The indirect attributes determine and also contribute to the direct attributes; hence indirectly influence the results. The D^3Q could be used in a complementary way with [1] to evaluate if such logical interdependencies are satisfied in a real scenario and, therefore, allowing to validate such taxonomy. Finally, a data-driven tool for data quality management is described in [22] which suggests rules and identifies conformant and non-conformant records. The authors focused on the discovery of context-dependent rules, namely conditional functional dependencies (CFDs), i.e., that hold only over a data portion. The tool outputs a set of functional dependencies together with the context in which they hold. To avoid returning an unnecessarily large number of CFDs a set of interest metrics are evaluated, and comparative results using real datasets are reported.

5 Conclusions

A framework to discover the dependency structure between data quality dimensions is described. The Dependency Discovery Algorithm uses the learning dataset, compiled by the Dependency Discovery Model, to learn the BN model for the data quality dimensions assessed on a real world database. The BN model implements the Dependency Knowledge component. Accuracy, completeness, and consistency are assessed on three attributes of the Italian Social Security Contributors's List database. The framework is general while the obtained BN model is context dependent, i.e., it depends on the specific database which has been analyzed. The obtained results allow to conclude that BNs

- provide the database analyst with an intuitive and efficient representation of the dependency structure between data quality dimensions;
- allow consistent evaluation of the data quality level associated with each tuple of the database to be certified;
- allow to compare alternative data quality improvement strategies, to perform costs/benefits analysis and thus to implement optimal decision making.

Directions for future work concern the evaluation of the proposed approach on a richer set of data quality dimensions and on a larger number of attributes. It is relevant to study how BNs can be exploited to develop effective data quality improvement strategies in an information system, by implementing a trade-off between the *cost of non quality* and the budget available for data quality improvement.

Acknowledgments

The authors are grateful to the anonymous referees whose insightful comments enabled to make significant improvements to the paper. Daniele Barone would like to acknowledge the many valuable suggestions made by Fabrizio De Amicis, BSI Bank.

References

1. Gackowski, Z.: Logical interdependence of some attributes of data/information quality. In: Proc. of the 9th Intl. Conference on Information Quality, Cambridge, MA, USA, pp. 126–140 (2004)
2. Ballou, D.P., Pazer, H.L.: Designing information systems to optimize the accuracy-timeliness tradeoff. *Information Sys. Research* 6(1), 51–72 (1995)
3. Han, Q., Venkatasubramanian, N.: Addressing timeliness/accuracy/cost tradeoffs in information collection for dynamic environments. In: Proc. of the 24th IEEE Intl. Real-Time Systems Symposium, Washington, DC, USA, p. 108 (2003)
4. Ballou, D.P., Pazer, H.L.: Modeling completeness versus consistency tradeoffs in information decision contexts. *IEEE Trans. Knowl. Data Eng.* 15(1), 240–243 (2003)
5. Sadeghi, A., Clayton, R.: The quality vs. timeliness tradeoffs in the BLS ES-202 administrative statistics. In: Federal Committee on Statistical Methodology (2002)
6. Fisher, C., Eitel, L., Chengalur-Smith, S., Wang, R.: Introduction to Information Quality, p. 126. The MIT Press, Poughkeepsie (2006)
7. DeAmicis, F., Barone, D., Batini, C.: An analytical framework to analyze dependencies among data quality dimensions. In: Proc. of the 11th Intl. Conference on Information Quality, pp. 369–383. MIT, Cambridge (2006)
8. Burstein, F. (ed.): Handbook on decision support systems. Intl. handbooks on information systems. Springer, Heidelberg (2008)
9. Berner, E., Kasiraman, R., Yu, F., Ray, M.N., Houston, T.: Data quality in the outpatient setting: impact on clinical decision support systems. In: AMIA Annu. Symp. Proc., vol. 41 (2005)
10. Eckerson, W.: Data Quality and the Bottom Line: Achieving Business Success through a Commitment to High Quality Data. Technical report, The Data Warehousing Institute (2002)
11. Oei, J.L.H., Proper, H.A., Falkenberg, E.D.: Evolving information systems: meeting the ever-changing environment. *Information Sys. Journal* 4(3), 213–233 (1994)
12. Batini, C., Cappiello, C., Francalanci, C., Maurino, A.: Methodologies for data quality assessment and improvement. *ACM Comput. Surv.* 41(3), 1–52 (2009)
13. International Organization for Standardization: Software engineering – Software product Quality Requirements and Evaluation (SQuaRE) – data quality model. In: ISO/IEC 25012 (2008)
14. Jensen, F.V.: Bayesian Networks and Decision Graphs. Springer, Heidelberg (2001)
15. Pearl, J.: Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann Publishers Inc., San Francisco (1988)
16. Baldi, P., Frasconi, P., Smyth, P.: Modeling the internet and the WEB: Probabilistic methods and algorithms. Wiley, Chichester (2003)
17. Heckerman, D.: A tutorial on learning Bayesian networks. Technical Report MSR-TR-95-06, Microsoft Research (1995)
18. Elmagarmid, A.K., Ipeirotis, P.G., Verykios, V.S.: Duplicate record detection: A survey. *IEEE Trans. Knowl. Data Eng.* 19(1), 1–16 (2007)
19. Reiter, R.: On closed world data bases. In: Logic and Data Bases, pp. 55–76 (1977)
20. Jarke, M., Jeusfeld, M., Quix, C., Vassiliadis, P.: Architecture and quality in data warehouses: an extended repository approach (1999)
21. Lee, Y.W., Strong, D.M., Kahn, B.K., Wang, R.Y.: AIMQ: a methodology for information quality assessment. *Information Management* 40(2), 133–146 (2002)
22. Chiang, F., Miller, R.J.: Discovering data quality rules. *PVLDB* 1(1), 1166–1177 (2008)

Rationality of Cross-System Data Duplication: A Case Study

Wiebe Hordijk and Roel Wieringa

University of Twente, The Netherlands
{hordijkwtb,roelw}@cs.utwente.nl

Abstract. Duplication of data across systems in an organization is a problem because it wastes effort and leads to inconsistencies. Researchers have proposed several technical solutions but duplication still occurs in practice. In this paper we report on a case study of how and why duplication occurs in a large organization, and discuss generalizable lessons learned from this. Our case study research questions are why data gets duplicated, what the size of the negative effects of duplication is, and why existing solutions are not used. We frame our findings in terms of design rationale and explain them by providing a causal model. Our findings suggest that next to technological factors, organizational and project factors have a large effect on duplication. We discuss the implications of our findings for technical solutions in general.

Keywords: Data duplication, design rationale, field study.

1 Introduction

Data duplication is the phenomenon that two or more systems store and maintain representations of the same real-world fact. For example, two systems may store a list of all the countries in the world and their names; one system may have them in a database table and the other in a file, or the character sets in which they are stored may differ, or the database schemas. All these cases are examples of data duplication, because different data represent the same real-world facts and are separately maintained (not replicated). Data duplication is an important problem in practice, because it leads to wasted effort to keep the duplicated data consistent, errors due to inconsistencies in data and effort to correct those errors.

Current published solutions to data duplication view it as a technical problem, which can be solved for example by ensuring connectivity between systems over a network, by transforming data to a different syntax, or by matching data based on formally defined semantics [10]. Many solutions focus on automating the task of finding out which records in one or more information systems represent the same real-world facts. For example, some approaches match records based on the similarity of their contents [13,16], or on the similarity of their relations with other records [2]. A recent proposal uses Bayesian networks to represent evidence for possible object matches [6]. Some approaches try to match definitions in ontologies [8,14] and others use schema matching techniques [11]. Data

duplication is often cited as one of the major problems ERP systems can solve, but integration with existing systems poses its own risks [13].

Despite this plethora of technical solutions, data duplication and its associated problems still exist in practice. One explanation for this is that different databases can only be merged when they share an institutional world, for example because they must interoperate in a value chain or because they report to a common regulator [4]. In all other cases, data will remain duplicated. However, this does not explain why data is duplicated within one (large) organization, i.e. one institutional world. And unless we have such an understanding, we cannot predict with reasonable certainty what technical solutions would be effective.

In order to understand the problem of data duplication better, in section 4 of this paper we will analyze a case study in-depth. This will yield an improved understanding of the causes of duplication and the size of its effects. We describe our findings in the form of Reusable Rationale Blocks, which are tables that summarize the arguments for and against different design options. We then generalize our findings in the form of a cause/effect graph showing hypotheses about the causes of data duplication, and finally speculate about organizational and technical solutions to avoid duplication and to mitigate its negative effects.

2 Case Context

The organization in which we have performed this case study is a large not-for-profit organization in The Netherlands with a strong international presence. It employs about 3000 workers in The Netherlands and 2000 abroad.

During the study period the first author worked as a consultant for the organization. He was involved in application portfolio management, which deals with improvements in the information systems in place in the organization. The results of this study in a different form were used to shape these improvements.

3 Research Design

In our case study we try to answer the following research questions:

- Q1. Why does data get duplicated?
- Q2. (a) What are the negative effects of this, and (b) how costly are they?
- Q3. (a) How can the negative effects of duplication be avoided, (b) what role can existing technical solutions play in this, and (c) why are available technical solutions not used in our case?

We started by mapping hypothesized causes and effects of cross-system data duplication in the form of a cause/effect graph based on literature, anecdotes by stakeholders in the organization, and our own experience. This graph shows what organizational and technical factors lead to duplicated maintenance of data, and problems this duplication causes (the final version is shown in figure 1). These cause-effect relations were hypotheses in our research, i.e. we expected them to be confirmed or falsified in this particular case.

In the second phase we quantified the problem by counting inconsistencies in a number of data sets in several systems, and by estimating the effort involved in various activities needed due to duplication of data, such as duplicate entry, checks and corrections. In some cases the effort was recorded, in others personnel could give an estimate. This quantification underscores the importance of mitigating data duplication.

The third phase is more exploratory and involves searching for the mechanisms through which duplication is brought about. The unit of investigation was the individual decision, made in a project, to integrate a system under design with another system using particular technology, or not to integrate it. We assume that such decisions are made under bounded rationality, that is, decision makers make rational decisions given the limited information and time they have available [12]. This means that we needed to find the argumentation actually used in past projects. To be able to compare those arguments, we will present them in the form of reusable rationale blocks (RRBs). RRBs [5] are a simple technique to present advantages and disadvantages of options to solve a problem in a generalized table format so that they can be reused across similar problems. We have derived the argumentations from project documentation and interviews with the original stakeholders.

From the RRBs of individual projects we can see that some arguments are more important than others, depending on the context of the project. These context factors should appear as causes in the cause/effect graph created in the first phase. The arguments as represented in the RRBs are the mechanisms by which those context factors increase duplication. That way the rationale blocks both validate and explain the cause/effect graph. From the rationale blocks we will derive an improved cause/effect graph in the fourth phase.

In the fifth phase we analyze the generalizability of the results from the previous phase. Though our results come from a single organization, the research design is a multiple case design with multiple embedded units of research [17] and we can make a case for analytical generalization, also called “accumulation of knowledge” by realist methodologists [9]. In our case we intend to accumulate design-oriented knowledge that is true in sufficiently many cases without claiming to be true always in all cases. Practical knowledge often accumulates at this intermediate level of generalization [15].

The sixth phase consists of deriving practical solutions from our findings for organizations to minimize cross-system data duplication and to mitigate its negative effects. We will also discuss in which contexts the technical solutions proposed so far could contribute to the mitigation of the problem.

4 Results

4.1 Initial Cause/Effect Graph

A workshop was held in July 2009 with 10 representatives from the organizations IT department, not for the purpose of this research but to identify problems and solutions concerning application integration. The workshop was organized by the

first author together with another consultant, and attended by system maintainers, programmers, designers, an enterprise architect and a senior manager from the organization. From the results of the workshop the first author has drawn an initial cause/effect graph.

4.2 Identifying Duplicate Data

We have created a matrix matching systems against the data stored in those systems, and from it we have identified the following data objects which are used in at least three systems. We have only counted systems where the data were maintained by users, not systems which merely store a copy of data which they automatically receive from another system.

- Countries, including their names, sometimes in different languages, and country codes such as from ISO 3166 and national standards.
- Nationalities occur in some of the systems storing countries.
- Employees, sometimes with specific information such as phone numbers, sometimes only with their account name for authorization purposes.
- Organizational units
- Foreign offices, a generic name for operations of the organization in other countries. There are about 150 of them. Some systems store only a subset because their business processes apply to specific types of offices.
- Contacts, a generic name for all kinds of external parties, including persons and other organizations that the organization deals with. Many systems have their own subset of contacts, but there is overlap, e.g. when a contact occurs in the financial system, a CRM system and someones email contacts directory.

4.3 Quantification of the Problem

We have counted the number of inconsistencies among systems concerning several of the data sets listed above by manually comparing database dumps from the systems and then estimated the wasted effort caused by this. We give the results for inconsistencies between countries tables and we give estimates of the effort wasted by duplication in general.

Inconsistencies in Country Lists. We compared the countries tables with the ISO 3166¹ standard, for which the Dutch Language Union² provides official Dutch country names. In table 1 we list the inconsistencies per system and per type as percentage of the total number of countries. Inconsistencies are classified as 'unmatched' countries, where a country is missing from either the system or the standard, or 'spelling' where the spelling of the name differs. We have also estimated the size of the user groups and the usage frequency. We can make some observations on this data.

¹ http://www.iso.org/iso/country_codes.htm

² <http://taalunieversum.org/taalunie/>

Table 1. Inconsistencies in countries tables per system as percentage of total number of countries

| System | Introduction | Users | Frequency | Unmatched | Spelling | Total |
|-----------|--------------|-------|-------------|-----------|----------|-------|
| System 1 | 1-1-1997 | 5000 | daily | 4% | 8% | 12% |
| System 2 | 1-1-1998 | 20 | daily | 8% | 12% | 20% |
| System 3 | 1-4-2001 | 20 | daily | 7% | 6% | 13% |
| System 4 | 1-8-2001 | 500 | 5x per year | 50% | 5% | 55% |
| System 5 | 1-8-2002 | 20 | daily | 14% | 9% | 23% |
| System 6 | 1-1-2003 | 500 | daily | 8% | 26% | 33% |
| System 7 | 1-10-2003 | 5000 | daily | 4% | 3% | 7% |
| System 8 | 1-1-2004 | 10 | ad hoc | 8% | 13% | 21% |
| System 9 | 1-10-2006 | 30 | daily | 11% | 19% | 30% |
| System 10 | 1-1-2007 | 5000 | ad hoc | 7% | 6% | 13% |
| System 11 | 1-1-2007 | 500 | daily | 14% | 14% | 28% |
| System 12 | 1-7-2007 | 500 | 1x per year | 9% | 23% | 32% |
| System 13 | 1-6-2008 | 10 | ad hoc | 10% | 21% | 31% |

Some systems have many unmatched countries. This can be explained because these systems only need specific subsets of countries, e.g. only those in which the organization has offices, or because countries have ceased to exist, such as Yugoslavia or the USSR, but have not been removed from a system. This may not be a problem for users of a system itself, but does pose problems when trying to integrate data from multiple systems, such as for reporting.

Spelling differences are remarkably frequent. Spelling issues arise often with respect to geographical names, but when analyzing the data we found that many of these inconsistencies were simple typing errors.

There is no relation between the age of a system and the number of inconsistencies, though we will see some evidence in subcases 3 and 4 in paragraph 4.4 that systems built on older technological platforms are harder to integrate. We do observe that generally systems with larger user groups have fewer inconsistencies than less-used systems. This can probably be explained because much-used systems receive more corrections at the request of their users.

Effort due to Duplication. In System 4, data about foreign offices are periodically aggregated from several other systems for financial and policy evaluation and prediction. These data are exported from other systems as MS Excel files, after which some manual polishing and formatting is done and the files are imported in System 4. Examples of this 'polishing' are removing negative numbers and checking total amounts after importing files. This process happens 5 times per year and takes an estimated total of 16 person-hours of all workers involved each time. This estimate was given by the application maintenance staff.

System 12 is a reporting system in which data from a large number of other systems are brought together to create management overviews of foreign operations.

Table 2. Integration effort per system

| System | Integration effort |
|-----------|---|
| System 1 | Inconsistencies in data cause users to request corrections through service desk |
| System 4 | File export/import, communication between maintainers, data checks, manual corrections (estimate 16 hrs per time, 5 times per year) |
| System 5 | Effort is hidden in large effort for periodical data consolidation |
| System 6 | Changes in countries via service desk |
| System 9 | Changes in countries via service desk. Manual corrections in output by users. |
| System 11 | Manual maintenance of translation tables for systems to exchange data with. |
| System 12 | Reporting takes 4 person-months per year. |

Partly because of differences in reference data (such as countries) in the source systems, creating these reports takes a lot of manual effort. The recurring effort for producing these reports is 4 person-months.

Table 2 lists qualitative and some quantitative data about effort spent on manual integration of data between systems. The total waste of effort adds up to several person-months per year.

4.4 Identifying Mechanisms That Cause Duplication

If we want to build a system without duplicating the effort to maintain the data, then the first thing we need to do is find out whether there is a source system to get the data from. The data in the source system must be of sufficient quality to serve the processes of the target system. Quality attributes of the data on which integration options have influence are (adapted from [7]):

- Accuracy: the degree to which data correctly reflect an external truth.
- Completeness: the degree to which data reflect an entire external truth.
- Timeliness: the speed with which data are adapted to changes in an external truth.

When a source system with suitable data quality is found, there are some options to integrate the data between the source and target system. Each of these possibilities has advantages and disadvantages which the project must weigh against the success criteria that the stakeholders apply to this project. This can make it logical for a project to deliver an isolated system, that duplicates data stored elsewhere too, even if that is disadvantageous for the organization as a whole.

- Manual re-typing of the data. This is flexible and no investment is needed but takes a lot of effort and usually leads to many typing errors.
- Manually exporting and importing files. Also very flexible and takes a small project investment but still costs effort, and errors still occur, especially when manual “improvements” to the data are necessary.
- Automatically exchanging files. Takes a slightly bigger investment. Errors still occur, e.g. due to incomplete files.

Table 3. RRB showing the general pros and cons of integration options. ++ is very good value and - - a very bad value for the stakeholder, respectively.

| Stakeholder | Quality attribute | Importance | Manual typing | Manual files | Autom. files | DB link | Messaging |
|-------------------------|--------------------------------------|------------|---------------|--------------|--------------|---------|-----------|
| Project; Owner | Project investment (+ is lower) | + | ++ | + | -- | - | -- |
| Project | Project risk (+ is lower) | + | ++ | + | -- | - | -- |
| Owner; Maintenance team | Independence of other systems | + | ++ | + | - | -- | - |
| Owner; User | Business process flexibility | + | ++ | + | -- | -- | - |
| User; Manager | Data integration effort (+ is lower) | ++ | -- | - | + | + | + |
| User | Data accuracy | ++ | -- | - | + | + | + |
| User | Data timeliness | ++ | -- | -- | - | + | + |
| Maintenance team | Maintainability | + | + | - | - | + | ++ |

- Database link to the source system. The investment is often smaller than with file export/import but the solution is less flexible because the source system cannot change its data structure without breaking the target system. The data are always up to date but the target system will depend for its availability on the source system and the network.
- Messaging, e.g. using web services. This option takes the largest investment but it is more flexible than a database link and more reliable and flexible than file exchange.

Together with professionals from the organization we have identified general quality attributes for these options. This general knowledge was codified in the form of a RRB in Table 3. It should be read as follows: all else being equal, the data integration effort will be worst (= highest) when manual typing is chosen and best when one of the automated options is chosen; data integration effort will usually matter more to stakeholders than the initial investment. We do not claim that the RRB of Table 3 is true in all cases but we do claim that this analysis method can be used in similar cases too, and will often contain very similar decision rationales.

We have identified four decisions in past projects to integrate systems in a certain way by interviewing the original stakeholders and reviewing project documentation. We have investigated how the context factors of a specific project shape the arguments that feed into the decision and lead to different outcomes. These arguments, represented below by means of RBBs, will be used to derive and corroborate hypotheses about causes and effects of data duplication. We aggregate the hypotheses into a causal effect graph in Figure 1 and Table 6.

We have noticed in this phase that most project decisions are not documented with their associated rationale (and sometimes not at all). This makes it essential

for case study research to have access to original stakeholders, which has driven our choice of cases.

Subcase 1. System 11: Countries. When system 11 was under development, architects have actively looked for a suitable source system for countries and nationalities. Documentation describes why systems 5 and 6 cannot deliver data with the required quality. Other source systems have not been considered, and architects involved in the process indicate that it was unknown at the time that other systems also stored country tables. The decision was made to give System 11 its own countries table. This case leads to hypothesis *H1: lack of knowledge about the architectural landscape of the organization is a contributing factor to data duplication by increasing the cost of search for potential source systems.*

When systems 5 and 6 turned out not to have the desired data quality, the project first tried to request from the respective owners of those systems to improve the data and then share it. The owners of both systems responded in a tactful way that they did not see this as their task. This teaches us that for systems to integrate their data an organization first needs to establish the organizational responsibilities and capabilities to maintain and provide data with the right quality. We postulate two hypotheses. *H2: poor or unknown data quality in potential source systems leads to data duplication. H3: Unwillingness to establish organizational dependencies leads to data duplication.*

Subcase 2. System 11: Currencies. At the same time when designing system 11, the system needed data about which currencies can be used in countries in the world and spot rates of those currencies against the euro. The currencies are updated about as often as countries themselves, but spot rates change daily. System 6 is a natural source for these data because it is an ERP system used for financial administration throughout the organization. It can be trusted to have accurate data about currencies and rates. This is consistent with hypothesis H2.

Table 4 shows the advantages and disadvantages of integration options between systems 6 and 11, adapted from Table 3 to the specific context factors of this project. We use the table to detect how variations in context factors for individual projects determine which option is best in individual decisions.

System 11 is a relatively new system built on a modern development platform and uses messaging between its own components. That makes the project investment and the project risk for the option ‘messaging’ lower. The same holds for ‘automated file transfer’, with which the maintenance team of system 6 has broad experience. We postulate hypotheses *H4: availability of integration infrastructure decreases the chance of data duplication by decreasing the relative cost of integration.*

The development of system 11 was a very large project of which this particular integration formed only a small part. This reduces the importance of the project investment. *H5: High pressure on a project increases the chance of data duplication by increasing the importance of project risk and project investment.*

For the business process of system 11 it is important that currencies and rates are received correctly each day. This increases the relative importance of

Table 4. Pros and cons of integration options for currencies in System 11; differences with Table 3 in brackets

| Stakeholder | Quality attribute | Importance | Manual typing | Manual files | Autom. files | DB link | Messaging |
|-------------------------|--------------------------------------|------------|---------------|--------------|--------------|---------|-----------|
| Project; Owner | Project investment (+ is lower) | (-) | ++ | + | (-) | - | (-) |
| Project | Project risk (+ is lower) | + | ++ | + | (-) | - | (-) |
| Owner; Maintenance team | Independence of other systems | + | ++ | + | - | -- | - |
| Owner; User | Business process flexibility | + | ++ | + | -- | -- | - |
| User; Manager | Data integration effort (+ is lower) | (+++) | -- | - | + | + | + |
| User | Data accuracy | (+++) | -- | - | + | + | + |
| User | Data timeliness | (+++) | -- | -- | (+) | + | + |
| Maintenance team | Maintainability | + | + | - | - | + | ++ |

accuracy and timeliness. Because the rates only change daily and automated files are suitable for daily transfers, the automated files option gets a + for timeliness. Because of the large amount of data to be transferred each day, data integration effort is relatively important. *H6: high-volume data is less susceptible for duplication than low-volume data, because the data integration effort is higher.*

In Table 4 we can see that the three automated options are still viable. Database link was not chosen because it leads to tight coupling between the systems. Messaging was favored over automated file transfer because it was thought to lead to more maintainable systems.

Subcase 3. System 4: Countries and financial data. System 4 is used to make budgets and to evaluate the financial performance of the organization as a whole. It receives financial information from system 6, the ERP system, and the countries list in system 6 is good enough for system 4, so system 6 is a viable source system. The advantages and disadvantages of the options to connect these systems are listed in Table 5.

System 4 is an old system built in MS Access, which does not offer standard messaging technology, just like the ERP system 6. Use of messaging is only possible with a large investment in technology and would incur considerable risk to a project. This corroborates hypothesis H4: availability of integration infrastructure decreases duplication.

There are plans to rebuild system 4 with .NET technology to a web-based system. This means that any investment in system 4 in the short term should be profitable in a short time. This increases the relative importance of the quality attribute 'Project investment'.

The data collected in system 4 and its business rules change slightly from year to year. These changes typically are agreed upon shortly before a new version

Table 5. Pros and cons of integration options for system 4; differences with Table 3 in brackets

| Stakeholder | Quality attribute | Importance | Manual typing | Manual files | Autom. files | DB link | Messaging |
|-------------------------|--------------------------------------|------------|---------------|--------------|--------------|---------|-----------|
| Project; Owner | Project investment (+ is lower) | (++) | ++ | + | -- | - | (---) |
| Project | Project risk (+ is lower) | + | ++ | + | -- | - | (---) |
| Owner; Maintenance team | Independence of other systems | + | ++ | + | - | -- | - |
| Owner; User | Business process flexibility | (++) | ++ | + | -- | -- | - |
| User; Manager | Data integration effort (+ is lower) | (+) | -- | (-/+) | + | + | + |
| User | Data accuracy | ++ | -- | (-/+) | + | + | + |
| User | Data timeliness | (+) | -- | -- | - | + | + |
| Maintenance team | Maintainability | + | + | - | - | + | ++ |

of the system goes live. This sometimes makes it necessary to manually change the contents of files transferred from system 6 to system 4. This increases the relative importance of the quality attribute ‘Business process flexibility’. This leads to *H7: required flexibility in business processes can lead to data duplication.*

Data integration between these systems is performed by employees who have been doing this for almost 10 years now. They know exactly what to do and for what kinds of errors to check. For the systems architecture this means that the option of manual file transfer scores better on the quality attributes ‘Data integration effort’ and ‘Data accuracy’. We propose *H8: expertise in maintaining duplicated data keeps duplication in place by minimizing its drawbacks.*

Data timeliness is less important for this integration because the whole process happens only 5 times per year. Under these specific circumstances, manual file transfer has the optimal balance between project investment and integration effort. This motivates *H9: frequently updated data is less likely to be duplicated than data which is less frequently updated.*

Subcase 4. System 12: Reporting data. System 12 is a system in which large amounts of data about the organizations foreign offices are aggregated into management information. Before this system was developed there was a lot of uncertainty about how and in what formats source systems would be able to deliver their data. For those reasons, business process flexibility and independence of other systems were very important. This corroborates hypotheses H2: poor or unknown data quality increases duplication, and H7: required flexibility in business processes can lead to data duplication.

The amount of data was not too large to make manual checks and corrections unfeasible. The reports from the system were only needed once per year. This decreases the relative importance of Data integration effort, Data timeliness and

Maintainability. This corroborates H6: high-volume data is duplicated less often, and H9: more frequently updated data is duplicated less often.

The resulting rationale table for system 12 is very much like that for system 4 in Table 5, and for reasons of space we omit the specific table for system 12. For the integration of data into system 12, manual file transfer was chosen for much the same reasons as for system 4.

4.5 Lessons Learned

Based on the hypotheses described in the cases above we have adapted the cause-effect graph drawn at the start of our project. The result is shown in figure 1.

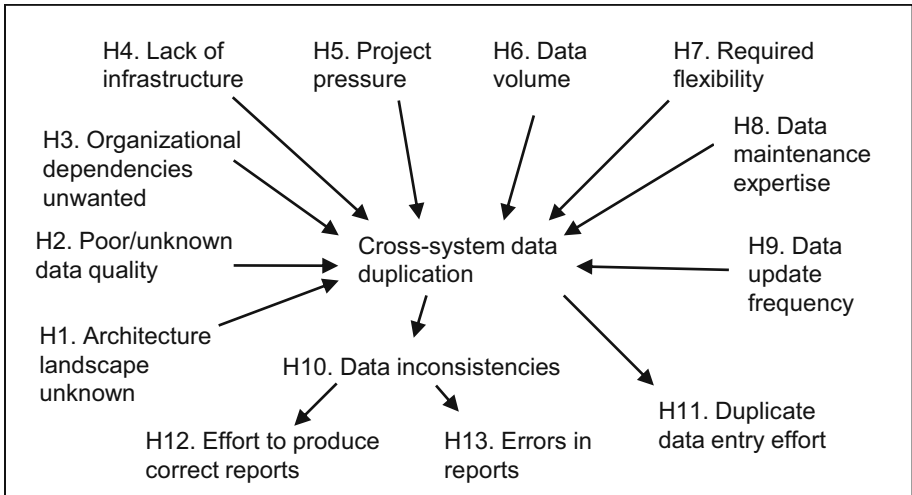


Fig. 1. Final Cause/Effect graph. Nodes are observable phenomena, marked with the hypotheses in which they play a role; arrows are positive influences

The nodes in Figure 1 are marked with the hypothesis numbers in which they play a role. In Table 6 we summarize the hypotheses with their evidence.

4.6 Generalizability

It is not possible to find in one case study arguments for generalization to all other cases. But it is possible to indicate which claims could be generalized to an interesting set of other cases. We claim that our results are generalizable to, and therefore reusable in, situations where the same mechanisms are in place as in our case study.

4.7 Solutions

Solutions to the problems around data duplication consist of a mix of organizational, technical and project management elements, just like the problems

Table 6. Hypotheses and their evidence

| Hypothesis | Evidence |
|--|--|
| H1: lack of knowledge about the architectural landscape of the organization is a contributing factor to data duplication by increasing the cost of search for potential source systems | Observation in Subcase 1. In the other subcases, source systems were found, but the search effort spent is unknown. Intuitive and matches anecdotal evidence from other organizations. |
| H2: poor or unknown data quality in potential source systems leads to data duplication | In subcase 1, data was maintained for a specific business process and did not meet the quality requirements of another process. In subcase 4, quality of available data was not known at design time, which led to duplication. In subcase 2 good data quality enabled integration. Intuitive and matches anecdotal evidence from other organizations. |
| H3: Unwillingness to establish organizational dependencies leads to data duplication | In Subcase 1, a potential data supplier was unwilling to maintain data at a higher quality than needed for their own process. Seems organization-dependent. |
| H4: availability of integration infrastructure decreases duplication. | In subcase 2, messaging infrastructure was available. In subcase 3 it was not; it would have made messaging a more likely option. Intuitive and matches anecdotal evidence from other organizations. |
| H5: High pressure on a project increases the chance of data duplication by increasing the importance of project risk and project investment | Observed only directly in subcase 2. In other subcases we can see that investments for improvement are not made until other major changes to systems are necessary. Seems organization-dependent. |
| H6: High-volume data is less susceptible for duplication than low-volume data, because the data integration effort is higher | Observed in subcase 2 (high volume) and subcase 4 (also high, but outweighed by other context factors). Intuitive and matches anecdotal evidence from other organizations. |
| H7: Required flexibility in business processes leads to data duplication | In subcases 3 and 4, required flexibility led to duplication. Seems organization-dependent. |
| H8: expertise in maintaining duplicated data keeps duplication in place by minimizing its drawbacks | Observed in subcase 3. Seems organization-dependent. |
| H9: frequently updated data is less likely to be duplicated than data which is less frequently updated | Observed in subcases 3 and 4 (low frequency) and subcase 2 (high frequency). Intuitive and matches anecdotal evidence from other organizations. |
| H10: Data duplication leads to inconsistencies between data sets | Observed and quantified for countries and offices in paragraph 4.3. Intuitive and matches anecdotal evidence from other organizations. |
| H11: Duplication costs extra effort to maintain data in multiple systems | Observed and quantified in paragraph 4.3. Intuitive and matches anecdotal evidence from other organizations. |
| H12: Due to inconsistencies between data, making reports with data from multiple systems takes more effort | Observed and quantified in paragraph 4.3. Intuitive and matches anecdotal evidence from other organizations. |
| H13: Inconsistencies between data sets cause errors in reports | Anecdotal evidence. Intuitive and matches anecdotal evidence from other organizations. |

themselves do. In this paragraph we speculate about solutions and their merits. Since this is not the main topic of this research effort, the solutions are presented only briefly.

To combat the lack of information about the application landscape we need a well-documented architecture landscape (solution S1), including information about what data is stored in which system. This documentation should be kept up to date and be accessible to projects.

The problem of systems only catering for their own business processes is hard to beat. One solution could be to establish an organizational entity that is responsible for maintaining data for the entire organization (S2). This has been done at several organizations where properly maintaining large quantities of data is important. This opens the extra opportunity to supply such data to other parties for added benefit. Another solution would be to add the responsibility for maintaining a particular data set to a department, along with the capabilities (people, budget) to fulfill that responsibility (S3). This is easier to do in some organizations than in others.

The relative importance of long term, organization-wide quality attributes should be increased at the cost of short-term, project-only interests. One possible way to achieve this is by adding an architect to the project team who gets paid by the IT organization instead of the system owner (S4). Another way is to use an architecture review process in which the designed system is reviewed by organization architects against criteria which reflect long term interests (S5).

Even if data is still being maintained in multiple systems, the problems that come with inconsistencies between the data sets can be minimized by agreeing on standards for the data and on procedures to check data against the standard (S6). For example, if our organization would agree on using the ISO 3166 standard for countries and all systems were up to date then integrating other data which uses countries as reference data would be much easier.

Central investments in infrastructure, e.g. for messaging, can decrease the project investment needed to provide an optimal solution which would otherwise be too expensive and too risky for a project (S7).

After all these organizational problems have been solved, the technological solutions listed in section II become viable. Data warehouse technology (S8) can make reporting easier, thus decreasing the reporting effort and reporting errors. A project to introduce a data warehouse in the organization is currently starting. First, however, inconsistencies in the data fed into the data warehouse from different systems must be reconciled. Entity linkage technologies (S9) and ontologies (S10) can help with that. These technologies are useful to integrate large amounts of data, but to start using them a considerable investment in skills will be needed. That means they are most useful in environments where the cost of manual integration and the risk posed by inconsistencies outweigh the investment in new technologies, that is, in situations where either the volume of the data are large, errors are very dangerous or expensive, or new technologies are easily integrated. Our organization does not have any of these characteristics. The volume of data in our organization is such that low-tech solutions are

adequate to solve the inconsistencies, and the organization has a policy of using only proven technology and not be technically innovative. This means that for our organization the risk associated with introducing new technologies outweighs the potential benefits.

Solutions S1, S2, S5, S6, S7 and S8 have been reported to the organization and are currently under review. Possible solutions for example involve changes to existing systems, to infrastructure, to procedures or to organizational structures. At the time of writing, these proposed changes are being considered in the organizations application and project portfolio management processes.

5 Discussion, Conclusions and Future Work

We have performed a series of historical case studies in a single organization. The external validity of such a research design is always questionable. The initial theory in the form of the cause/effect graph, the general rationale table and the information about individual cases were all taken from the organization under investigation. In paragraph 4.6 we have reflected on the generalizability of the results in the cause/effect graph of Figure 1.

We have provided quantitative results about the problems caused by data duplication. The data about the causes of data duplication is mostly qualitative in nature. At present this is the best we can do, and given the current state of the research we think that even such qualitative results are a contribution to the advancement of theory in this field.

Table 3 with its advantages and disadvantages of integration options is not meant to represent a general truth about relative preferences of these options: we do not claim that messaging for example is always better than database links; rather we have shown that these preferences are situation-dependent. The methodology of rationale blocks, however, can be reused as an intuitive tool for decision support, rationale documentation and as a research method to investigate the sensitivity of option preferences to context factors, as we have done. It is a pragmatic approach which stakeholders can quickly understand. Discussions in the course of our investigation immediately centered on the pros and cons of options, not about the notation technique or methodology. Rationale tables allow arguments for individual decisions to be reused in other situations. We have shown that the decision to integrate or not to integrate systems can often be explained using arguments that are rational from the standpoint of the decision maker.

Our results show that organizational, technical and project factors can cause data duplication across systems in an organization. We have suggested a set of mitigation strategies to reduce data duplication and to decrease its negative effects. These can be used by organizations to get projects to give higher priority to organization-wide long term interests and to deliver better integrated systems.

Future work should include repeating this research in other organizations and evaluation of the performance of the introduced solutions.

We thank the anonymous reviewers for their helpful suggestions for improving this paper.

References

1. Ananthakrishna, R., Chaudhuri, S., Ganti, V.: Eliminating fuzzy duplicates in data warehouses. In: VLDB 2002: Proceedings of the 28th international conference on Very Large Data Bases, pp. 586–597 (2002)
2. Bhattacharya, I., Getoor, L.: Deduplication and group detection using links. In: Workshop on Link Analysis and Group Detection (LinkKDD 2004), Seattle, WA, USA. ACM, New York (2004)
3. Cohen, W.W.: Integration of heterogeneous databases without common domains using queries based on textual similarity. In: SIGMOD 1998: Proceedings of the 1998 ACM SIGMOD international conference on Management of data, pp. 201–212 (1998)
4. Colomb, R.M., Ahmad, M.N.: Merging ontologies requires interlocking institutional worlds. *Appl. Ontol.* 2(1), 1–12 (2007)
5. Hordijk, W., Wieringa, R.: Reusable rationale blocks: Improving quality and efficiency of design choices. In: Dutoit, A.H., McCall, R., Mistrik, I., Paech, B. (eds.) *Rationale Management in Software Engineering*, pp. 353–371. Springer, Heidelberg (2006)
6. Ioannou, E., Niederee, C., Nejdil, W.: Probabilistic entity linkage for heterogeneous information spaces. In: Bellahsene, Z., Léonard, M. (eds.) *CAiSE 2008*. LNCS, vol. 5074, pp. 556–570. Springer, Heidelberg (2008)
7. Jiang, L., Borgida, A., Mylopoulos, J.: Towards a compositional semantic account of data quality attributes. In: Li, Q., Spaccapietra, S., Yu, E., Olivé, A. (eds.) *ER 2008*. LNCS, vol. 5231, pp. 55–68. Springer, Heidelberg (2008)
8. Noy, N.F.: Semantic integration: a survey of ontology-based approaches. *SIGMOD Rec.* 33(4), 65–70 (2004)
9. Pawson, R., Tilley, N.: *Realistic Evaluation*. SAGE Publications, London (1997)
10. Pollock, J.T.: Integration’s dirty little secret: It’s a matter of semantics. Technical report, *Modulant* (2002)
11. Rahm, E., Bernstein, P.A.: A survey of approaches to automatic schema matching. *The VLDB Journal* 10(4), 334–350 (2001)
12. Simon, H.A.: *The Sciences of the Artificial*, 3rd edn. MIT Press, Cambridge (1996)
13. Sumner, M.: Risk factors in enterprise-wide/erp projects. *Journal of Information Technology* 15(4), 317–327 (2000)
14. Uschold, M., Gruninger, M.: Ontologies and semantics for seamless connectivity. *SIGMOD Rec.* 33(4), 58–64 (2004)
15. Wieringa, R.J.: Design science as nested problem solving. In: Proceedings of the 4th International Conference on Design Science Research in Information Systems and Technology, Philadelphia, pp. 1–12 (2009)
16. Winkler, W.E.: The state of record linkage and current research problems. Technical report, Statistical Research Division, U.S. Census Bureau (1999)
17. Yin, R.K.: *Case study research: design and methods*, 3rd edn. Applied Social Research Methods Series, vol. 5. SAGE Publications, Thousand Oaks (2003)

Probabilistic Models to Reconcile Complex Data from Inaccurate Data Sources

Lorenzo Blanco, Valter Crescenzi, Paolo Merialdo, and Paolo Papotti

Università degli Studi Roma Tre
Via della Vasca Navale, 79 – Rome, Italy
{blanco,crescenz,merialdo,papotti}@dia.uniroma3.it

Abstract. Several techniques have been developed to extract and integrate data from web sources. However, web data are inherently imprecise and uncertain. This paper addresses the issue of characterizing the uncertainty of data extracted from a number of inaccurate sources. We develop a probabilistic model to compute a probability distribution for the extracted values, and the accuracy of the sources. Our model considers the presence of sources that copy their contents from other sources, and manages the misleading consensus produced by copiers. We extend the models previously proposed in the literature by working on several attributes at a time to better leverage all the available evidence. We also report the results of several experiments on both synthetic and real-life data to show the effectiveness of the proposed approach.

1 Introduction

As the Web is offering increasing amounts of data, important applications can be developed by integrating data provided by a large number of data sources. However, web data are inherently imprecise, and different sources can provide conflicting information. Resolving conflicts and determining what values are (likely to be) true is a crucial issue to provide trustable reconciled data.

Several proposals have been developed to discover the true value from those provided by a large number of conflicting data sources. Some solutions extend a basic vote counting strategy by recognizing that values provided by accurate sources, i.e. sources with low error rates, should be weighted more than those provided by others [10,12]. Recently, principled solutions have been introduced to consider also the presence of sources that copy from other sources [6]. As observed in [1], this is a critical issue, since copiers can cause misleading consensus on false values. However, they still suffer some limitations, as they are based on a model that considers only simple (atomic) data. Sources are seen as providers that supply data about a collection of objects, i.e. instances of a real world entity, such as a collection of stock quotes. Nevertheless, it is assumed that objects are described by just one attribute, e.g. the price of a stock quote. On the contrary, data sources usually provide complex data, i.e. collections of tuples with many attributes. For example, sources that publish stock quotes always deliver values for price, volume, max and min values, and many other attributes.

Existing solutions, focused on a single attribute only, turn out to be rather restrictive, as different attributes, by their very nature, may exhibit drastically different properties and evidence of dependence. This statement is validated by the observation that state-of-the-art algorithms, when executed on real datasets lead to different conclusions if applied on different attributes published by the same web sources.

| Source 1 | | | | Source 2 | | | | Source 3 | | | |
|----------|-------------|-----|-----|----------|-------------|-----|-----|----------|---------|------------|------------|
| | volume | min | max | | volume | min | max | | volume | min | max |
| AAPL | 699.9k | 90 | 150 | AAPL | 699.9k | 90 | 150 | AAPL | 699.9 | 90 | 150 |
| GOOG | 1.1m | 380 | 545 | GOOG | 1.1m | 380 | 545 | GOOG | 1100.0k | 381 | 541 |
| YHOO | 125k | 21 | 48 | YHOO | 125k | 21 | 48 | YHOO | 125.0k | 21 | 44 |

| Source 4 | | | | <i>True values</i> | | | |
|----------|---------|------------|------------|--------------------|----------------|------------|------------|
| | volume | min | max | | volume | min | max |
| AAPL | 699.9k | 91 | 150 | AAPL | <i>699.9k</i> | <i>90</i> | <i>150</i> |
| GOOG | 1100.0k | 381 | 541 | GOOG | <i>1100.0k</i> | <i>380</i> | <i>545</i> |
| YHOO | 125.0k | 21 | 44 | YHOO | <i>125.0k</i> | <i>21</i> | <i>48</i> |

Fig. 1. Three sources reporting stock quotes values.

This behavior can be caused by two main reasons: lack of evidence (copiers are missed) or misleading evidence (false copiers are detected). In Figure 1 we make use of an example to illustrate the issues: four distinct sources report financial data for the same three stocks. For each stock symbol are reported three attributes: volume, min and max values of the stock. The fifth table shows the true values for the considered scenario: such information is not provided in general, in this example we consider it as given to facilitate the discussion.

Consider now the first attribute, the stock volume. It is easy to notice that Source 1 and Source 2 are reporting the same false value for the volume of GOOG (errors are in bold). Following the intuition from [6], according to which copiers can be detected as the sources share false values, they should be considered as copiers. Conversely, observe that Source 3 and Source 4 report only true values for the volume and therefore there is not any significant evidence of dependence. The scenario radically changes if we look at the other attributes. Source 3 and Source 4 are reporting the same incorrect values for the max attribute, and they also make a common error for the min attribute. Source 4 also reports independently an incorrect value for the min value of AAPL. In this scenario our approach concludes that Source 3 and Source 4 are certainly dependent, while the dependency between Source 1 and Source 2 would be very low. Using previous approaches and by looking only at the volume attribute, Source 1 and Source 2 would be reported as copiers because they share the same formatting rule for such data (i.e., false copiers detected), while Source 3 and Source 4 would be considered independent sources (i.e., real copiers missed).

In this paper, we extend previous proposals to deal with sources providing complex data, without introducing any remarkable computation efforts. We formally

describe our algorithms and give a detailed comparison with previous proposals. Finally, we show experimentally how the evidence accumulated from several attributes can significantly improve the performance of the existing approaches.

Paper Outline. The paper is organized as follows: Section 2 illustrates related work. Section 3 describes our probabilistic model to characterize the uncertainty of data in a scenario without copier sources. Section 4 illustrates our model for analyzing the copying relationships on several attributes. Section 5 presents the result of the experimental activity. Section 6 concludes the paper.

2 Related Work

Many projects have been active in the study of imprecise databases and have achieved a solid understanding of how to represent uncertain data (see 5 for a survey on the topic). The development of effective data integration solutions making use of probabilistic approaches has also been addressed by several projects in the last years. In 8 the redundancy between sources is exploited to gain knowledge, but with a different goal: given a set of text documents they assess the quality of the extraction process. Other works propose probabilistic techniques to integrate data from overlapping sources 9.

On the contrary, so far there has been little focus on how to populate such databases with sound probabilistic data. Even if this problem is strongly application-specific, there is a lack of solutions also in the popular fields of data extraction and integration. Cafarella et al. have described a system to populate a probabilistic database with data extracted from the Web 4, but they do not consider the problems of combining different probability distributions and evaluating the reliability of the sources.

TruthFinder 12 was the first project to address the issue of discovering true values in the presence of multiple sources providing conflicting information. It is based on an iterative algorithm that exploits the mutual dependency between source accuracy and consensus among sources. Other approaches, such as 11 and 10, presented fixpoint algorithms to estimate the true values of data reported by a set of sources, together with the accuracy of the sources. These approaches do not consider source dependencies and they all deal with simple data.

Some of the intuitions behind TruthFinder were formalized by Dong et al. 6 in a probabilistic Bayesian framework, which also takes into account the effects related to the presence of copiers among the sources. Our probabilistic model is based on such Bayesian framework and extends it to the case with sources that provide complex data. A further development by the same authors also considers the variations of truth values over time 7. This latter investigation applies for time evolving data and can lead to identify outdated sources.

3 Probabilistic Models for Uncertain Web Data

In our setting, a source that provides the values of a set of properties for a collection of objects is modeled as a *witness* that reports an *observation*. For

example, on the Web there are several sources that report the values of price, volume, dividend for the NASDAQ stock quotes. We say that these sources are witnesses of all the cited properties for the NASDAQ stock quotes.

Different witnesses can report inconsistent observations; that is, they can provide inconsistent values for one or more properties of the same object. We aim at computing: (i) the probability that the observed properties of an object assume certain values, given a set of observations that refer to that object from a collection of witnesses; (ii) the accuracy of a witness with respect to each observed property, that is, the probability that a witness provides the correct values of each observed property for a set of objects. With respect to the running example, we aim at computing the probability distributions for volume, min and max values of each observed stock quote, given the observations of the four witnesses illustrated in Figure 1. Also, for each witness, we aim at computing its accuracies in providing a correct value for volume, min and max property.

We illustrate two models of increasing complexity. In the first model we assume that each witness provides its observations independently from all the other witnesses (*independent witnesses assumption*). Then, in Section 4 based on the framework developed in [6], we remove this assumption and consider also the presence of witnesses that provide values by copying from other witnesses. The first model is developed considering only one property at a time, as we assume that a witness can exhibit different accuracies for different properties. More properties at a time are taken into account in the second model, which considers also the copying dependencies. As we discussed in the example of Figure 1, considering more properties in this step can greatly affect the results of the other steps, and our experimental results confirmed this intuition, as we report in Section 5.

For each property, we use a discrete random variable X to model the possible values it assumes for the observed object. $\mathcal{P}(X = x)$ denotes the prior probability distribution of X on the x_1, \dots, x_{n+1} possible values, of which one is correct, denoted x_t , and the other n are incorrect. Also, let \dot{x} denote the event $X = x = x_t$, i.e. the event “ X assumes the value x , which is the correct value x_t ”. The individual observation of a witness is denoted o ; also, $v(o)$ is used to indicate the reported value. The *accuracy* of a witness w , denoted A , corresponds to the conditional probability that the witness reports x_t , given \dot{x} ; that is: $A = P(o|\dot{x})$, with $v(o) = x_t$.

In the following we make several assumptions. First, we assume that the values provided by a witness for an object are independent on the values provided for the other objects (*independent values assumption*). Also, we assume that the value provided by a witness for a property of an object is independent of the values provided for the other properties of the same object, and that the accuracy of a witness for a property is independent of the property values (*independent properties assumptions*). Finally, for the sake of simplicity, we consider a uniform distribution (then $\mathcal{P}(X = x) = \frac{1}{n+1}, \forall x$), and we assume that the cardinality of the set of values provided by the witnesses is a good estimation for n (*uniform distribution assumption*).

Given an object, the larger is the number of witnesses that agree for the same value, the higher is the probability that the value is correct. However, the agreement of the witnesses' observations contributes in increasing the probability that a value is correct in a measure that depends also on the accuracy of the involved witnesses. The accuracy of a witness is evaluated by comparing its observations with the observations of other witnesses for a set of objects. A witness that frequently agrees with other witnesses is likely to be accurate.

Based on these ideas of mutual dependence between the analysis of the consensus among witnesses and the analysis of the witnesses accuracy, we have developed an algorithm [3] that computes the distribution probabilities for the properties of every observed object and the accuracies of the witnesses. Our algorithm takes as input the observations of some witnesses on multiple properties of a set of objects, and is composed of two main steps:

1. *Consensus Analysis*: based on the agreement of the witnesses among their observations on individual objects and on the current accuracy of witnesses, compute the probability distribution for the properties of every object (Section 3.1);
2. *Accuracy Analysis*: based on the current probability distributions of the observed object properties, evaluate the accuracy of the witnesses (Section 3.2).

The iterations are repeated until the accuracies of the witnesses do not significantly change anymore.

3.1 Probability Distribution of the Values

The following development refers to the computation of the probability distribution for the values of one property of an object, given the observations of several witnesses, and the accuracies of the witnesses with respect to that property. The same process can be applied for every object and property observed by the witnesses.

Given a set of witnesses w_1, \dots, w_k , with accuracy A_1, \dots, A_k that report a set of observations o_1, \dots, o_k our goal is to calculate: $P\left(\dot{x} \mid \bigcap_{i=1}^k o_i\right)$; i.e. we aim at computing the probability distribution of the values an object may assume, given the values reported by k witnesses.

First, we can express the desired probability using the Bayes' Theorem:

$$P\left(\dot{x} \mid \bigcap_{i=1}^k o_i\right) = \frac{P(\dot{x})P\left(\bigcap_{i=1}^k o_i \mid \dot{x}\right)}{P\left(\bigcap_{i=1}^k o_i\right)} \quad (1)$$

The events \dot{x}_j , with $j = 1 \dots n + 1$, form a partition of the event space. Thus, according to the Law of Total Probability:

$$P\left(\bigcap_{i=1}^k o_i\right) = \sum_{j=1}^{n+1} P(\dot{x}_j)P\left(\bigcap_{i=1}^k o_i \mid \dot{x}_j\right) \quad (2)$$

Assuming that the observations of all the witnesses are independent,¹ for each event \hat{x} we can write:

$$P\left(\bigcap_{i=1}^k o_i \mid \hat{x}\right) = \prod_{i=1}^k P(o_i \mid \hat{x}) \quad (3)$$

Therefore:

$$P\left(\hat{x} \mid \bigcap_{i=1}^k o_i\right) = \frac{P(\hat{x}) \prod_{i=1}^k P(o_i \mid \hat{x})}{\sum_{j=1}^{n+1} P(\hat{x}_j) \prod_{i=1}^k P(o_i \mid \hat{x}_j)} \quad (4)$$

$P(\hat{x})$ is the prior probability that X assumes the value x , then $P(\hat{x}) = \mathcal{P}(\hat{x}) = \frac{1}{n+1}$; $P(o_i \mid \hat{x})$ represents the probability distribution that the i -th witness reports a value $v(o_i)$. Observe that if $v(o_i) = x_t$ (i.e. the witness reports the correct value) the term coincides with the accuracy A_i of the witness. Otherwise, i.e. if $v(o_i) \neq x_t$, $P(o_i \mid \hat{x})$ corresponds to the probability that the witness reports an incorrect value. In this case, we assume that $v(o_i)$ has been selected randomly from the n incorrect values of X .

Since $P(o_i \mid \hat{x})$ is a probability distribution:

$$\sum_{v(o_i) \neq x_t} P(o_i \mid \hat{x}) = 1 - A_i.$$

Assuming that every incorrect value is selected according to the uniform prior probability distribution, we can conclude:

$$P(o_i \mid \hat{x}) = \begin{cases} A_i & , v(o_i) = x_t \\ \frac{1-A_i}{n} & , v(o_i) \neq x_t \end{cases}. \quad (5)$$

Combining (4) and (5), we obtain the final expression to compute $P\left(\hat{x} \mid \bigcap_{i=1}^k o_i\right)$.

3.2 Witnesses Accuracy

We now illustrate the evaluation of the accuracy of the witnesses with respect to one property, given their observations for that property on a set of objects, and the probability distributions associated with the values of each object computed as discussed in the previous section.

Our approach is based on the intuition that the accuracy of a witness can be evaluated by considering how its observations for a number of objects agree with those of other witnesses. Indeed, assuming that a number of sources independently report observations about the same property (e.g. trade value) of a shared set of objects (e.g. the NASDAQ stock quotes), these observations unlikely agree by chance. Therefore, the higher are the probabilities of the values reported by a witness, the higher is the accuracy of the witness.

¹ This assumption is a simplification of the domain that we will remove later by extending our model to deal with witnesses that may copy.

We previously defined the accuracy A_i of a witness w_i as the probability that w_i reports the correct value. Now, given the set of m objects for which the witness w_i reports its observations $o_i^j, j = 1 \dots m$, and the corresponding probability distributions $P_j(x | \bigcap_{q=1}^k o_q^j)$, computed from the observations of k witnesses with the equation (4), we estimate the accuracy of w_i as the average of the probabilities associated with the values reported by w_i :

$$A_i = \frac{1}{m} \sum_{j=1}^m P_j\left(X = v(o_i^j) \mid \bigcap_{q=1}^k o_q^j\right) \quad (6)$$

where $v(o_i^j)$ is the value of the observation reported by w_i for the object j .

Our algorithm [3] initializes the accuracy of the witnesses to a constant value, then it starts the iteration that computes the probability distribution for the value of every object (by using equations (4) and (5)) and the accuracy of witnesses (equation (6)).

4 Witnesses Dependencies over Many Properties

We now introduce an extension of the approach developed in [6] for the analysis of dependence among witnesses that removes the *independent witnesses assumption*. The kind of dependence that we study is due to the presence of copiers: they create “artificial” consensus which might lead to misleading conclusions.

As we consider witnesses that provide several properties for each object, we model the provided values by means of tuples. We assume that a copier either copies a whole tuple from another witness or it does not copy any property at all (*no-record-linkage assumption*). In other words, we assume that a copier is not able to compose one of its tuple by taking values (of distinct properties) from different sources. Otherwise, note that a record-linkage step would be needed to perform its operation, and it would be more appropriate to consider it as an integration task rather than a copying operation.

As in [6], we assume that the dependency between a pair of witnesses is independent of the dependency between any other pair of witnesses; the copiers may provide a copied tuple with a-priori probability $0 \leq c \leq 1$, and they may provide some tuples independently from other witnesses with a-priori probability $1 - c$ (*independent copying assumption*).

Under these assumptions, the evidence of copying could greatly improve by considering several properties, since it is much less likely that multiple values provided by two witnesses for the same object coincide by chance.

4.1 Modeling Witnesses Dependence

In order to deal with the presence of copiers, in the following we exploit the approach presented in [6] to modify the equations obtained with the independency assumption.

Let $W_o(x)$ be the set of witnesses providing the value x on an object and let W_o be the set of witnesses providing any value on the same object, the equation (3) can be rewritten as follows:

$$P\left(\bigcap_{i=1}^k o_i \mid \dot{x}\right) = \prod_{w \in W_o(x)} A_w \prod_{w \in W_o - W_o(x)} \frac{1 - A_w}{n} = \prod_{w \in W_o(x)} \frac{n \cdot A_w}{1 - A_w} \prod_{w \in W_o} \frac{1 - A_w}{n} \tag{7}$$

Among all the possible values x_1, \dots, x_{n+1} , assuming as before a uniform a-priori probability $\frac{1}{n+1}$ for each value, the equation (2) can be rewritten as follows:

$$P\left(\bigcap_{i=1}^k o_i\right) = \sum_{j=1}^{n+1} P\left(\bigcap_{i=1}^k o_i \mid \dot{x}_j\right) P(\dot{x}_j) = \frac{1}{n+1} \sum_{j=1}^{n+1} \prod_{w \in W_o(x_j)} \frac{n \cdot A_w}{1 - A_w} \prod_{w \in W_o} \frac{1 - A_w}{n}$$

The probability that a particular value is true given the observations (equation (4)), denoted $P(x)$, can be rewritten as follows:

$$P(x) = P\left(\dot{x} \mid \bigcap_{i=1}^k o_i\right) = \frac{P\left(\bigcap_{i=1}^k o_i \mid \dot{x}\right) \frac{1}{n+1}}{P\left(\bigcap_{i=1}^k o_i\right)} = \frac{\prod_{w \in W_o(x)} \frac{n \cdot A_w}{1 - A_w}}{\sum_{j=1}^{n+1} \prod_{w \in W_o(x_j)} \frac{n \cdot A_w}{1 - A_w}}$$

The denominator is a *normalization factor*, it is independent of $W_o(x)$ and it will be denoted ω to simplify the notation.

For taking into account the witnesses' dependency, it is convenient to rewrite $P(x) = \frac{e^{C(x)}}{\omega}$ where $C(x)$ is the *confidence* of x , which is basically the probability expressed according to a logarithmic scale:

$$C(x) = \ln P(x) + \ln \omega = \sum_{w \in W_o(x)} \ln \frac{n \cdot A_w}{1 - A_w}$$

If we define the *accuracy score* of a witness w as:

$$A'_w = \ln \frac{n \cdot A_w}{1 - A_w}$$

it arises that we can express the confidence of a value x as the sum of the accuracy scores of the witnesses that provide that value for independent witnesses:

$$C(x) = \sum_{w \in W_o(x)} A'_w$$

We can now drop the independent witnesses assumption; to take into account the presence of copiers the confidence is computed as the weighted sum of the accuracy scores:

$$C(x) = \sum_{w \in W_o(x)} A'_w I_w$$

where the weight I_w is a number between 0 and 1 that we call the *probability of independent opinion* of the witness w . It essentially represents which “portion” of the opinion of w is expressed independently of the other witnesses. For a perfect copier I_w equals to 0, whereas for a perfectly independent witness I_w equals to 1.

I_w can be expressed as the probability that a value provided by w is not copied by any other witness:

$$I_w = \prod_{w' \neq w} (1 - cP(w \rightarrow w'))$$

where $P(w \rightarrow w')$ is the probability that w is a copier of w' , and c is the a-priori probability that a copier actually copies the value provided.

Next, we will discuss how to compute a reasonable value of $P(w \rightarrow w')$ for a pair of witnesses.

4.2 Dealing with Many Properties

In [6] it is illustrated a technique to compute the probability $P(w_1 \rightarrow w_2)$ that w_1 is copier of w_2 , and the probability $P(w_1 \perp w_2)$ that w_1 is independent of w_2 starting from the observations of the values provided by the two witnesses for one given property.

Intuitively, the dependence between two witnesses w_1 and w_2 can be detected by analyzing for which objects they provide the same values, and the overall consensus on those values. Indeed, whenever two witnesses provide the same value for an object and the provided value is false, this is an evidence that the two witnesses are copying each other. Much less evidence arises when the two have a common true value for that object: those values could be shared just because both witnesses are accurate, as well as independent.

We consider three probabilities, $P(w_1 \perp w_2)$, $P(w_1 \rightarrow w_2)$, $P(w_2 \rightarrow w_1)$, corresponding to a partition of the space of events of the dependencies between two witnesses w_1 and w_2 : either they are dependent or they are independent; if they are dependent, either w_1 copies from w_2 or w_2 copies from w_1 . $P(w_1 \perp w_2 | \Phi) =$

$$\frac{P(\Phi | w_1 \perp w_2)P(w_1 \perp w_2)}{P(\Phi | w_1 \perp w_2)P(w_1 \perp w_2) + P(\Phi | w_1 \rightarrow w_2)P(w_1 \rightarrow w_2) + P(\Phi | w_2 \rightarrow w_1)P(w_2 \rightarrow w_1)}$$

Here Φ corresponds to $\bigcap_{i=1}^k o_i$, i.e. the observations of the values provided by the k witnesses, and namely, o_i corresponds to the observation of the tuples provided by the witness w_i on the object.

The a-priori knowledge of witnesses dependencies can be modeled by considering a parameter $0 < \alpha < 1$, and then setting the a-priori probability $P(w_1 \perp w_2)$ to α ; $P(w_1 \rightarrow w_2)$ and $P(w_2 \rightarrow w_1)$ are both set to $1 - \frac{\alpha}{2}$ [2]

² A similar discussion for $P(w_1 \rightarrow w_2 | \Phi)$, and $P(w_2 \rightarrow w_1 | \Phi)$ is omitted for space reasons.

The probabilities $P(\Phi|w_1 \perp w_2)$, $P(\Phi|w_1 \rightarrow w_2)$, $P(\Phi|w_2 \rightarrow w_1)$ can be computed with the help of the *independent values* assumption: the values independently provided by a witness on different objects are independent of each other.

For the sake of simplicity, here we detail how to compute, given the assumptions above, and considering our generative model of witnesses, $P(\Phi|w_1 \perp w_2)$, i.e. the probability that two independent witnesses w_1 and w_2 provide a certain observation Φ in the case of two properties denoted A and B for which they respectively exhibit error rates³ of ϵ_1^A , ϵ_1^B , ϵ_2^A , ϵ_2^B . A similar development would be possible in the case of witnesses providing more than two properties.

Given the set of objects O for which both w_1 and w_2 provide values for properties A and B , it is convenient to partition O in these subsets: $O_{tt} \cup O_{tf} \cup O_{ft} \cup O_{ff} \cup O_d = O$. For objects in $O_{tt} \cup O_{tf} \cup O_{ft} \cup O_{ff}$, w_1 and w_2 provide the same values of properties A and B , whereas for objects in O_d the two witnesses provide different values for at least one property. In the case of objects in O_{tt} , the witnesses agree on the true value for both properties; for objects in O_{tf} they agree on the true value of A and on the same false value of B ; similarly for O_{ft} they agree on the same false value of A and on the true value of B ; finally, in the case of O_{ff} they agree on the same false values for both properties.

We first consider the case of both witnesses independently providing the same values of A and B and these values are either both true or both false. According to the *independent properties* assumption, w_i provides the pair of true values for A and B with probability $(1 - \epsilon_i^A)(1 - \epsilon_i^B)$, and a particular pair of false values with probability $\frac{\epsilon_i^A}{n_A} \frac{\epsilon_i^B}{n_B}$, with n_A (respectively n_B) being the number of possible false values for the property A (resp. B). Given that the witnesses are independent, and there are $n_A \cdot n_B$ possible pairs of false values on which the two witnesses may agree, we can write:

$$\begin{aligned} P(o \in O_{tt}|w_1 \perp w_2) &= (1 - \epsilon_1^A)(1 - \epsilon_2^A)(1 - \epsilon_1^B)(1 - \epsilon_2^B) = P_{tt} \\ P(o \in O_{ff}|w_1 \perp w_2) &= \frac{\epsilon_1^A \epsilon_2^A}{n_A} \frac{\epsilon_1^B \epsilon_2^B}{n_B} = P_{ff} \end{aligned}$$

A witness w_i independently provides a true value of A and a particular false value for B with probability $(1 - \epsilon_i^A) \frac{\epsilon_i^B}{n_B}$ (similarly for $P(o \in O_{ft}|w_1 \perp w_2)$):

$$\begin{aligned} P(o \in O_{tf}|w_1 \perp w_2) &= (1 - \epsilon_1^A)(1 - \epsilon_2^A) \frac{\epsilon_1^B \epsilon_2^B}{n_B} = P_{tf} \\ P(o \in O_{ft}|w_1 \perp w_2) &= (1 - \epsilon_1^B)(1 - \epsilon_2^B) \frac{\epsilon_1^A \epsilon_2^A}{n_A} = P_{ft} \end{aligned}$$

All the remaining cases are in O_d :

$$P(o \in O_d|w_1 \perp w_2) = 1 - P_{tt} - P_{tf} - P_{ft} - P_{ff} = P_d$$

³ The error rate ϵ of a witness with respect to a property is the complement at 1 of its accuracy A with respect to the same property: $\epsilon = 1 - A$.

The *independent values assumption* allows us to obtain $P(\Phi|w_1 \perp w_2)$ by multiplying the probabilities and appropriately considering the cardinalities of the corresponding subsets of O :

$$P(\Phi|w_1 \perp w_2) = P_{tt}^{|O_{tt}|} \cdot P_{tf}^{|O_{tf}|} \cdot P_{ft}^{|O_{ft}|} \cdot P_{ff}^{|O_{ff}|} \cdot P_d^{|O_d|}.$$

Now we detail how to compute $P(\Phi|w_1 \rightarrow w_2)$, but we omit $P(\Phi|w_2 \rightarrow w_1)$ since it can be obtained similarly. Recall that, according to our model of copier witnesses, a copier with a-priori probability $1 - c$ provides a tuple independently. In this case, we can reuse the probabilities $P_{tt}, P_{ff}, P_{tf}, P_{ft}, P_d$ obtained above for independent witnesses with weight $1 - c$. However, with a-priori probability c , a copier witness w_1 provides a tuple copied from the witness w_2 and hence generated according to the same probability distribution function of w_2 . For instance, w_2 would generate a pair of true values with probability $(1 - \epsilon_2^A)(1 - \epsilon_2^B)$. Concluding:

$$\begin{aligned} P(o \in O_{tt}|w_1 \rightarrow w_2) &= (1 - \epsilon_2^A)(1 - \epsilon_2^B)c + P_{tt}(1 - c) \\ P(o \in O_{ff}|w_1 \rightarrow w_2) &= \epsilon_2^A \epsilon_2^B c + P_{ff}(1 - c) \\ P(o \in O_{tf}|w_1 \rightarrow w_2) &= (1 - \epsilon_2^A)\epsilon_2^B c + P_{tf}(1 - c) \\ P(o \in O_{ft}|w_1 \rightarrow w_2) &= (1 - \epsilon_2^B)\epsilon_2^A c + P_{ft}(1 - c) \end{aligned}$$

For the remaining cases, we have to consider that since the witnesses are providing different values for the same object, it cannot be the case that one is copying the other.

$$P(o \in O_d|w_1 \rightarrow w_2) = (1 - P_{tt} - P_{tf} - P_{ft} - P_{ff})(1 - c)$$

Again, the *independent values assumption* allows us to obtain $P(\Phi|w_1 \rightarrow w_2)$ by multiplying these probabilities raised to the cardinality of the corresponding subset of O .

5 Experiments

We now describe the settings and the data we used for the experimental evaluation of the proposed approach. We conducted two sets of experiments. The first set of experiments was done with generated synthetic data, while the second set was performed with real world data extracted from the Web.

For the following experiments we set $\alpha=0.2$ and $c=0.8$.

5.1 Synthetic Scenarios

The goal of the experiments with synthetic data was to analyze how the algorithms perform with sources of different quality.

We conducted two sets of experiments EXP1 and EXP2 to study the performances of the approach with different configurations as summarized in Figure 2. In the two sets there are three possible types of sources: *authorities*, which provide true values for every object and every attribute; *independents*, which make

| | #authorities | #independents | #copiers | \bar{A} |
|------|--------------|---------------|----------|-----------|
| EXP1 | 0 | 8 | 10 | 0.1 - 0.9 |
| EXP2 | 1 | 7 | 10 | 0.1 - 0.9 |

Fig. 2. Configurations for the synthetic scenarios

mistakes according to the source accuracy \bar{A} ; *copiers*, which copy according to a copying rate r from the independents, and make mistakes according to the source accuracy \bar{A} when they report values independently. The experiments aim at studying the influence of the varying source accuracies and the presence of an authority source (notice that the authority is not given: the goal of the experiments is to detect it).

In all the experiments we generated sources with $N = 100$ objects, each described by a tuple with 5 attributes with values for all the objects; the copiers copy from an independent source with a frequency $r = 0.8$. In all the scenarios each copier copies from three independents, with the following probabilities: 0.3, 0.3, 0.4.

In order to evaluate the influence of complex data, for each of these configurations we varied the number of attributes given as input to the algorithm with three combinations: 1, 3, and 5 attributes. We remark that our implementation coincides with the current state of the art when only one attribute is considered [6]. To highlight the effectiveness of our algorithm, we also compared our solution with a *naive approach*, in which the probability distribution is computed with a simple voting strategy, ignoring the accuracy of the sources.

To evaluate the performance of the algorithms we report the *Precision* (P), i.e. the fraction of objects on which we select the true values, considering as candidate true values the ones with the highest probability.

Results. The results of our experiments on the synthetic scenarios are illustrated in Figure 3. For each set of experiments we randomly generated the datasets and applied the algorithms 100 times. We report a graphic with the average Precision for the naive execution and for the three different executions of our approach. We used in fact executions of MultiAtt(1) with only one attribute given as input, executions of MultiAtt(3) with three attributes, and executions of MultiAtt(5) with five.

From the two sets it is apparent that the executions with multiple attributes always outperform the naive execution and the one considering only one attribute. In the first set EXP1, MultiAtt(3) and MultiAtt(5) present some benefits compared to previous solutions, but are not able to obtain excellent precision in presence of high error rates. This is not surprising: even if MultiAtt(3) and MultiAtt(5) are able to identify perfectly what sources are copiers, there are 8 independent sources reporting true values with a very low frequency and for most of the objects the evidence needed to compute the true values is missing. The scenario radically changes in EXP2, where an authority exists and MultiAtt(5) is able to return all the correct values even for the worst case, while MultiAtt(3) and MultiAtt(1) start significantly mixing dependencies at 0.8 and 0.5 error rates, respectively.

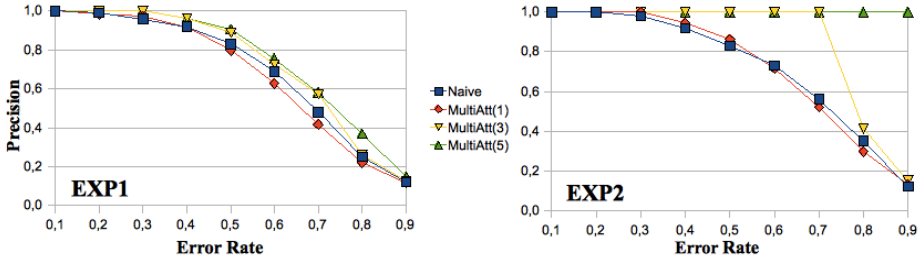


Fig. 3. Synthetic experiments: MultiAtt(5) outperforms alternative configurations in all scenarios

It is worth remarking that our algorithm does not introduce regressions with respect to previous solutions. In fact, we have been able to run all the synthetic examples in [6] obtaining the same results with all the configurations of MultiAtt. This can be explained by observing that in those examples the number of copiers is minor than the number of independent sources and MultiAtt(1) suffices for computing correctly all the dependencies. In the following, we will show that real data are significantly affected by the presence of copiers, but there are cases where considering only one attribute does not suffice to find the correct dependencies between sources.

5.2 Real-World Web Data

We used collections of data extracted from web sites about NASDAQ stock quotes.⁴ All the extraction rules were checked manually, and the pages were downloaded on November 19th 2009.⁵

The settings for the real-world experiments are reported in Figure 4, which shows the list of attributes we studied. Among hundreds of available stock quotes we have chosen the subset that maximizes the inconsistency between sources.

It is worth observing that in this domain an authority exists: it is the official NASDAQ website (<http://www.nasdaq.com>). We ran our algorithm over the available data and we evaluated the results considering the data published by that source as the truth. The experiments were executed on a FreeBSD machine with Intel Core Duo 2.16GHz CPU and 2GB memory.

To test the effectiveness of our approach we executed the algorithm considering one attribute at a time, considering all the 10 possible configurations of three attributes, and, finally, considering five attributes at the same time. In Figure 5, are reported the average of the precisions obtained over the five attributes by these configurations. The worst average precision (0.39) is obtained considering only one attribute at a time: this is due to the lack of clear majorities in the

⁴ We relied on *Flint*, a system for the automatic extraction of web data [2].

⁵ Since financial data change during the trading sessions, we downloaded the pages while the markets were closed.

| Attribute | #sites | %null | #symbols | #objects |
|--------------|--------|-------|----------|----------|
| last price | 39 | 0.3 | 544 | 250 |
| open price | 34 | 16.09 | 568 | 250 |
| 52 week high | 34 | 16.59 | 531 | 250 |
| 52 week low | 34 | 16.59 | 487 | 250 |
| volume | 39 | 1.98 | 1259 | 250 |

Fig. 4. Settings for the real-world experiments

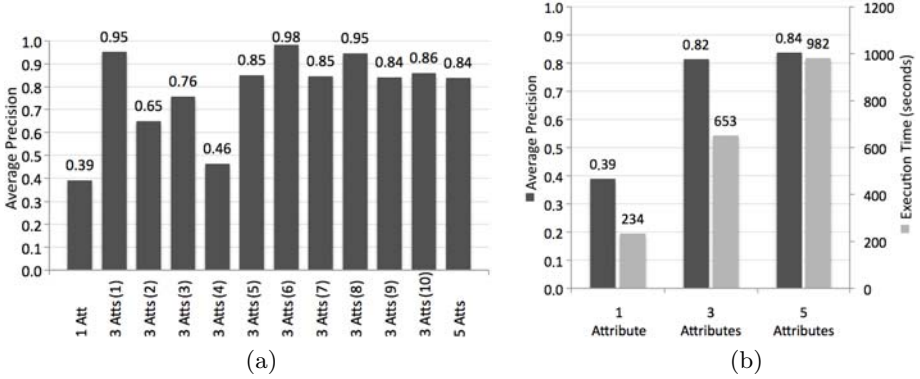


Fig. 5. Real-world experiments results

setting and the consequent difficulty in the discovery of the dependencies. We obtained interesting results considering the configurations of three attributes. In fact, it turned out that some configurations perform significantly better than others. This is not surprising, since the quality of the data exposed by an attribute can be more or less useful in the computation of the dependencies: for example, an attribute does not provide information to identify copiers if either all the sources provide the correct values or all the sources provide different values. However, it is encouraging to notice that considering all the five attributes we obtained a good precision (0.84). This shows that even if there exist attributes that do not contribute positively (or provide misleading information), their impact can be absorbed if they are considered together with the good ones.

Figure 5b reports the average precision scores for the three configurations compared with their execution times (the average in the cases with one and three attributes). It can be observed that the execution times increase linearly with the number of attributes involved in the computation, with a maximum of 16 minutes for the configuration with five attributes.

6 Conclusions and Future Work

We developed an extension of existing solutions for reconciling conflicting data from inaccurate sources. Our extension takes into account complex data, i.e.

tuples, instead of atomic values. Our work shows that the dependence analysis is the point in which the state-of-the-art models can be extended to analyze several properties at a time. Experiments showed that our extension can greatly affect the overall results.

We are currently studying further developments of the model. First, we are investigating an extension of the model beyond the uniform distribution assumption. Second, we are studying more complex forms of dependencies such as those implied by integration processes that include a record linkage step.

References

1. Berti-Equille, L., Sarma, A.D., Dong, X., Marian, A., Srivastava, D.: Sailing the information ocean with awareness of currents: Discovery and application of source dependence. In: CIDR (2009)
2. Blanco, L., Crescenzi, V., Merialdo, P., Papotti, P.: Flint: Google-basing the web. In: EDBT (2008)
3. Blanco, L., Crescenzi, V., Merialdo, P., Papotti, P.: A probabilistic model to characterize the uncertainty of web data integration: What sources have the good data? Technical report, DIA - Roma Tre - TR146 (June 2009)
4. Cafarella, M.J., Etzioni, O., Suci, D.: Structured queries over web text. *IEEE Data Eng. Bull.* 29(4), 45–51 (2006)
5. Dalvi, N.N., Suci, D.: Management of probabilistic data: foundations and challenges. In: PODS, pp. 1–12 (2007)
6. Dong, X.L., Berti-Equille, L., Srivastava, D.: Integrating conflicting data: The role of source dependence. *PVLDB* 2(1), 550–561 (2009)
7. Dong, X.L., Berti-Equille, L., Srivastava, D.: Truth discovery and copying detection in a dynamic world. *PVLDB* 2(1), 562–573 (2009)
8. Downey, D., Etzioni, O., Soderland, S.: A probabilistic model of redundancy in information extraction. In: IJCAI, pp. 1034–1041 (2005)
9. Florescu, D., Koller, D., Levy, A.Y.: Using probabilistic information in data integration. In: VLDB, pp. 216–225 (1997)
10. Galland, A., Abiteboul, S., Marian, A., Senellart, P.: Corroborating information from disagreeing views. In: Proc. WSDM, New York, USA (2010)
11. Wu, M., Marian, A.: Corroborating answers from multiple web sources. In: WebDB (2007)
12. Yin, X., Han, J., Yu, P.S.: Truth discovery with multiple conflicting information providers on the web. *IEEE Trans. Knowl. Data Eng.* 20(6), 796–808 (2008)

Monitoring and Analyzing Service-Based Internet Systems through a Model-Aware Service Environment

Ta'id Holmes¹, Uwe Zdun¹, Florian Daniel², and Schahram Dustdar¹

¹ Distributed Systems Group, Institute of Information Systems
Vienna University of Technology, Vienna, Austria
{tholmes, zdun, dustdar}@infosys.tuwien.ac.at

² Information Engineering and Computer Science Department
University of Trento, Trento, Italy
daniel@disi.unitn.it

Abstract. As service-based Internet systems get increasingly complex they become harder to manage at design time as well as at runtime. Nowadays, many systems are described in terms of precisely specified models, e.g., in the context of model-driven development. By making the information in these models accessible at runtime, we provide better means for analyzing and monitoring the service-based systems. We propose a model-aware repository and service environment (MORSE) to support model access and evolution at both design time and runtime. MORSE focuses on enabling us to monitor, interpret, and analyze the monitored information. In an industrial case study, we demonstrate how compliance monitoring can benefit from MORSE to monitor violations at runtime and how MORSE can ease the root cause analysis of such violations. Performance and scalability evaluations show the applicability of our approach for the intended use cases and that models can be retrieved during execution at low cost.

1 Introduction

In the Internet of services, systems get increasingly complex. Often it is hard to manage, analyze, and monitor such complex service-based systems at runtime. This is, (1) the complexity of service-based systems needs to be mastered and (2) requirements that are imposed on such systems have to be monitored.

For various reasons, many systems (cf. [12]) and requirements (cf. [34]) today are modeled with precisely specified and detailed models. One of these reasons is the increasing use of model-driven development (MDD) [5] that helps to master the complexity of systems during development. Management, analysis, and monitoring results could be improved by making the information in the precisely specified models accessible at runtime. We propose a repository-based approach, in which the MDD models of a service-based system and its system requirements can be used at runtime to interactively interpret or analyze the monitored information.

As an illustrative case for system requirements, consider compliance to regulations: A business information system needs to comply with regulations, such as Basel II [6] or the Sarbanes-Oxley Act (SOX) [7]. Runtime monitoring of service-based business processes can be used to detect violations of such regulations at execution time. If a

violation is detected, a report can be generated and a root cause analysis started. In order to trace back from process instances that have caused a violation to the model elements that have driven the execution of those instances, information described in models of the system needs to be queried.

In addition, service-based systems not only require read access to the models, but also write access. In the compliance management case, for example, once a root cause analysis indicates a problem in a model, the developer should be able to (1) open the respective model for modification, (2) perform the modifications on-the-fly, and (3) add the new model version to the running system so that newly created model instances can immediately use the corrected, evolved version.

In this paper, we propose to address these issues using an architecture and software environment called the Model-Aware Repository and Service Environment (MORSE) [8]. MORSE supports the tasks of the MDD life cycle by managing MDD artifacts, such as models, model instances, and transformation templates. Models are first-class citizens in MORSE, and they can be queried, changed, and updated both at design time and runtime. That is, the models in the repository can be used by the generated service-based system (e.g., the compliance monitoring infrastructure) via Web services to query and change the models while the system runs. Thanks to its generic, service-based interfaces, MORSE can be integrated in various monitoring and analysis infrastructures. We have evaluated our prototype using exhaustive performance and scalability tests to validate the applicability of the approach in practice.

This paper is structured as follows: In the next section we will give a motivating example and illustrate the monitoring architecture employed for our approach. In Section 3 we present MORSE the Model-Aware Repository and Service Environment and describe how Internet-based systems can be enhanced with traceability information for monitoring and analyzing these systems. Next, in Section 4 we illustrate our work with a case study. Performance and Scalability evaluations are then presented in Section 5. Section 6 compares our approach to related work, and in Section 7 we conclude and refer to future work.

2 A Model-Aware Service Environment in Compliance Monitoring and Analysis

First we outline a concrete example motivating the need for a model-aware repository and service environment for monitoring and analysis purposes. We consider the problem of monitoring and analyzing business processes to identify compliance violations. In particular, we focus on a monitoring infrastructure that observes a business IT system while it is running business processes modeled, for instance, as BPEL [9] processes. Using a dedicated model for compliance concerns, the system knows about how to identify violations. A simple example of a compliance rule is the four eyes principle, which requires that a single operation in a business process must be performed by two different actors. If the monitoring infrastructure observes a violation of a compliance concern, it reports it to the user via a compliance dashboard.

Upon the detection of a violation, the user typically wants to understand which process has caused the violation and why. Hence, the user should access the process model

that has caused the violation. However, sometimes the process model is not the only relevant information or not the root cause of the violation. Other models linked to the process model might carry the answer to the violation; hence, they must be accessed, too. Examples are the model specifying the compliance rule that has been violated or the models of the processes that have invoked the process leading to the violation.

Finally, once the root cause has been identified, it is likely that the user will fix the affected models. Then, the corrected models should from then on be used for new instances of the business processes. Here, a transparent versioning support is crucial, as it is typically not possible to automatically migrate running process instances from one model to another. Old model versions must be supported as long as instances of them are running, unless the execution of such instances is halted.

To enable using models at runtime as described in this scenario, we propose the following runtime features:

Model-aware repository and service environment. The repository contains all models of the system. These can be queried and changed via service-based interfaces at runtime (see also Figure 2).

Eventing infrastructure. In our approach, the code setting up the emission of events is generated via MDD from process and compliance models. That is, the process engine is instrumented to emit events, such as *process started*, *process ended*, or *process activity entered*. Each of these events includes a unique identifier to the model artifact, such as *process model P* or *activity A in the process model P*, that is the source of the event.

Monitoring infrastructure. A monitoring infrastructure is required to continuously observe and analyze emitted events to check compliance. Runtime requirements-monitoring systems, such as proposed by Feather et al. [10] and Skene and Emerich [3], can be integrated with MORSE and used for the compliance checking. If a violation occurs, the model responsible for the violation can easily be identified using the identifier contained in the event(s) that caused the violation.

In the MORSE repository, the various models are linked via relationships. Hence, the root cause of the violation can be determined by traversing these relationships. For instance, a process model can be annotated in different models exogenously with compliance concerns, and the monitoring infrastructure needs to resolve what concern has been violated. To do so, another service-based request is sent to the repository for retrieving the compliance concerns that annotate the source model. In a number of interactive steps, the root cause of a violation can be detected.

Figure 1 shows the application of our approach for our illustrative case study in an event-driven architecture [11] for compliance monitoring combining both online monitoring (on-the-fly analysis of events at runtime) and offline monitoring (analysis of audit trails and event logs) in a service-oriented environment. Online monitoring and on-the-fly event analysis is necessary to react as quickly as possible to violations. However, not all violations of compliance in service-based Internet systems can be determined online (consider, e.g., long-running processes). Such violations typically require offline analysis such as an investigation of the event history.

Figure 1 shows a representative example of a monitoring infrastructure combining the two monitoring approaches with MORSE: The execution of services and business

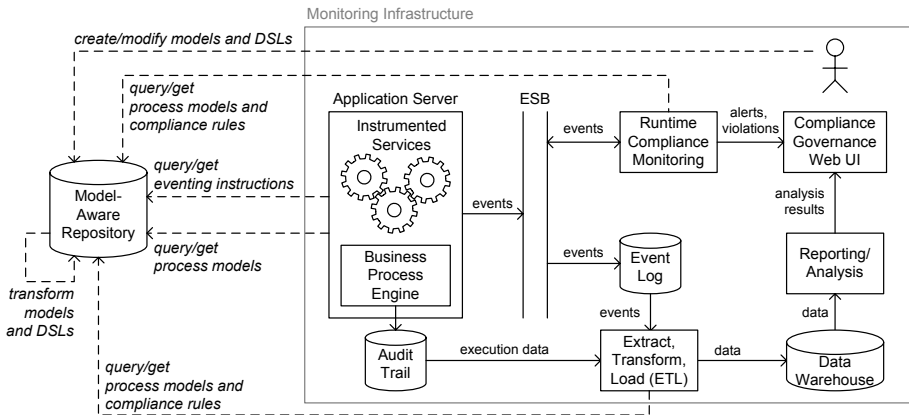


Fig. 1. MORSE Combined with an Online and Offline Monitoring Infrastructure

processes emits events, which provide information about the concrete execution of services and process instances. Events are, therefore, emitted by the business process engine and by the instrumented services and published via a central enterprise service bus (ESB). The ESB provides publish/subscribe support for events consumed by the monitoring components. In the proposed architecture, the event log component is subscribed to any event that is relevant for monitoring and further processing. Additional process execution data (e.g., data that is not carried with events, such as engine-internal events) are stored in a dedicated audit trail.

Starting from the event log and the audit trail, the analysis components such as the data warehouse (that stores all execution data), the extract/transform/load (ETL) procedures (that are responsible for extracting the data needed for the compliance analyses from the event log and the audit trail, transform them into the data format of the data warehouse, and load them into the warehouse), and the compliance governance Web user interface (UI), which includes a compliance governance dashboard for human users.

Processes and compliance concern models are stored in the MORSE repository and deployed onto the compliance monitoring infrastructure for execution, monitoring, and analysis. Specifically, the process engine and instrumented services query the model-aware repository for process models and eventing instructions. Eventing instructions are automatically generated from the compliance concerns in the repository. The runtime compliance monitoring component and the ETL procedures query the repository for process models and compliance concern models for compliance evaluation. Finally, the user of the compliance governance UI creates and modifies models and compliance concerns on-the-fly, directly working on the MORSE repository.

3 Model-Aware Repository and Service Environment

The integration of MORSE into the compliance governance infrastructure with its on-line and offline monitoring features poses a variety of requirements to its model management capabilities. As most stringent we specifically highlight the following:

- It is necessary to *store, deploy, maintain* and *evolve* process and service models as well as compliance annotations (expressed via dedicated DSLs) in a way that allows one to keep consistent *relationships* among these artifacts, which are typically subject to change over time. For instance, it is necessary to be able to associate a set of compliance rules to a process model and to correctly maintain such association even after the modification of the process model or the addition, modification or deletion of individual compliance rules.
- For the online analysis of compliance violations, it is necessary to be able to *drill down* from high-level process models down to individual activities or event annotations, associated with the process model. It is therefore necessary that the repository is aware of the hierarchical composition of process definitions and that each element in the model can be properly indexed. Given a violation of a compliance rule in a process instance, it might, for example, be necessary to retrieve the process definition and to drill down to the specific activity that caused the violation, in order to understand how to mitigate future violations.
- Finally, given its use at runtime of the monitoring infrastructure, *fast response times* and *high query throughput* are paramount. Individual compliance rules associated with a given process model might, for instance, be queried at each instantiation of the process, in order to set up the runtime compliance monitoring module (based on complex event processing [11]).

MORSE supports the above compliance governance scenario (and similar cases) in that it suitably addresses these requirements, going beyond simple file system based model management. In particular, the main features of MORSE that simplify the development of the monitoring infrastructure include:

- The MORSE repository *stores* and *manages* models, model elements, model instances, and other MDD artifacts. It offers read and write access to all artifacts at runtime and design time. Moreover, it stores relationships among the MDD artifacts, e.g., model-relationships such as instance, inheritance, and annotation relations (for details see also Figure 5 in [8]).
- *Information retrieval* (i.e., the querying of models) is supported for all MDD artifacts and relationships via service-based interfaces, which ease the integration of MORSE into service-oriented environments (for details see also Table I in [8]). For reflectively exploiting the relationships of MDD artifacts, the MORSE information retrieval interface provides various methods, too. An example of such a reflective relationship access is to retrieve all model instances for a model. By traversing the relationships and exploiting properties, complex queries can be constructed in an interactive, stepwise manner.
- The MORSE repository provides *versioning* capabilities not only to the artifacts, but also to their relationships. This way, models can be manipulated at runtime of the client system with minimal problems regarding maintenance and consistency. New versions and old versions of the models can be maintained in parallel, so that old model versions can be used until all their model instances are either deleted or migrated to the new model version.

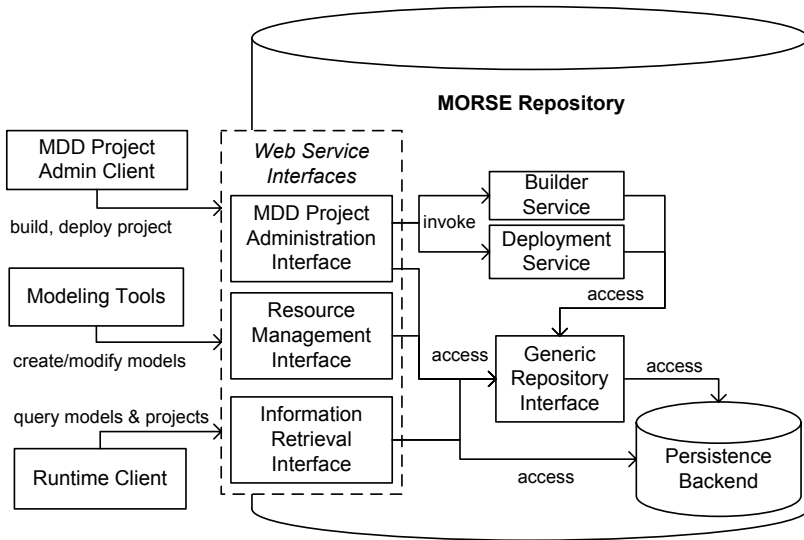


Fig. 2. MORSE Architecture

Figure 2 shows the internal architecture of MORSE that can be used by various clients via its Web service interfaces. The models are created by human modelers using modeling tools. Using admin clients, MDD projects in the MORSE repository can be created. Import clients allow the modelers to add models, model elements, and model relationships, or to provide them in a new version.

The model repository is the main component of MORSE and has been designed with the goal to abstract from specific technologies. Thus, while concepts are taken from, e.g., UML [12], and also version control systems such as Subversion [13], MORSE is particularly agnostic to specific modeling frameworks or technologies. The model repository itself has been realized with technologies such as Eclipse Modeling Framework (EMF) [14], Teneo [15], EclipseLink [16], PostgreSQL [17], and Apache CXF [18].

```

CSource source = Singletons.FACTORY.createCSource();
source.setDescription("SOX Sec.409");
CRisk risk = Singletons.FACTORY.createCRisk();
risk.setDescription("Penalty");
risk.setImpact(EnumWeight.HIGH);
risk.setProbability(EnumWeight.LOW);
CRequirement requirement = Singletons.FACTORY.createCRequirement();
requirement.setDescription("Rapid publication of Form 8-K");
CRequirementService requirementService = new CRequirementWSPProxy(requirement);
requirementService.create();
requirementService.addSources(source);
requirementService.addRisks(risk);

```

Listing 1. Populating MORSE with Instances of Compliance Domain Concepts

MORSE can also generate Web services for domain concepts that are expressed in EMF models. This is, for every concept a service is generated with basic *create*, *retrieve*, *update*, *delete* and *query* operations. For the different relations between concepts appropriate *add* and *remove* operations are generated in addition for associating and deassociating role instances.

Besides these service implementations, MORSE provides developers with Java libraries that are distributed using a Maven [19] repository. Listing 2 shows an example from a script that populates MORSE with instances of compliance domain concepts using the generated libraries. In the example a compliance source that relates to SOX Section 409 as well as a compliance risk are associated with a compliance requirement.

3.1 The MORSE Approach for Service-Based Internet Systems

Our approach applies MDD to generate services and process code, deployment artifacts, and monitoring directives. Usually, in MDD there are no backward or traceability links

```

<process name="ReportIntrusion">
  <extensions>
    <extension mustUnderstand="yes"
      namespace="http://xml.vitalab.tuwien.ac.at/ns/morse/traceability.xsd"
    />
  </extensions>
  <import importType="http://www.w3.org/2001/XMLSchema"
    namespace="http://xml.vitalab.tuwien.ac.at/ns/morse/traceability.xsd"
    location="http://xml.vitalab.tuwien.ac.at/ns/morse/traceability.xsd"
  />
  <morse:traceability
    build="56810150-5bd8-4e8e-9ec5-0b88a205946b">
    <row query="/process[1]"
      queryLanguage="urn:oasis:names:tc:wsbpel:2.0:sublang:xpath1.0">
      <uuid>6338b114-3790-4566-a5c4-a35aa4efe41b</uuid>
      <uuid>cd2865e2-73a7-4c8d-8235-974057a40228</uuid>
      <uuid>4bcf3d70-9c23-4713-8602-3b64160c45e8</uuid>
      <uuid>c568c290-e03e-46c8-9a9a-d7afde80cc3a</uuid>
    </row>
    <row query="/process[1]/sequence[1]/receive[1]">
      <uuid>354b5161-dfab-44ef-9d52-3fb6a9d3411d</uuid>
    </row>
    <row query="/process[1]/sequence[1]/invoke[3]">
      <uuid>7d32b4f4-4f63-4223-8860-db213f7e0fe1</uuid>
    </row>
  </morse:traceability>
  <sequence>
    <!-- ... //-->
  </sequence>
</process>

```

Listing 2. BPEL Process with an Extension for MORSE Traceability

in the sense that the generated source code “knows” from which model it has been created. For correlating model instances or code at runtime with source models or model instances, respectively, traceability of model transformations is essential.

To achieve traceability, models (as the output of the generator) can hold a reference to their source models. As MDD artifacts in MORSE repositories are identifiable by Universally Unique Identifiers (UUIDs) [20], MORSE annotates the destination models with the UUIDs of the models. The generator can automatically weave references to these UUIDs into the generated source code or configuration instructions, so that the corresponding models can be identified and accessed from the running system. Please note, that UUIDs are transparently created and used in the MORSE repository (e.g., the *create* operation returns the assigned UUID). Particularly, a user or developer (cf. Listing 1) does not necessarily have to specify UUIDs for interacting with MORSE.

The code in Listing 2 shows a generated BPEL process that contains traceability information as a BPEL extension. The executing BPEL engine emits events for, e.g., the process activities that contain matching UUIDs. Finally, the events are processed by the monitoring infrastructure.

4 Case Study: Compliance to Regulations

Let us consider an industrial case study in which MORSE has been applied: A US credit card company that wants to comply with Section 409 (Real Time Issuer Disclosures) of SOX [7]. Section 409 requires that a publicly traded company discloses information regarding material changes in the financial condition of the company in real-time (usually meaning “within two to four business days”), see also Figure 3. Changes in the financial condition of a company that demand for disclosure are, for example, bad debts, loss of production capacity, changes in credit ratings for the company or large clients, mergers, acquisitions, or major discoveries.

For the case of the credit card company, we consider the following reference scenario regarding the change in the financial condition: security breaches are detected in the company’s IT system, where personal information about customers might get stolen or lost. The business process in the lower part of Figure 3 shows a possible practice that the company may internally follow to react to a detected intrusion. After an initial assessment of the severity of the intrusion, the company immediately starts the necessary response action to mitigate risks and prevent similar future intrusions. After the response, if personal information got lost or stolen, the disclosure procedure is started. As detailed in the process, the actual disclosure is performed by filing a so-called Form 8-K report, a specific form used to notify investors and the U.S. Securities and Exchange Commission (who is in charge of controlling the compliance with SOX).

Note that full compliance with Section 409, of course, requires that all business practices in the company are compliant; the case of stolen or lost personal information represents only one out of multiple business practices in the credit card company that are subject to Section 409 of SOX.

The sole implementation of the compliant business process does not yet guarantee compliance: failures during process execution may happen (e.g., due to human errors,

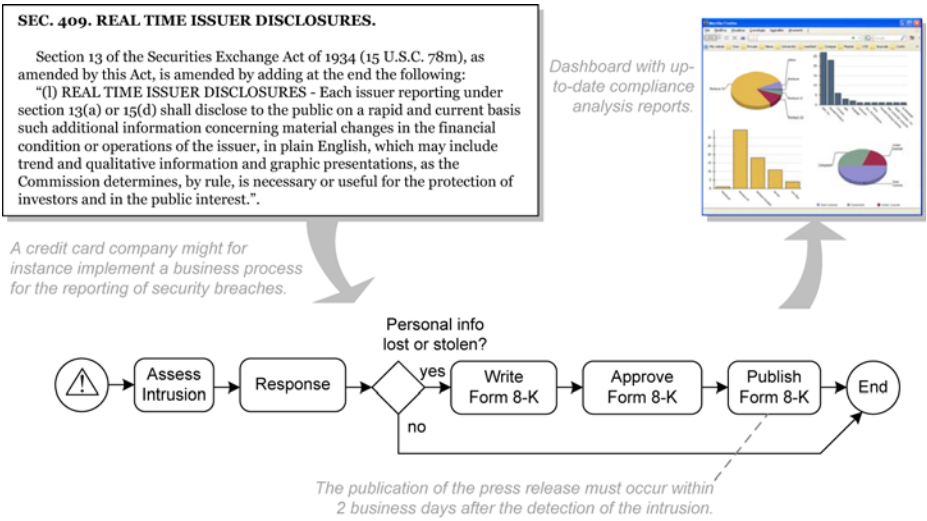


Fig. 3. SOX Example

system failures, or the like), and the preparation and publication of the Form 8-K might be delayed, erroneous, or even forgotten. Awareness of such problems is of utmost importance to the company, in order to be able to react timely and, hence, to assure business continuity. In this regard, the ability to perform root cause analyses to understand the reason for specific compliance violations is needed.

We assume that the monitoring infrastructure in Figure 1 is used throughout the architecture and that MDD is used to make sure, all models are placed in the MORSE repository, and the UUIDs are used in all events emitted in the system. The MORSE repository can be used to support creating reports and running root cause analyses by making all MDD artifacts accessible at runtime. Once the cause of a violation has been understood, the developers of the company should be able to redesign the MDD artifacts (e.g., the business processes) to avoid similar violations in the future.

Figure 4 illustrates the interplay of MORSE, the monitoring infrastructure, and the compliance governance Web UI when dealing with compliance violations. All models are placed in the MORSE repository. A model of a business process is annotated by compliance models. They relate to a certain regulation or specify details for an implementation. In our example, a `ComplianceConcern` annotates the process and a `PublishDeadline` annotates the process activity `Publish Form 8-K`. These annotation models will be retrieved and evaluated by the monitoring infrastructure for detecting violations during runtime. From MORSE, the business process is automatically deployed on a business process engine (1). The business process engine emits various events such as when a process is initialized or when an activity is invoked or completed (2). These events contain the corresponding UUIDs and are intercepted by the

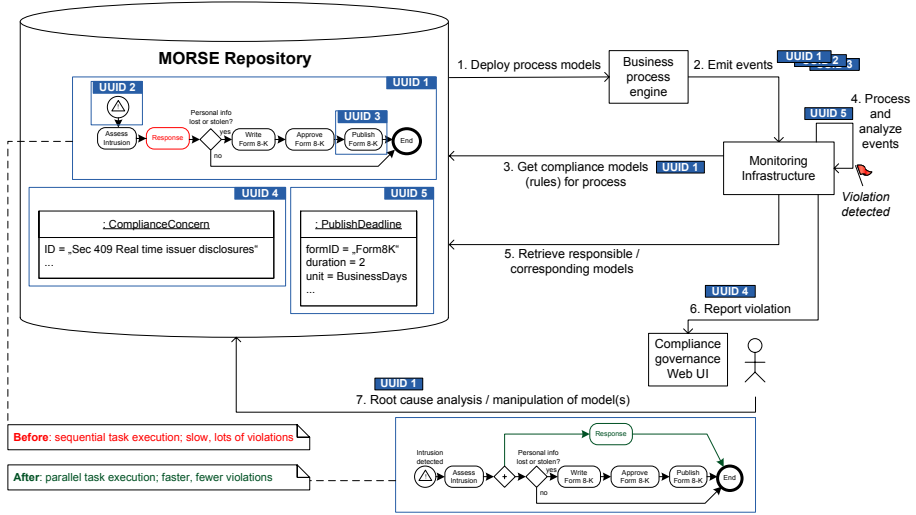


Fig. 4. Resolved SOX Example

monitoring infrastructure, which requests the compliance rules related to the intercepted events from MORSE (3).

Validation then takes place in online or offline operation mode (4). In case of a violation (i.e., Form 8-K has not been published within two business days according to the `PublishDeadline`), it is signaled in the dashboard. To present meaningful information to the user of the dashboard, the models responsible for the violation are retrieved from the MORSE repository and shown to the user (5). That is, the monitoring infrastructure first requests the original MDD artifact by UUID and then explores its relationships. Particularly, a compliance concern model instance that relates to the process or process activity can be identified and displayed for adequate feedback towards the user (6).

The user can now analyze the violation by traversing the models and/or performing additional queries. That is, the dashboard or the user consults the repository for resolving and identifying the root cause of the violation. In our example, the root cause lies in the sequential structure of the control flow of the process. The user can now improve the responsible model so that new processes may not violate the compliance concern any longer (7). In our example, the business expert improves the process so that independent tasks are executed in parallel. As a result the execution becomes faster and fewer violations occur. Using the MORSE versioning capabilities, the new model can be added to the repository, and used in MDD generations henceforth.

5 Performance and Scalability Evaluation

For determining how many queries per second can be processed by the MORSE repository, we have conducted runtime performance and scalability tests on our prototype as

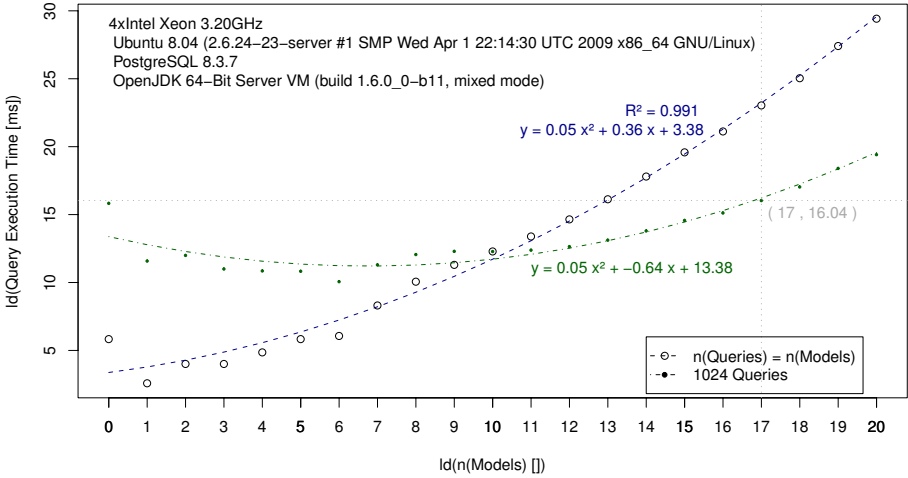


Fig. 5. MORSE Performance and Scalability

shown in Figure 5. We measured the execution time for queries (ordinate) of a repository containing a given number of models (abscissa) and found polynomial execution time ($R^2 > 0.99$). For example, to interpret Figure 5 a MORSE repository with up to 2^{17} (131 072) models can process at least 15 queries ($\leq (2^{16.04}/1024/10^3)^{-1}$) within one second. This means, performance and scalability is far better than what is needed in scenarios similar to our case study that work with long-running process instances which emit events only from time to time.

Thus, models can be retrieved during execution at a low cost, especially when assuming typical Web service invocation latency. Limitations of our prototype and hardware arise with increased number of models, process instances, and events that trigger lookups, e.g., huge MORSE repositories with more than $\approx 2^{17}$ models cannot perform 10^6 lookups per day (arising, e.g., from 10^2 processes with 10^2 instances each that generate 10^2 events per day each). Further scalability, however, can be achieved using clusters and caches.

6 Related Work

In our approach we assume that a monitoring infrastructure is used throughout the architecture for observing and analyzing runtime events (cf. Section 2). While such monitoring infrastructure can be integrated with MORSE and used for the compliance checking, our work particularly focuses on relating to models, the monitored systems have been generated from. Thus, our work makes such models accessible at runtime. Note, that not only, e.g., process models but also compliance concern models are managed by MORSE. This allows for the novel and direct linkage and correlation of model-driven system and requirements models. In this section we refer and relate to work in the areas of runtime requirements-monitoring and model management in form of model repositories.

Feather et al. [10] discuss an architecture and a development process for monitoring system requirements at runtime. It builds on work on goal-driven requirements engineering [21] and runtime requirements monitoring [22].

Skene and Emmerich [3] apply MDD technologies for producing runtime requirements monitoring systems. This is, required behavior is modeled and code is generated for, e.g., the eventing infrastructure. Finally, a meta-data repository collects system data and runs consistency checks to discover violations. While in our work we also showcase the generation of code for the eventing infrastructure (see Section 3), our approach assumes an existent monitoring infrastructure. In case of a violation the MORSE approach not only allows us to relate to requirement models but also to the models of the monitored system.

Chowdhary et al. [4] present a MDD framework and methodology for creating Business Performance Management (BPM) solutions. This is, a guideline is described for implementing complex BPM solutions using an MDD approach. Also, with inter alia the specification of BPM requirements and goals the framework provides runtime support for (generating) the eventing infrastructure, data warehouse, and dashboard. The presented approach allows for the monitoring and analysis of business processes in respect of their performance. Thus, similarly to our approach, compliance concerns such as quality of service concerns as found in service level agreements can be specified and monitored by the framework. Besides the monitoring of business processes and service-based systems in general, our approach particularly focuses on also relating to conceptual models of the systems from the runtime, not only their requirements. As a consequence, the system and end-users can directly relate to the MDD artifacts of a system in case of a violation. This allows for the subsequent reflection, adaptation, and evolution of the system. In contrast, the BPM solution supports compensation, i.e., the execution of business actions according to a decision map.

Another model-based design for the runtime monitoring of quality of service aspects is presented by Ahluwalia et al. [23]. Particularly, an interaction domain model and an infrastructure for the monitoring of deadlines are illustrated. In this approach, system functions are abstracted from interacting components. While a model-driven approach is applied for code generation, the presented model of system services is only related to these in a sense that it reflects them. This is, it is not a source model for the model-driven development of the services. In contrast, MORSE manages and is aware of the real models, systems are generated from. This allows for root cause analysis and evolution of as demonstrated in the presented case study (see Figure 4).

Besides the monitoring of runtime requirements in form of compliance concern models, the MORSE approach particularly focuses on the management of models of service-based systems and their accessibility during runtime. For this reason, a model repository with versioning capabilities is deployed (see Section 3). It abstracts from modeling technologies and its UUID-based implementation allows for a straightforward identification of models and model elements. Other model repositories such as ModelBus [24] and ModelCVS [25] primarily aim at model-based tool integration. AMOR [26,27] and Odyssey-VCS 2 [28] particularly have a focus on the versioning aspect of model management (see also [29]), e.g., for the conflict resolution in collaborative development (cf. [30]). These works mainly focus on the design time: e.g.,

ModelBus addresses the heterogeneity and distribution of modeling tools and focuses on integrating functionality such as model verification, transformation, or testing into a service bus. MORSE, in contrast, focuses on runtime services and processes and their integration, e.g., through monitoring, with the repository and builds on the simple identification for making models accessible at runtime.

7 Conclusion

Monitoring and analysis of models and model elements at runtime is a real and urgent requirement in complex, Internet-based systems. Given the continuously growing adoption of model-driven development practices and the rising complexity of service-based systems, we have shown the usefulness of shifting the focus of model management from design time to runtime. As a first step into this direction, in this article we have presented MORSE, an implementation of a model-aware repository and service environment concept that treats models as first-class citizens at runtime of a service-based system. The MORSE approach significantly eases the management of complex service-based systems by improving the analyzability, e.g., of monitoring data. This has been demonstrated in an industrial case study for compliance management. The benefits of our approach can be achieved with acceptable performance and scalability impacts. While our approach facilitates monitoring, it can also be beneficial to other fields of application that profit from accessing models at runtime, e.g., in adaptive systems.

Acknowledgments

This work was supported by the European Union FP7 project COMPAS, grant no. 215175.

References

1. Mayer, P., Schroeder, A., Koch, N.: MDD4SOA: Model-driven service orchestration. In: EDOC, pp. 203–212. IEEE Computer Society, Los Alamitos (2008)
2. Zdu, U., Hentrich, C., Dustdar, S.: Modeling process-driven and service-oriented architectures using patterns and pattern primitives. *TWEB* 1(3) (2007)
3. Skene, J., Emmerich, W.: Engineering runtime requirements-monitoring systems using mda technologies. In: De Nicola, R., Sangiorgi, D. (eds.) *TGC 2005*. LNCS, vol. 3705, pp. 319–333. Springer, Heidelberg (2005)
4. Chowdhary, P., Bhaskaran, K., Caswell, N.S., Chang, H., Chao, T., Chen, S.K., Dikun, M.J., Lei, H., Jeng, J.J., Kapoor, S., Lang, C.A., Mihaila, G.A., Stanoi, I., Zeng, L.: Model driven development for business performance management. *IBM Systems Journal* 45(3), 587–606 (2006)
5. Völter, M., Stahl, T.: *Model-Driven Software Development: Technology Engineering Management*. Wiley, Chichester (2006)
6. Bank for International Settlements: *Basel II: International Convergence of Capital Measurement and Capital Standards: A Revised Framework - Comprehensive Version (June 2006)*, <http://www.bis.org/publ/bcbsca.htm> (accessed in February 2010)

7. Congress of the United States: Public Company Accounting Reform and Investor Protection Act (Sarbanes-Oxley Act), Pub.L. 107-204, 116 Stat. 745 (July 2002), <http://www.gpo.gov/fdsys/pkg/PLAW-107publ204/content-detail.html> (accessed in February 2010)
8. Holmes, T., Zdun, U., Dustdar, S.: MORSE: A Model-Aware Service Environment. In: Proceedings of the 4th IEEE Asia-Pacific Services Computing Conference (APSCC), December 2009, pp. 470–477. IEEE Computer Society Press, Los Alamitos (2009)
9. Organization for the Advancement of Structured Information Standards: Web service business process execution language version 2.0. OASIS Standard, OASIS Web Services Business Process Execution Language (WSBPEL) TC (January 2007) (accessed in February 2010)
10. Feather, M., Fickas, S., van Lamsweerde, A., Ponsard, C.: Reconciling system requirements and runtime behavior. In: Proceedings of Ninth International Workshop on Software Specification and Design, April 1998, pp. 50–59 (1998)
11. Michelson, B.: Event-Driven Architecture Overview: Event-Driven SOA Is Just Part of the EDA Story (February 2006), <http://www.omg.org/soa/Uploaded%20Docs/EDA/bda2-2-06cc.pdf> (accessed in February 2010)
12. International Organization for Standardization: ISO/IEC 19501:2005 information technology – open distributed processing – unified modeling language (UML), v1.4.2 (April 2005), <http://www.omg.org/cgi-bin/doc?formal/05-04-01> (accessed in February 2010)
13. The Apache Software Foundation: Apache Subversion (2000), <http://subversion.apache.org> (accessed in February 2010)
14. Eclipse Modeling Framework Project (EMF) (2002), <http://www.eclipse.org/modeling/emf/> (accessed in February 2010)
15. The Elver Project: Teneo (2005), <http://www.eclipse.org/modeling/emf/?project=teneo> (accessed in February 2010)
16. Eclipse Persistence Services Project (EclipseLink) (2008), <http://www.eclipse.org/eclipselink> (accessed in February 2010)
17. PostgreSQL Global Development Group: PostgreSQL (1997), <http://www.postgresql.org> (accessed in February 2010)
18. The Apache Software Foundation: Apache CXF: An Open Source Service Framework, <http://cxf.apache.org> (accessed in February 2010)
19. The Apache Software Foundation: Apache Maven, <http://maven.apache.org> (accessed in February 2010)
20. International Telecommunication Union: ISO/IEC 9834-8 information technology – open systems interconnection – procedures for the operation of OSI registration authorities: Generation and registration of universally unique identifiers (UUIDs) and their use as ASN.1 object identifier components (September 2004), <http://www.itu.int/ITU-T/studygroups/com17/oid/X.667-E.pdf> (accessed in February 2010)
21. Dardenne, A., van Lamsweerde, A., Fickas, S.: Goal-directed requirements acquisition. *Sci. Comput. Program.* 20(1-2), 3–50 (1993)
22. Cohen, D., Feather, M.S., Narayanaswamy, K., Fickas, S.S.: Automatic monitoring of software requirements. In: ICSE 1997: Proceedings of the 19th International Conference on Software Engineering, pp. 602–603. ACM, New York (1997)
23. Ahluwalia, J., Krüger, I.H., Phillips, W., Meisinger, M.: Model-based run-time monitoring of end-to-end deadlines. In: Wolf, W. (ed.) *EMSOFT*, pp. 100–109. ACM, New York (2005)

24. Sriplakich, P., Blanc, X., Gervais, M.P.: Supporting transparent model update in distributed case tool integration. In: Haddad, H. (ed.) SAC, pp. 1759–1766. ACM, New York (2006)
25. Kramler, G., Kappel, G., Reiter, T., Kapsammer, E., Retschitzegger, W., Schwinger, W.: Towards a semantic infrastructure supporting model-based tool integration. In: GaMMa 2006: Proceedings of the 2006 international workshop on Global integrated model management, pp. 43–46. ACM, New York (2006)
26. Altmanninger, K., Kappel, G., Kusel, A., Retschitzegger, W., Seidl, M., Schwinger, W., Wimmer, M.: AMOR – towards adaptable model versioning. In: 1st International Workshop on Model Co-Evolution and Consistency Management, in conjunction with MODELS 2008 (2008)
27. Brosch, P., Langer, P., Seidl, M., Wimmer, M.: Towards end-user adaptable model versioning: The by-example operation recorder. In: CVSM 2009: Proceedings of the 2009 ICSE Workshop on Comparison and Versioning of Software Models, Washington, DC, USA, pp. 55–60. IEEE Computer Society, Los Alamitos (2009)
28. Murta, L., Corrêa, C., Prudêncio, J., Werner, C.: Towards Odyssey-VCS 2: Improvements over a UML-based version control system. In: CVSM 2008: Proceedings of the 2008 international workshop on Comparison and versioning of software models, pp. 25–30. ACM, New York (2008)
29. Altmanninger, K., Seidl, M., Wimmer, M.: A survey on model versioning approaches. *IJWIS* 5(3), 271–304 (2009)
30. Brosch, P., Seidl, M., Wieland, K., Wimmer, M., Langer, P.: We can work it out: Collaborative conflict resolution in model versioning. In: ECSCW 2009: Proceedings of the 11th European Conference on Computer Supported Cooperative Work, pp. 207–214. Springer, Heidelberg (2009)

Modeling and Reasoning about Service-Oriented Applications via Goals and Commitments

Amit K. Chopra, Fabiano Dalpiaz, Paolo Giorgini, and John Mylopoulos

Department of Information Engineering and Computer Science, University of Trento
{chopra,dalpiaz,paolo.giorgini,jm}@disi.unitn.it

Abstract. Service-oriented applications facilitate the exchange of business services among participants. Existing modeling approaches either apply at a lower of abstraction than required for such applications or fail to accommodate the autonomous and heterogeneous nature of the participants. We present a business-level conceptual model that addresses the above shortcomings. The model gives primacy to the participants in a service-oriented application. A key feature of the model is that it cleanly decouples the specification of an application's architecture from the specification of individual participants. We formalize the connection between the two—the reasoning that would help a participant decide if a specific application is suitable for his needs. We implement the reasoning in datalog and apply it to a case study involving car insurance.

Keywords: Business modeling, Commitments, Goals, Service engagements, Service-oriented architecture.

1 Introduction

Service-oriented applications exemplify programming-in-the-large [8]: the architecture of the application takes precedence over the specification of services. An individual service may be designed using any methodology in any programming language as long as it structurally fits in with the rest of the system. Component-based systems embody this philosophy, but service-oriented applications are fundamentally different in that they represent *open systems* [16,21]. A service-oriented application is characterized by the autonomy and heterogeneity of the participants. Application participants engage each other in a service enactment via interaction. Applications are dynamic implying that participants may join or leave as they please. The identity of the participants need not even be known when designing the application. In a sense, open systems take the idea of programming-in-the-large to its logical extreme.

An example of a service-oriented application are auctions on eBay. Multiple autonomous and heterogeneous participants are involved: eBay itself, buyers, sellers, payment processors, credit card companies, shippers, and so on. eBay (the organization) specified the architecture of the application in terms of the roles (seller, bidder, shipper, and so on) and the interaction among them without knowing the identity of the specific participants that would adopt those roles.

For service-oriented applications, it is especially useful to treat the architecture as being largely synonymous with the application itself. The auctions application on eBay

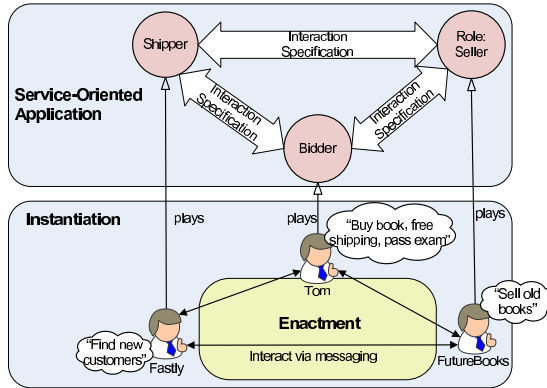


Fig. 1. A service-oriented application is specified in terms of roles. It is instantiated when participants adopt those roles; it is enacted when participants interact according to the adopted roles.

exists whether some auction is going on or not. The application is *instantiated* when participants adopt (play) roles in the application. The application is *enacted* when participants interact according to the roles (see Figure 1). Moreover, the application is, in general, specified independently from the specification of the individual participants. Clearly, the notion of roles, participants, and interaction are key elements in the modeling of service-oriented applications.

The real value of service-oriented computing is realized for applications in which participants engage each other in business transactions, for example, in an auction on eBay. Each individual participant has his own business goals, and he would need to interact flexibly with others so as to be able to fulfill his goals. Ideally, we would want to model both applications and participants in terms of business-level abstractions. We would also want to characterize and reason about properties critical to doing business, such as *goal fulfillment*, *compliance*, *interoperability*, and so on, in similarly high-level terms. This is key to alleviating the business-IT gap.

Existing conceptual modeling approaches either (i) lack the notion of roles, participants, and interactions altogether, or (ii) are lacking in business-level abstractions—they are typically rooted in control and data flow. Workflow-based modeling of applications, as is done using BPMN (the *Business Process Modeling Notation*), exemplifies the former; choreography-based modeling, as is done using WS-CDL (the *Web Services Choreography Description Language*), exemplifies the latter. Many approaches fall somewhere in between (discussed extensively in Section 5).

We propose a conceptual model for service-oriented applications that addresses both the above concerns. It gives primacy to the autonomy and heterogeneity of participants, and works at the business-level. The key insight behind the model is this. A participant will have business *goals* that he wants to achieve. However, given his autonomy, a participant cannot force another to bring about any goal. In fact, a participant wouldn't even know the internal construction—in the forms of business rationale, rules, goals, strategies, procedures, or however otherwise specified—of any other participant. In such situations, the best a participant can do is to deal in *commitments* (concerning the goals he

wants to achieve) with other participants. For example, a bidder on eBay cannot force a seller to deliver even if he has won the auction; the interaction between them proceeds on the understanding that there is a commitment from the seller to deliver if the bidder has won.

This paper synthesizes results from two influential lines of research: goal-oriented requirements engineering [24] and agent communication [19]. Specifically, we specify participants in terms of their goal models, and application architecture in terms of commitments. The conceptual model enables reasoning about properties at a business-level. In this paper, we focus on axiomatizing the *supports* relation, which essentially formalizes the notion of whether adopting a role in a particular application is compatible with a participant’s goals. We implement a prototype reasoning tool that encodes the supports relation in datalog. We evaluate the usefulness of our conceptual model by modeling a car insurance scenario, and describe how we may encode and reason about the model. We also report on experiments that show the scalability of the reasoning.

The rest of the paper is organized as follows. Section 2 describes our conceptual model in detail and shows the relation between individual participants and the application. Section 3 shows how we reason about compatibility between the commitments a participant might be party to and his goals. Section 4 evaluates our approach. We model elements of a car insurance application, and show some queries one may run. The section also reports on the scalability results. Section 5 summarizes our contribution, discusses the relevant literature, and highlights future directions.

2 Conceptual Model

From here on, we refer to the participants in an application as *agents*. This term is appropriate given their autonomous and heterogeneous nature. Figure 2 shows the proposed conceptual model: the left box concerns a service-oriented application, the right box is about an agent’s requirements.

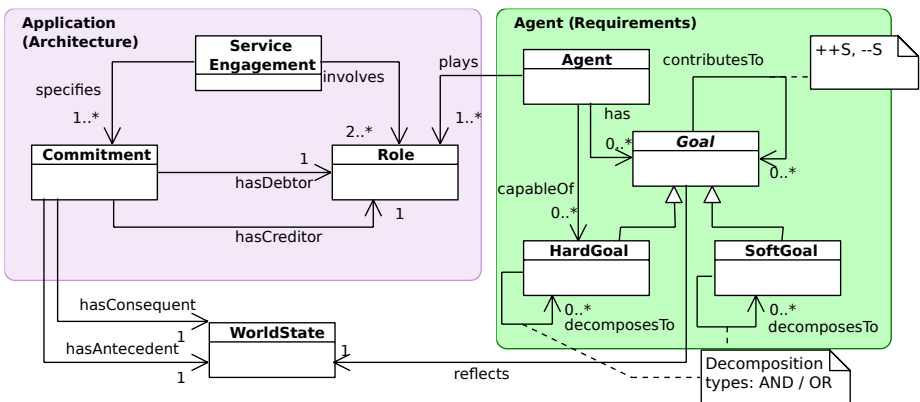


Fig. 2. Conceptual model for service-oriented applications and participating agents

2.1 Specifying Agents via Goal Models

An agent is specified in terms of a goal model, as formalized in the Tropos methodology [2]. Goal modeling captures important aspects of requirements—not just what they are, but why they are there. An agent’s goal model represents his motivations, and abstracts away from low-level details of control and data flow. We now briefly revisit the aspects of goal modeling relevant to this paper.

As shown in Figure 2, an Agent *has* some goals. A Goal may be a HardGoal or a SoftGoal. A softgoal has no clear-cut criteria for satisfaction (its satisfaction is subjectively evaluated). A goal *reflects* a state of the world desired by the agent. A goal may contribute to other goals: $++S(g, g')$ means g contributes positively to the achievement of g' ; $--S(g, g')$ means g contributes negatively to the achievement of g' . Both hard and soft goals may be AND-decomposed or OR-decomposed into subgoals of the same type. Additionally, an agent may be capable of a number of hard-goals; the notion of capability abstracts the means-end relation in Tropos.

2.2 Specifying Applications via Service Engagements

Conceptually, the *commitment* $C(\text{Debtor}, \text{Creditor}, \text{antecedent}, \text{consequent})$ means that the debtor is committed to the creditor for the consequent if the antecedent holds. The *antecedent* and *consequent* are propositions that refer to the *states of the world* of relevance to the application under consideration. A commitment is *discharged* when its consequent is achieved; it is *detached* when the antecedent holds. An unconditional commitment is one where the antecedent is \top (true).

For example, in an auction application, one can imagine a commitment $C(\text{Bidder}, \text{Seller}, \text{wonBid}, \text{paymentMade})$. Informally, it means that the bidder commits to the seller that if the world state is such that he has won the bid, then he will bring about the world state where the payment has been made.

We use commitments as the basis of architectural connections. As Figure 2 shows, a Service Engagement *involves* two or more roles and *specifies* one or more commitments among the involved roles. A Role role can be debtor (creditor) in one or more commitments; each commitment has exactly one debtor (creditor). A commitment has an antecedent and a consequent, each representing some state of the world. Table 1 introduces the message types by which agents update commitments [5]. In the table, x, y, \dots are variables over agents, and p, q, \dots are variables over propositions.

Table 1. Messages and their effects; a commitment is understood as a contractual relation

| Message | Sender | Receiver | Effect | Business Significance |
|-----------------------------|--------|----------|----------------------|--------------------------------------|
| Create(x, y, r, u) | x | y | $C(x, y, r, u)$ | brings about a relation |
| Cancel(x, y, r, u) | x | y | $\neg C(x, y, r, u)$ | dissolves relation |
| Release(x, y, r, u) | y | x | $\neg C(x, y, r, u)$ | dissolves relation |
| Delegate(x, y, z, r, u) | x | z | $C(z, y, r, u)$ | delegates relation to another debtor |
| Assign(x, y, z, r, u) | y | x | $C(x, z, r, u)$ | assigns relation to another creditor |
| Declare(x, y, p) | x | y | p | informs about some aspect of state |

Conceptually, a *service engagement* is a business-level specification of interaction. It describes the *possible* commitments that may arise between agents adopting the roles, and via the standard messages of Table 1, how the commitments are updated. An engagement should not be interpreted to mean that by simply adopting roles in this engagement the agents will become committed as stated. The commitments themselves would come about at runtime via exchange of messages. Moreover, whether an agent sends a particular message is solely his own decision.

Commitments are made in a certain sociolegal context and represent a contractual relationship between the debtor and the creditor. They yield a notion of compliance expressly suited for service-oriented applications. Agent compliance amounts to the agent not violating any of his commitments towards others. A service engagement specified in terms of commitments does not dictate specific operationalizations (runtime enactments) in terms of when an agent should send or expect to receive particular messages; as long as the agent discharges his commitments, he can act as he pleases [9].

Figure 3 shows the (partial) service engagement for an auction application. Figure 4 shows a possible enactment for the service engagement of Figure 3. The bidder first creates c_B . Then he places bids, possibly increasing his bids (indicated by the dotted bidirectional Bidding arrow). The seller informs the bidder that he has won the bid, which detaches c_B and causes the unconditional commitment $c_{UB} = C(\text{Bidder}, \text{Seller}, \top, \text{paymentMade})$ to hold. Finally, the bidder discharges his commitment by sending the payment.

| |
|---|
| $c_B = C(\text{Bidder}, \text{Seller}, \text{wonBid}, \text{paymentMade})$ $c_S = C(\text{Seller}, \text{Bidder}, \text{paymentMade}, \text{itemDelivered})$ $c_A = C(\text{Auctioneer}, \text{Bidder}, \top, \text{itemCheckedForAuthenticity})$ |
|---|

Fig. 3. A (partial) service engagement depicting an auction application. The labels are for reference purposes only. Figure 4 shows an enactment of this engagement between a bidder agent and a seller agent.

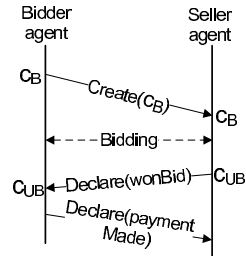


Fig. 4. An enactment

Where do service engagements come from? Domain experts specify these from scratch or by reusing existing specifications that may be available in a repository. In eBay's case, presumably architects, experts on the various kinds of businesses (such as payment processing, shipping, and so on) and processes (auctions) involved, and some initial set of stakeholders got together to define the architecture. How *application requirements* (as distinct from an individual participant's requirements) relate to the specification of service engagements is studied in [9].

2.3 Binding

As Figure 2 shows, an agent may choose to *play*, in other words, adopt one or more roles in a service engagement. Such an agent is termed an *engagement-bound agent*. Adopting a role is the key notion in instantiating an application, as shown in Figure 1.

However, before a bound agent may start interacting, he may want to verify that he is compatible with the engagement. The semantic relationship between a service engagement and an agent's goals is the following. To fulfill his goals, an agent would select a role in some service engagement and check whether adopting that role is compatible with the fulfillment of his goals. If it is compatible, then the agent would presumably act according to the role to fulfill his goals; else, he would look for another service engagement. For example, the bidder may have the requirement of a complete refund from the seller if the seller delivers damaged goods. The bidder must check whether the service engagement with the seller includes a commitment from the seller to that effect; if not, he may try a different service engagement. We formalize compatibility via the notion of *supports* in Section 3.

Notice in Figure 2 that both commitments and goals are expressed in terms of world states. This provides the common ontological basis for reasoning between goal and commitments.

3 Goal and Commitment Support

The conceptual model supports two kinds of compatibility reasoning. Given some role in a service engagement and some goal that the agent wants to achieve, *goal support* verifies whether an agent can *potentially* achieve his goal by playing that role. *Commitment support* checks if an agent playing a role is *potentially* able to honor the commitments he may make as part of playing the role.

Note the usage of the words *support* and *potentially*. Goal (commitment) support is a weaker notion than goal fulfillment; support gives no guarantee about fulfillment at runtime. And yet, it is a more pragmatic notion for open systems, where it is not possible to make such guarantees anyway. For instance, a commitment that an agent depends upon to fulfill his goal may be violated.

Goal support. Agent x (at runtime, or his designer) may execute a query to check whether playing a role ρ in the service engagement under consideration supports x 's goal g . Intuitively, a goal g is supported if (i) no other goal g' that x intends to achieve negatively contributes to g ; and (ii) either of the following holds:

1. x is capable of g , or
2. x can get from some other agent playing role ρ' the commitment $C(\rho', \rho, g', g)$ and x supports g' (akin to x requesting an offer from ρ'), or
3. x can make $C(\rho, \rho', g, g')$ to some agent playing ρ' ; in other words, x commits to g' if the other agent achieves g (akin to x making an offer to ρ'), or
4. g is and-decomposed (or-decomposed) and x supports all (at least one) subgoals, or
5. some other supported goal that x intends to achieve positively contributes to g .

Notice the difference between clauses 2 and 3. In clause 2, x can get a commitment to support a goal g only if x supports the antecedent; in other words, x cannot realistically hope that some agent will play ρ' and will benevolently bring about g . According to clause 3, x can support g without supporting the consequent g' of the commitment.

Table 2. Datalog (DLV-complex) axiomatization of the supports relation

| | |
|--|--|
| $\text{-gs}(X) :- \text{goal}(X), \text{not } v(X).$ | R1. Goals out of the scope cannot be supported. |
| $\text{do}(X) \vee \text{-do}(X) :- \text{cap}(X).$ | R2. Capabilities can be exploited or not. |
| $\text{gs}(X) :- \text{do}(X).$ | R3. Using a capability implies goal support. |
| $\text{gs}(X) :- v(X), \text{gs}(Y), \text{pps}(Y,X).$ $\text{-gs}(X) :- v(X), \text{gs}(Y), \text{mms}(Y,X).$ | R4. ++S and --S apply from and to visible goals. |
| $v(X) :- v(Y), \text{anddec}(Y,L), \text{goal}(X), \#\text{member}(X,L).$ $v(X) :- v(Y), \text{orddec}(Y,L), \text{goal}(X), \#\text{member}(X,L).$ | R5. A subgoal is visible if its parent is visible. |
| $\text{gs}(X) :- \text{orddec}(X,L), \#\text{member}(Y,L), \text{gs}(Y).$ | R6. An or-decomposed goal is supported if any of its subgoals is supported. |
| $\text{gs}(X) :- \text{anddec}(X,Y), \text{suppAllS}(X,Y).$ $\text{wand}(X,Y) :- \text{anddec}(X,Y).$ $\text{wand}(X,Y) :- \text{wand}(X,[A Y]), \text{goal}(A).$ $\text{suppAllS}(A,X) :- \text{wand}(A,X), \#\text{length}(X,N), N=0.$ $\text{suppAllS}(A,X) :- \text{wand}(A,X), \#\text{length}(X,N), N>0, \#\text{memberNth}(X,1,E), \text{gs}(E), \#\text{tail}(X,X1), \text{suppAllS}(A,X1).$ | R7. An and-decomposed goal is supported if all of its subgoals are supported. These rules split the subgoals list into atomic goals. |
| $\text{comm}(X,Y,C) :- \text{cc}(A,B,X,Y,C), \text{plays}(B).$ $\text{comm}([],X,C) :- \text{cc}(A,B,X,Y,C), \text{plays}(A).$ | R8. The agent can exploit only those commitments where it plays debtor or creditor. |
| $e(X) \vee \text{-e}(X) :- \text{comm}(_,_,X).$ $\text{-e}(C) :- \text{comm}(L,Y,C), \text{not } \text{suppAll}(C,L).$ | R9. Commitments can be exploited only if the precondition is supported. |
| $\text{gs}(Y) :- \text{comm}(X,L,C), \#\text{member}(Y,L), \text{goal}(Y), \text{suppAll}(C,X), e(C).$ | R10. A goal is supported by a commitment if it is in the consequent, the antecedent is supported, and the commitment is exploited. |
| $\text{suppAll}(C,X) :- \text{wcomm}(X,_,C), \#\text{length}(X,0).$ $\text{suppAll}(C,X) :- \text{wcomm}(X,_,C), \#\text{length}(X,N), N>0, \#\text{memberNth}(X,1,E), \text{gs}(E), \#\text{tail}(X,X1), \text{suppAll}(C,X1).$ $\text{wcomm}(L1,L2,C) :- \text{comm}(L1,L2,C).$ $\text{wcomm}(L1,L2,C) :- \text{wcomm}([A L1],L2,C), \text{goal}(A).$ | R11. A commitment's antecedent is supported if all the goals in the antecedent are supported. These rules split the antecedent into atomic components. |

Support of the consequent by the debtor (here x) is a matter of checking for commitment support, as explained below.

An important aspect in our reasoning is that of visibility (or scope). Visibility roughly amounts to the goals that an agent intends to achieve. The content of a goal query defines the reasoning scope, namely which are the goals that the agent intends to achieve. Given a query for a goal g , the query scope consists of all the subgoals of the tree starting from the top-level goal ancestor of g . Visibility is important in order to rule out contributions from goals which are not intended.

Goal support is presented with respect to a single goal for the sake of exposition, but this notion is easily generalized to propositions. For instance, one might query for the support of a goal proposition $g_1 \wedge g_2$. In this case, the query scope is the union of the scopes of g_1 and g_2 . Similarly, the antecedent and consequent of a commitment can be expressed using propositions.

Table 2 axiomatizes the above rules as a logic program in datalog¹ (any sufficiently expressive logic programming language would have sufficed for our present purposes).

¹ www.mat.unical.it/dlv-complex

A goal is expressed as an atomic proposition. Antecedent and consequent of commitments are expressed as lists of atomic propositions (the list is interpreted as a conjunction). Given (i) an agent's goal model; (ii) a service engagement; and (iii) the role played by the agent in the engagement, the predicate $gs(g_0 \wedge \dots \wedge g_n)$ is true if and only if each of g_0, \dots, g_n is supported.

A goal model is defined by the following predicates: $goal(g)$ states that g is a goal; $anddec(g, [g_1, \dots, g_n])$ ($orddec(g, [g_1, \dots, g_n])$) denotes that g is and-decomposed (or-decomposed) to g_1, \dots, g_n ; $pps(g_1, g_2)$ ($mms(g_1, g_2)$) represents a ++S (-S) contribution from g_1 to g_2 ; $cap(g)$ says that the agent is capable of goal g . The predicate $cc(r_1, r_2, [g_1, \dots, g_l], [g_m, \dots, g_n])$ indicates the commitment $C(r_1, r_2, g_1 \wedge \dots \wedge g_l, g_m \wedge \dots \wedge g_n)$. The predicate $plays(r)$ states that the considered agent plays role r .

The query scope (the visibility predicate v) is manually defined for what concerns the top-level goals; then it is propagated top-down by rule R5. This may be automated by macros.

Commitment support. It makes sense to check whether an agent will be able to support the commitments it undertakes as part of a service engagement. In other words, let's say to support g , x makes $C(x, y, g, g')$ to an agent y . Now if y brings about g , x will be unconditionally committed to bringing about g' . If x is not able to support g' , then x will potentially be in violation of the commitment. Commitment support reduces to goal support for the commitment consequent.

A reckless or malicious agent may only care that his goals are supported regardless of whether his commitments are supported; a prudent agent on the other hand would ensure that the commitments are also supported.

Reasoning for support as described above offers interesting possibilities. Some examples: (i) x can reason that $C(x, y, g_0, g_1)$ is supported by $C(z, x, g_2, g_1)$ if x supports g_2 ; (ii) x can support a conjunctive goal $g_0 \wedge g_1$ by getting commitments for g_0 and g_1 from two different agents, (iii) to support g in a redundant manner, x may get commitments for g from two different agents; and so on.

4 Evaluation

First, we model a real-life scenario using our conceptual model, and show how we may reason about it. Second, we demonstrate the scalability of the *supports* reasoning.

4.1 Case Study: Insurance Claim Processing

We show how the model and the reasoning techniques can be used to model a real life setting concerning car insurance claim processing. We base our scenario on the documentation that the Financial Services Commission of Ontario (FSCO) provides online, specifically on the description of the claim process². The process describes the perspective of a driver involved in a car accident in Ontario; it also highlights what happens behind the scenes. It describes a service engagement that is independent of specific insurance companies, car repairers, and damage assessors. We assume the car driver is not at fault and his policy has no deductible.

² http://www.fSCO.gov.on.ca/english/insurance/auto/after_auto_accident_ENG.pdf

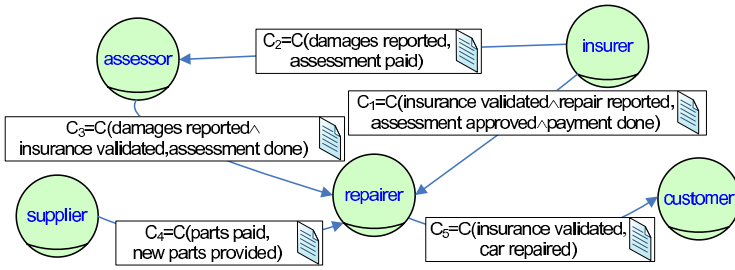


Fig. 5. Role model for the insurance claim processing scenario. Commitments are rectangles that connect (via directed arrows) debtors to creditors.

Table 3. Commitments in the car insurance service engagement

| | |
|-------|--|
| C_1 | insurer to repairer: if insurance has been validated and the repair has been reported, then the insurer will have paid and approved the assessment |
| C_2 | insurer to assessor: if damages have been reported, the assessment will have been paid |
| C_3 | assessor to repairer: if damages have been reported and the insurance has been validated, a damage assessment will have been performed |
| C_4 | supplier to repairer: if parts have been paid for, new parts will have been provided |
| C_5 | repairer to customer: if the insurance has been validated, then the car will have been repaired |

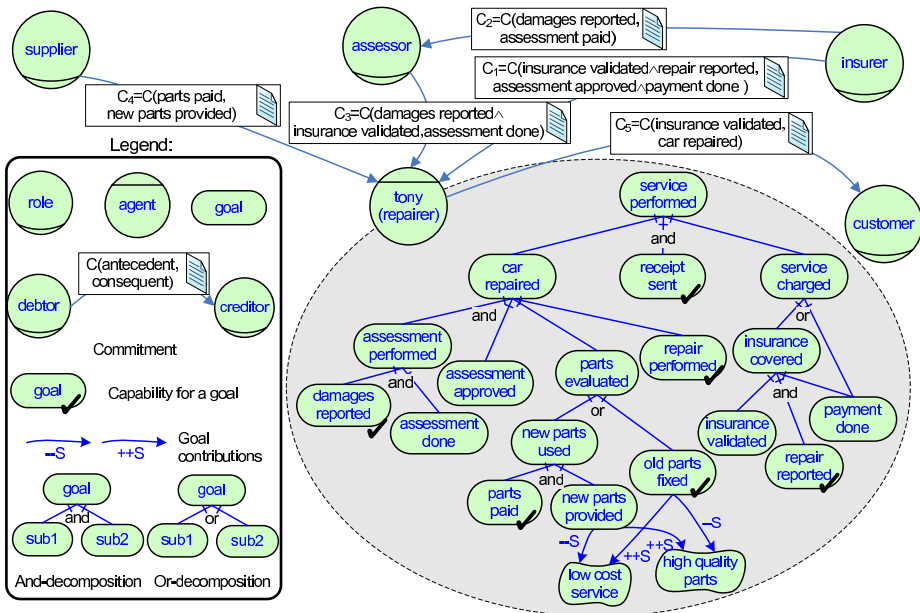


Fig. 6. Visual representation of Tony's insurance-engagement bound specification. Tony plays repairer.

Table 4. Datalog representation of Figure 6

```

% AGENT-ROLE plays relation
plays(repairer).

% GOALS: each goal node is declared, only three shown below
goal(servicePerformed). goal(carRepaired). goal(receiptSent).

% CAPABILITIES
cap(receiptSent). cap(damagesReported). cap(partsPaid).
cap(oldPartsFixed). cap(repairPerformed). cap(repairReported).

% GOAL MODEL: DECOMPOSITIONS
anddec(servicePerformed, [carRepaired, receiptSent, serviceCharged]).
anddec(carRepaired, [assessmentPerformed, assessmentApproved,
    partsEvaluated, repairPerformed]).
anddec(assessmentPerformed, [damagesReported, assessmentDone]).
ordc(partsEvaluated, [newPartsUsed, oldPartsFixed]).
anddec(newPartsUsed, [partsPaid, newPartsProvided]).
ordc(serviceCharged, [insuranceCovered, paymentDone]).
anddec(insuranceCovered, [insuranceValidated, repairReported, paymentDone]).

% GOAL MODEL: CONTRIBUTIONS
mms(newPartsProvided, lowCostService). pps(newPartsProvided, highQualityParts).
pps(oldPartsFixed, lowCostService). mms(oldPartsFixed, highQualityParts).

% COMMITMENTS IN THE SERVICE ENGAGEMENT
cc(insurer, repairer, [insuranceValidated, repairReported],
    [assessmentApproved, paymentDone], c1).
cc(insurer, assessor, [damagesReported], [assessmentPaid], c2).
cc(assessor, repairer, [damagesReported, insuranceValidated], [assessmentDone], c3).
cc(supplier, repairer, [partsPaid], [newPartsProvided], c4).
cc(repairer, customer, [insuranceValidated], [carRepaired], c5).

```

Figure 5 describes the service engagement in the car insurance claim processing scenario. The engagement is defined as a set of roles (circles) connected via commitments; the commitments are labeled c_i . Table 3 explains the commitments.

Figure 6 shows an agent model where agent Tony plays role repairer. The main goal of Tony is to perform a repair service. This is and-decomposed to subgoals car repaired, receipt sent, and service charged. The goal model contains two variation points: the or-decompositions of goals parts evaluated and service charged. The former goal is or-decomposed to subgoals new parts provided and old parts fixed. Note the softgoals low cost service and high quality parts. Using new parts has a negative contribution to low cost service and a positive one to high quality parts, whereas fixing old parts contributes oppositely to those soft goals.

Table 4 show the insurance engagement-bound specification of Tony in datalog. Even though both the service engagement and Tony's requirements are in a single table, we remark that they are independently defined artifacts. The binding of Tony to the repairer role in the engagement is indicated by *plays(repairer)* at the beginning of the specification. We now demonstrate the reasoning.

Table 5 shows some queries for support of particular goals and their solutions. The solutions represent the output that our implementation provides; each solution is a possible strategy to support a goal. A strategy consists of a set of exploited capabilities and

Table 5. Queries (and their solutions) against Tony’s insurance-engagement bound specification

| |
|--|
| <p>Query 1: Can Tony support “service performed”?</p> <p>$v(\text{servicePerformed})$. $gs(\text{servicePerformed})?$</p> <p>Solutions:</p> <p>1: $\{do(\text{receiptSent}), do(\text{repairPerformed}), do(\text{damagesReported}), do(\text{oldPartsFixed}), do(\text{partsPaid}), do(\text{repairReported}), e(c1), e(c3), e(c4), e(c5)\}$</p> <p>2: $\{do(\text{receiptSent}), do(\text{repairPerformed}), do(\text{damagesReported}), do(\text{partsPaid}), do(\text{repairReported}), e(c1), e(c3), e(c4), e(c5)\}$</p> <p>3: $\{do(\text{receiptSent}), do(\text{repairPerformed}), do(\text{damagesReported}), do(\text{oldPartsFixed}), do(\text{partsPaid}), do(\text{repairReported}), e(c1), e(c3), e(c5)\}$</p> <p>4: $\{do(\text{receiptSent}), do(\text{repairPerformed}), do(\text{damagesReported}), do(\text{oldPartsFixed}), do(\text{repairReported}), e(c1), e(c3), e(c5)\}$</p> |
| <p>Query 2: Can Tony support “service performed” and “high quality”?</p> <p>$v(\text{servicePerformed})$. $v(\text{highQualityParts})$. $gs(\text{servicePerformed})$, $gs(\text{highQualityParts})?$</p> <p>Solutions:</p> <p>1: $\{do(\text{receiptSent}), do(\text{repairPerformed}), do(\text{damagesReported}), do(\text{partsPaid}), do(\text{repairReported}), e(c1), e(c3), e(c4), e(c5)\}$</p> |
| <p>Query 3: Can Tony support “service performed” with “high quality” and “low cost”?</p> <p>$v(\text{servicePerformed})$. $v(\text{highQualityParts})$. $v(\text{lowCostService})$. $gs(\text{servicePerformed})$, $gs(\text{lowCostService})$, $gs(\text{highQualityParts})?$</p> <p>Solutions: none</p> |

a set of commitments that the agent can get or make to other agents. Below, we describe the posed queries and we provide some details to explain why the alternatives are valid solutions to the query. The queries pertain to the insurance engagement-bound Tony.

Query 1. Can Tony support service performed?

This query has four solutions (see Table 5). Solution 1 includes both options to fix cars (to support goal parts evaluated): new parts are bought and old parts are fixed. Tony can make commitment c_5 to a customer in order to support insurance validated; he can get c_4 from supplier since Tony supports the antecedent by using his capability for parts paid. In order to get commitments c_1 and c_3 , Tony has to chain commitments: get or make other commitments in order to support the antecedent of another commitment. Tony can get c_3 from an assessor by using his capability for damages reported and chaining c_5 to support insurance validated. Tony can get c_1 from insurer by using his capability for repair reported and chaining c_5 for insurance validated. Solution 1 contains redundant ways to achieve a goal, thus might be useful in order to ensure reliability. Solution 2 involves buying new parts only, and it has the same commitments of solution 1. Solution 4 involves fixing old parts only. Solution 3 includes fixing old parts and also paying new parts, but not c_4 . This option is legitimate, even though not a smart one. Notice how solution 1 and solution 3 are not minimal: indeed, solution 2 is a subset of solution 1, while solution 4 is a subset of solution 3.

Query 2. Can Tony support service performed with high quality?

Verifying this query corresponds to checking goal support for the conjunction of the two goals. This means that goal high quality is in the scope. The effect of this modification is that three solutions for Query 1 are not valid for Query 2. The only valid solution is the former Solution 2. The reason why the other three solutions are not valid is simple: they include goal old parts fixed, which contributes negatively (-S) to high quality.

Query 3. Can Tony support service performed with high quality and low cost?

The third query adds yet another goal in the scope, namely low cost. The effect is that Tony cannot support the conjunction of the three queried goals. The reason for this is that goal new parts provided has a negative contribution to low cost, therefore the only valid solution for Query 2 is not valid for Query 3.

4.2 Scalability Experiments

We evaluated the applicability of our reasoning to medium- and large-sized scenarios by performing some experiments on goal models and service engagements of growing size. Our tests are not intended to assess the absolute performance of the reasoner, rather they aim to check empirically if the query execution time grows linearly or exponentially with the size of the problem.

We create our experiments using a scenario cloning technique: a basic building block is cloned to obtain larger scenarios. The building block consists of a goal model with 9 goals (with one top-level goal, 3 and-decompositions and 1 or-decomposition) and a service engagement with 2 commitments. Cloning this scenario produces a new scenario with 2 top-level goals, 18 goals and 4 commitments; another cloning operation outputs 3 top-level goals, 27 goals and 6 commitments, and so on. The posed query consists of the conjunction of all the top-level goals in the cloned scenario.

Note that cloning linearly increases the number of goals and commitments, whereas it exponentially increases the number of solutions. Cloning is a useful technique to check scalability for our reasoning, given that the important thing is not the number of goals and commitments but the number of solutions. The cloned scenario is characterized by high variability.

The experiments were run on a machine with an AMD Athlon(tm) 64 X2 Dual Core Processor 4200+ CPU, 2GB DIMM DDR2 memory, Linux version 2.6.31-15-generic kernel, DLV-Complex linux static build 20090727. We executed three runs for every experiment; we consider the average time; time was measured using the linux *time* utility and summing the *user* and *sys* values.

Table 6 present the results of the scalability experiments. The first three columns show the number of goals, commitments and solutions, respectively. Notice how the number of solutions grows exponentially: the biggest experiment has almost two millions solutions. The fourth column shows the total time needed to run the experiment; the reasoning is applicable at design time to medium-large models, given that 2 millions solutions are computed in 200 seconds on a desktop computer. The most significant result, however, is in the last column. It shows the average time to derive one solution in microseconds. Notice how the time per solution does not grow exponentially. The

Table 6. Experiments evaluating the scalability of goal support reasoning

| # goals | # comms | # solutions | time (s) | $\frac{time}{\#sol}$ (μs) |
|---------|---------|-------------|----------|----------------------------------|
| 9 | 2 | 5 | 0.009 | 1866 |
| 18 | 4 | 25 | 0.013 | 533 |
| 27 | 6 | 125 | 0.033 | 266 |
| 36 | 8 | 625 | 0.112 | 179 |
| 45 | 10 | 3125 | 0.333 | 107 |
| 54 | 12 | 15625 | 1.361 | 87 |
| 63 | 14 | 78125 | 7.017 | 90 |
| 72 | 16 | 390625 | 37.043 | 95 |
| 81 | 18 | 1953125 | 199.920 | 102 |

average time for smaller experiments is higher because initialization time has a strong impact; time grows pseudo-linearly for bigger experiments.

5 Discussion

The principal contribution of this paper lies in formulating a conceptual model for service-oriented applications that accommodates the autonomy and heterogeneity of their participants, and applies naturally at a business level. We accomplish this by specifying the participants in terms of goals, and the engagements between them in terms of commitments. Our conceptual model has the following salient features. (1) Application architecture is specified in terms of commitments. (2) Commitment are conditional and capture the reciprocal nature of business relationships. (3) Commitments decouple agents: if an agent has a commitment from another, it may not care what goals the other has. We also encoded the reasoning relationship between goals and commitments in datalog, and applied it to a real car insurance scenario. The reasoning helps determine if a chosen service engagement is suitable for a participant’s requirements.

Prominent goal-oriented methodologies such as Tropos [2] and KAOS [23] do not distinguish between application architecture and the requirements of individual agents. The reason is their basis in traditional information systems development where stakeholders cooperate in building a fully specified system. Gordijn *et al.* [10] combine goal modeling with profitability modeling for the various stakeholders; however, their approach shares the monolithic system-development point of view.

One may understand dependencies between *actors* in i^* [24] as an architectural description of the application. However, dependencies do not capture business relationships as commitments do. Guizzardi *et al.* [11] and Telang and Singh [22] highlight the advantages of commitments over dependencies for capturing relationships between roles. Both Telang and Singh [22] and Gordijn *et al.* [10] especially note that dependencies do not capture the reciprocal nature of a business transaction. Bryl *et al.* [3] use a planning-based approach to explore the space of possible alternatives for satisfying some goal; however, unlike us, they follow goal dependencies inside the dependee actors, thus violating heterogeneity. Castro *et al.* [4] highlight the lack of modularity in goal models. Since commitments decouple agents, they significantly alleviate the modularity problem.

Compatibility between a participant and a service engagement is a different kind of correctness criterion compared to checking for progress or safety properties over procedural specifications, as is done for example, in [7]. Mahfouz *et al.* [14] consider the alignment between the goal model of an application, in terms of both dependencies between the actors and their internal goals, and the choreography under consideration. Their approach could be applied in the design of choreographies, that might be then made available as architectural specifications.

Abstractions such as goals and intentions have been used to describe services [13,18]; however such approaches violate heterogeneity by making assumptions about other participants' internals. Specifications of service engagements are eminently more reusable than the goal models of actors [9]. Liu *et al.* [12] formalize commitments in a weaker sense—as a relation between an actor and a service, not between actors, as in done in our approach.

Workflow-based approaches for business processes, for example, [15,17], capture interaction from the viewpoint of a single participant. As such, they may be used to code up individual agents—either as an alternative to goal models or as their operationalization. Benatallah *et al.* [1] formalize properties such the similarity and replaceability for choreographies. Although such approaches are valuable, they are at a lower level of abstraction than service engagements. Such properties have begun to be formalized for service engagements [20]. Especially interesting is the formalization of interoperability in terms of commitments in completely asynchronous settings [5]. The formalization therein completely obviates the need for control flow constructs in service engagements, for example, that an *Accept* or *Reject* should follow the *Order*.

One feature that distinguishes our model from some others in the literature, for example [6], is the emphasis on roles and participants as opposed to on the service itself. In our approach, a service is something that is *realized* when participants interact according to the service engagement. Notice that there is no “service” entity in our conceptual model.

Our approach opens up interesting directions of work. An agent would ideally monitor both his goals and commitments. Compliance with legal and contractual requirements may be formulated directly in terms of commitments, instead of in terms of following a process. An agent would adapt in case some goal is threatened by adopting new strategies; however, in doing so it should ideally also consider his outstanding commitments, else it risks being noncompliant.

Acknowledgments. Research partially supported by FP6-EU project SERENITY contract 27587.

References

1. Benatallah, B., Casati, F., Toumani, F.: Representing, analysing and managing web service protocols. *Data and Knowledge Engineering* 58(3), 327–357 (2006)
2. Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F., Mylopoulos, J.: Tropos: An agent-oriented software development methodology. *Autonomous Agents and Multi-Agent Systems* 8(3), 203–236 (2004)
3. Bryl, V., Giorgini, P., Mylopoulos, J.: Designing socio-technical systems: From stakeholder goals to social networks. *Requirements Engineering* 14(1), 47–70 (2009)

4. Castro, J., Kolp, M., Liu, L., Perini, A.: Dealing with complexity using conceptual models based on Tropos. In: Borgida, A.T., Chaudhri, V.K., Giorgini, P., Yu, E.S. (eds.) Mylopoulos Festschrift. LNCS, vol. 5600, pp. 335–362. Springer, Heidelberg (2009)
5. Chopra, A.K., Singh, M.P.: Multiagent commitment alignment. In: Proceedings of AAMAS 2009, pp. 937–944 (2009)
6. Colombo, M., Di Nitto, E., Di Penta, M., Distanto, D., Zuccalá, M.: Speaking a common language: A conceptual model for describing service-oriented systems. In: Benatallah, B., Casati, F., Traverso, P. (eds.) ICSOC 2005. LNCS, vol. 3826, pp. 48–60. Springer, Heidelberg (2005)
7. Decker, G., Weske, M.: Behavioral consistency for B2B process integration. In: Krogstie, J., Opdahl, A.L., Sindre, G. (eds.) CAiSE 2007 and WES 2007. LNCS, vol. 4495, pp. 81–95. Springer, Heidelberg (2007)
8. De Remer, F., Kron, H.H.: Programming-in-the-large versus programming-in-the small. *IEEE Transactions on Software Engineering* 2(2), 80–86 (1976)
9. Desai, N., Chopra, A.K., Singh, M.P.: Amoeba: A methodology for modeling and evolution of cross-organizational business processes. *ACM Transactions on Software Engineering and Methodology* 19(2) (2010)
10. Gordijn, J., Yu, E., van der Raadt, B.: E-service design using i^* and e^3 value modeling. *IEEE Software* 23(3), 26–33 (2006)
11. Guizzardi, R.S.S., Guizzardi, G., Perini, A., Mylopoulos, J.: Towards an ontological account of agent-oriented goals. In: Choren, R., Garcia, A., Giese, H., Leung, H.-f., Lucena, C., Romanovsky, A. (eds.) SELMAS. LNCS, vol. 4408, pp. 148–164. Springer, Heidelberg (2007)
12. Liu, L., Liu, Q., Chi, C.-H., Jin, Z., Yu, E.: Towards a service requirements modelling ontology based on agent knowledge and intentions. *International Journal of Agent-Oriented Software Engineering* 2(3), 324–349 (2008)
13. Lo, A., Yu, E.: From business models to service-oriented design: A reference catalog approach. In: Parent, C., Schewe, K.-D., Storey, V.C., Thalheim, B. (eds.) ER 2007. LNCS, vol. 4801, pp. 87–101. Springer, Heidelberg (2007)
14. Mahfouz, A., Barroca, L., Laney, R., Nuseibeh, B.: Requirements-driven collaborative choreography customization. In: Baresi, L., Chi, C.-H., Suzuki, J. (eds.) ICSOC-ServiceWave 2009. LNCS, vol. 5900, pp. 144–158. Springer, Heidelberg (2009)
15. Nguyen, D.K., van den Heuvel, W.-J., Papazoglou, M.P., de Castro, V., Marcos, E.: GAM-BUSE: A gap analysis methodology for engineering SOA-based applications. In: Borgida, A.T., Chaudhri, V.K., Giorgini, P., Yu, E.S. (eds.) Mylopoulos Festschrift. LNCS, vol. 5600, pp. 293–318. Springer, Heidelberg (2009)
16. Di Nitto, E., Ghezzi, C., Metzger, A., Papazoglou, M.P., Pohl, K.: A journey to highly dynamic, self-adaptive service-based applications. *Automated Software Engineering* 15(3-4), 313–341 (2008)
17. Ouyang, C., Dumas, M., Van Der Aalst, W.M.P., Ter Hofstede, A.H.M., Mendling, J.: From business process models to process-oriented software systems. *ACM Transactions on Software Engineering and Methodology* 19(1), 1–37 (2009)
18. Rolland, C., Kaabi, R.S., Kraïem, N.: On ISOA: Intentional services oriented architecture. In: Krogstie, J., Opdahl, A.L., Sindre, G. (eds.) CAiSE 2007 and WES 2007. LNCS, vol. 4495, pp. 158–172. Springer, Heidelberg (2007)
19. Singh, M.P.: Agent communication languages: Rethinking the principles. *IEEE Computer* 31(12), 40–47 (1998)
20. Singh, M.P., Chopra, A.K.: Correctness properties for multiagent systems. In: Baldoni, M., van Riemsdijk, M.B. (eds.) DALT 2009. LNCS, vol. 5948, pp. 192–207. Springer, Heidelberg (2010)

21. Singh, M.P., Huhns, M.N.: *Service-Oriented Computing: Semantics, Processes, Agents*. John Wiley & Sons, Chichester (2005)
22. Telang, P.R., Singh, M.P.: Enhancing Tropos with commitments: A business metamodel and methodology. In: Borgida, A.T., Chaudhri, V.K., Giorgini, P., Yu, E.S. (eds.) *Mylopoulos Festschrift*. LNCS, vol. 5600, pp. 417–435. Springer, Heidelberg (2009)
23. van Lamsweerde, A.: From system goals to software architecture. In: Bernardo, M., Inverardi, P. (eds.) *SFM 2003*. LNCS, vol. 2804, pp. 25–43. Springer, Heidelberg (2003)
24. Yu, E.S.K.: Towards modelling and reasoning support for early-phase requirements engineering. In: *Proceedings of ISRE 1997*, pp. 226–235 (1997)

Conceptualizing a Bottom-Up Approach to Service Bundling

Thomas Kohlborn¹, Christian Luebeck², Axel Korthaus¹, Erwin Fieft¹,
Michael Rosemann¹, Christoph Riedl², and Helmut Krömar²

¹ Faculty of Science and Technology, Queensland University of Technology,
126 Margaret Street, 4000 Brisbane, Australia
{t.kohlborn, axel.korthaus, e.fieft, m.rosemann}@qut.edu.au

² Lehrstuhl für Wirtschaftsinformatik, Technische Universität München
85748 Garching b. München, Germany
c.luebeck@mytum.de,
{riedlc, krcmar}@in.tum.de

Abstract. Offering service bundles to the market is a promising option for service providers to strengthen their competitive advantages, cope with dynamic market conditions and deal with heterogeneous consumer demand. Although the expected positive effects of bundling strategies and pricing considerations for bundles are covered well by the available literature, limited guidance can be found regarding the identification of potential bundle candidates and the actual process of bundling. The contribution of this paper is the positioning of bundling based on insights from both business and computer science and the proposition of a structured bundling method, which guides organizations with the composition of bundles in practice.

Keywords: Service, service-orientation, bundling.

1 Introduction

The creation of bundled offers of services and goods with distinguishing and superior characteristics compared to existing offers has long been recognized as an opportunity for companies to increase their competitive advantages over rival contenders in the market [1]. Generally, a bundle represents a package that contains at least two elements and presents a value-add to potential consumers.

While a considerable amount of literature addressing the process of service design or new service development can be found today, less is known about approaches that facilitate the creation of superior service bundles. Despite the fact that companies across all industry sectors with increased market pressures are challenged by the issue of service bundling [2], Moreover, little guidance has been provided so far for the identification of potential bundle candidates and for the actual process of bundling. The single work that specifically targets service bundling is from Baida [3]. The author used an ontology-based approach “to facilitate the automation of the service bundling task”. Using a given customer demand by expressing required resources, the

configuration method (“Serviguration”) creates service bundles that satisfy the demand and adhere to the predefined set of dependencies between services.

In this paper, we provide insights into the foundation and the process of bundling. We propose a new service bundling method that supports organizations in identifying potential service bundles that they could offer to consumers.

The remainder of this paper is structured as follows. Based on the problem description that has been provided in this section, we first define and clarify the term bundling along with related terms from business and computer science to explicate the underlying understanding of the concept of bundling for this work. Subsequently, core aspects and foundations of a proposed approach are presented. The paper ends with a conclusion and directions for further research.

2 Positioning Service Bundling

In order to be able to elaborate further on what service bundling entails, we derive the meaning of the terms service and bundle mainly from marketing, while we refer to the field of computing for characterizing the terms aggregation and composition. Fig. 1 provides an overview of how the concepts denoted by these terms relate.

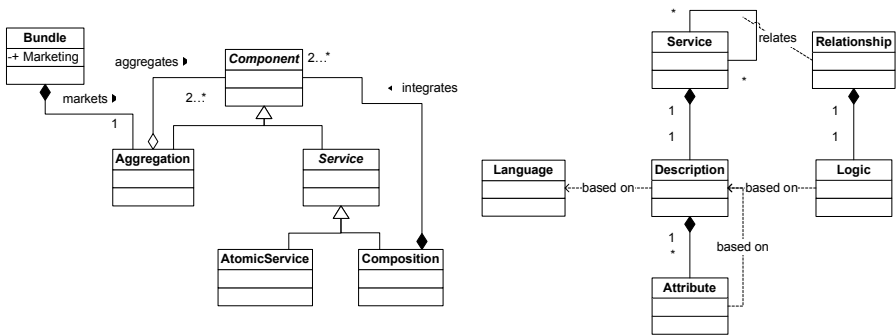


Fig. 1. Conceptual Relationships (in Unified Modeling Language notation)

Service: The term “service” is loaded with different meanings depending on the specific context and universe of discourse. There is no overall standardized definition of service [4]. Taking a marketing perspective, the most cited service characteristics are intangibility, inseparability (of production and consumption), heterogeneity (or non-standardization), and perishability (or exclusion from inventory) [5]. However, these characteristics are more and more critiqued. Therefore, Edvardsson et al. [6] conclude that “we should not generalize the characteristics to all services, but use them for some services when they are relevant and in situations where they are useful and fruitful.” They conclude that at a general level, a service is better conceived as a ‘perspective’ on value creation.

Aggregation: The generic term “aggregation” is defined as “a group, body or mass composed of many distinct parts or individuals” [7]. Hereby, the distinct elements may be loosely associated with each other or share certain attributes. However, the

elements within are distinctively identifiable, only sharing certain commonalities in their characteristics. Elements may be ordered along a process or integrated to a certain extent as long as the elements are still distinctively identifiable. The typical understanding in the computer science domain is that an aggregation will still exist, even if component services are removed from the aggregate [8]. That also relates to the business domain, where an aggregation comprises multiple services and provide access to them in a single location [9].

Composition: A service can either be an atomic service, which is not composed of other services, or it can be a composite service, which comprises other services. Thus, a composition can be regarded as a “condition consisting in the combination or union (material, practical, or ideal) of several things” [10]. Similar to the term “aggregation”, the term “composition” can be found in the domain of software engineering as well. However, in contrast to an aggregation, which still exists if one component element is removed from the aggregation, a composition ceases to exist in case a constituent component service is removed, based upon structural dependencies between these elements [8]. A composition refers to a tightly-coupled integration of sub-services, thus adding value not present in the individual constituent services [9].

Bundle: The generic definition of a bundle is “a collection of things bound or otherwise fastened together” [11]. While the generic definition basically forms no constraints on the elements within the bundle, the marketing literature is more specific and generally agrees on the definition by Stremersch and Tellis [12], who define bundling as “*the sale of two or more separate products in one package*”. The authors further define separate products as products for which separate markets exist. With this definition they try to draw a distinct line between compositions and bundles to preserve the strategic importance of bundling. Thus, bundling adds marketing aspects to aggregations. A bundle is not equivalent to an aggregation, as an aggregation does not possess additional properties (e.g. price) for the whole. Although a pure composition is also characterized by additional properties, it is not equivalent to a bundle, as a bundle consists of distinguishable components and a composition tightly integrates its components to form a single new service.

3 Conceptual Framework for a Service Bundling Method

The proposed method is targeted at the identification of possible service bundles by supporting the early stages of the bundle creation process. The method therefore focuses on limiting the solution space of possible bundles, using indicators that express some form of bundling motivation. It is important to point out that this method is not supposed to omit the evaluation of bundles by a domain expert. It has to be acknowledged that the domain expert is still needed to evaluate the overall feasibility of bundles, since this requires complex analysis, often utilizing tacit knowledge across a range of different disciplines (e.g. economy, marketing, legal). Rather, the aim of this method is to limit the scope of the necessary evaluation for the domain expert. This is in particular relevant with a large number of services and, therefore, many bundling options. The proposed approach leverages existing service descriptions and does not necessitate a time-consuming step of (manually) explicating relationships between services as it is the case with the method described by Baida [3]. Instead,

commonalities of attributes indicate such a relationship. As long as services are consistently described and attributes relevant for this bundling approach are present, the proposed method can be employed. Moreover, Baida [3] relies on a given customer demand to drive the creation of service bundles. While useful for situations where customer demand is well known and understood, poor performance can be expected from this approach when demand is hard to capture or anticipate. Furthermore, the economically desirable situation where customer demand is induced by a new service offering is not supported at all. Our proposed method explicitly targets the latter case by focusing on the creation of new and innovative service bundles. Therefore, customer demand is not utilized to reason about the suitability of potential bundles in this method. Instead, the driving source of this method is a repository of services that are available for bundling. Depending on the given context, this repository might consist of the services of a single provider, a provider network or even contain all available services in a service ecosystem.

Herrmann et al. [13] found that functionally complementary components in a bundle lead to high intentions to purchase compared to bundles in which no complementary components are present. The authors state that, “*as the relationship among the components increased from ‘not at all related’ through ‘somewhat related’ to ‘very related’, intention to purchase also increased*”. The proposed method builds upon these findings and the conjecture that other commonalities or relationships between services can also indicate potentially useful bundles. We define the term relationship as *a connection, whose existence can be evaluated by a logic expression* utilizing service description attributes. Every relationship refers to previously specified attributes (e.g. location of the hotel, destination of the flight) and evaluates them using a given logic (e.g. distance between destination airport and location of the hotel). This evaluation can be realized ranging from simple value comparisons of single attributes to complex algorithms using multiple attributes. The right side of Figure 1 illustrates the corresponding conceptual model using UML.

We distinguish between two types of relationships, namely *generic* and *domain-specific* relationships. A generic relationship is used independently of a concrete domain. These relationships evaluate connections of a general nature that can be found across a range of different domains. The evaluation of generic relationships does not require a domain-specific awareness. A specific relationship only applies to certain domains and can be tailored for concrete bundling scenarios. In this context the notion of domains refers to distinguishable spheres of knowledge that have their own distinct terminologies and semantics. Thus, generic relationships relate to concepts that are similar across existing domains.

Based on given service descriptions and derived relationships, the vast amount of possible service bundles can be filtered in a structured manner to finally extract the most promising bundling candidates. Service bundling can be seen as a configuration task [3] assembling a bundle from a set of services that can only be connected together in certain ways. Ten Teije et al. [14] consider a configuration task as a search problem. The authors state that the configuration space can be restricted in multiple steps. Restricting the configuration space by the possible connections leads to the *possible configuration space*. Applying further constraints leads to the *valid configuration space*. Based on this, user requirements are applied to form the *suitable configuration space*. The approach of constraining a solution space by adding requirements in multiple steps (Fig. 2) adequately supports the act of service bundling.

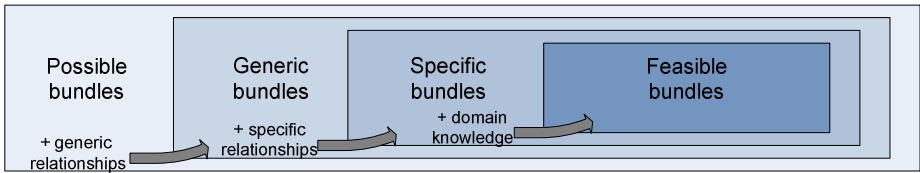


Fig. 2. Constraining the Solution Space

The service repository containing all available services serves as a starting point to form the overall solution space. *Possible bundles* refers to all possible combinations of these services, regardless of validity or feasibility. *Generic bundles* are a subset of all possible bundles which have generic relationships. Since generic relationships do not have to be created or tailored for a specific scenario or domain, they can be easily applied. These bundles are called *generic*, as the indication to bundle is of general nature and oblivious of the domain. Bundles that do not fulfill the requirements of applied generic relationships (e.g. a bundle containing two services that are offered in different cities) are excluded from this set. Based on the set of generic bundles, *specific relationships* that are specific to the domain are evaluated, which leads to a set of *specific bundles*. These bundles are called *specific*, as domain-specific relationships are strong indicators for bundling (compared to generic relationships), since they take a specific environment into account. Once specific bundles are identified, further domain knowledge has to be applied to extract a set of *feasible bundles*. This includes the validation of the bundles with regard to internal and external requirements. Internal requirements might include the strategic alignment of the bundle, quality, service level and risk assessments and other aspects along these lines. External requirements, for example customer demand, market saturation and legislation, also have to be evaluated. The value of a bundle increases with each step-up into a smaller subset of the solution space. As this work focuses on the identification of bundling candidates, the creation of feasible bundles is out of the scope of this work. While *generic* and *specific* bundles can be identified using the presented notion of relationships, feasible bundles require a domain expert, as the final compilation of a bundle requires complex analysis, which can only be supported to a certain extent by analyzing the relationships between services.

4 Conclusion

This paper defines service bundling and related concepts and proposes a novel approach for service bundling that identifies service bundle candidates. While the process of new service development has been extensively researched and conceptualized, the process of finding suitable service bundling candidates is still ill-defined. The proposed method facilitates the creation of bundles by providing organizations with systematic and practical approach. The developed method builds on service bundling concepts from both the marketing and the technological literature, thereby addressing the increased need for multi-disciplinary approaches and business-IT alignment. Multiple directions for further research can be identified. First, research in the area of service descriptions has to be conducted to develop a universal language that is

applicable across industries and covers business as well as software services. Second, strategies and rationales of service bundling need to be analyzed further, to provide valuable insights for the internal and external validation of initially identified bundles. At this stage, the proposed relationships have to be seen as a working set, which will evolve as additional studies and evaluations are carried out.

Acknowledgements. This research was carried out as part of the activities of, and funded by, the Smart Services Cooperative Research Centre (CRC) through the Australian Government's CRC Programme (Department of Innovation, Industry, Science and Research).

References

1. Lawless, M.W.: Commodity Bundling for Competitive Advantage: Strategic Implications. *Journal of Management Studies* 28, 267–280 (1991)
2. Akkermans, H., Baida, Z., Gordijn, J., Peiiia, N., Altuna, A., Laresgoiti, I.: Value Webs: Using Ontologies to Bundle Real-World Services. *IEEE Int. Systems* 19, 57–66 (2004)
3. Baida, Z.: Software-Aided Service Bundling. *Intelligent Methods & Tools for Graphical Service Modeling*. In: Dutch Graduate School for Information and Knowledge Systems. Vrije Universiteit, Amsterdam (2006)
4. Baida, Z., Gordijn, J., Omelayenko, B.: A Shared Service Terminology for Online Service Provisioning. In: *Proceedings of the 6th International Conference on Electronic Commerce (ICEC)*, pp. 1–10 (2004)
5. Zeithaml, V.A., Parasuraman, A., Berry, L.L.: Problems and Strategies in Services Marketing. *Journal of Marketing* 49, 33–46 (1985)
6. Edvardsson, B., Gustafsson, A., Roos, I.: Service Portraits in Service Research: A Critical Review. *International Journal of Service Industry Management* 16, 107–121 (2005)
7. Anonymous: Aggregation, Merriam-Webster Online Dictionary (2009), <http://www.merriam-webster.com/dictionary/aggregation>
8. Evermann, J., Wand, Y.: Ontology Based Object-Oriented Domain Modelling: Fundamental Concepts. *Requirements Engineering* 10, 146–160 (2005)
9. O'Sullivan, J., Edmond, D., ter Hofstede, A.: What's in a Service? *Distributed Parallel Databases* 12, 117–133 (2002)
10. Anonymous: Composition. In: *Oxford English Dictionary*. Oxford University Press, Oxford (1989)
11. Anonymous: Bundle, n. In: *Oxford English Dictionary*. Oxford University Press, Oxford (1989)
12. Stremersch, S., Tellis, G.J.: Strategic Bundling of Products and Prices: A New Synthesis for Marketing. *Journal of Marketing* 66, 55–72 (2002)
13. Herrmann, A., Huber, F., Coulter, R.H.: Product and Service Bundling Decisions and Their Effects on Purchase Intention. In: Fuerderer, R., Herrmann, A., Wuebker, G. (eds.) *Optimal Bundling: Marketing Strategies for Improving Economic Performance*, pp. 253–268. Springer, Heidelberg (1999)
14. ten Teije, A., van Harmelen, F., Schreiber, A.T., Wielinga, B.J.: Construction of Problem-Solving Methods as Parametric Design. *International Journal of Human-Computer Studies* 49, 289–363 (1998)

Dynamic Authorisation Policies for Event-Based Task Delegation

Khaled Gaaloul, Ehtesham Zahoor, François Charoy, and Claude Godart

LORIA - Nancy University - UMR 7503
BP 239, F-54506 Vandœuvre-lès-Nancy Cedex, France
{kgaaloul,zahoor,charoy,godart}@loria.fr

Abstract. Task delegation presents one of the business process security leitmotifs. It defines a mechanism that bridges the gap between both workflow and access control systems. There are two important issues relating to delegation, namely allowing task delegation to complete, and having a secure delegation within a workflow. Delegation completion and authorisation enforcement are specified under specific constraints. Constraints are defined from the delegation context implying the presence of a fixed set of delegation events to control the delegation execution.

In this paper, we aim to reason about delegation events to specify delegation policies dynamically. To that end, we present an event-based task delegation model to monitor the delegation process. We then identify relevant events for authorisation enforcement to specify delegation policies. Moreover, we propose a technique that automates delegation policies using event calculus to control the delegation execution and increase the compliance of all delegation changes in the global policy.

Keywords: Workflow, task, delegation, policy, event calculus.

1 Introduction

The pace at which business is conducted has increased dramatically over recent years, forcing many companies to re-evaluate the efficiency of their business processes. The restructuring of organisational policies and methods for conducting business has been termed "Business Process Re-engineering" [1]. These refined business processes are automated in workflows that ensure the secure and efficient flow of information between activities and users that constitute the business process. Workflows aim to model and control the execution of business processes cross organisations. Typically, organisations establish a set of security policies, that regulate how the business process and resources should be managed [2].

In previous work, we argued that business processes execution are determined by a mix of ad-hoc as well as process-based interactions. This highly dynamic environment must be supported by mechanisms allowing flexibility, security and on-the-fly shift of rights and responsibilities both on a (atomic) task level and on a (global) process level [3]. To address those issues, we present a task delegation approach as a mechanism supporting organisational flexibility in workflow management systems, and ensuring delegation of authority in access control systems

[4]. However, most of the work done in the area of business process security does not treat delegation in sufficient details. On one hand, Atluri *et al.* presented the Workflow Authorisation Model (WAM) that concentrates on the enforcement of authorisation flow in the domain of workflow security [5]. WAM remains static in time and poor in terms of delegation constraints within a workflow. On the other hand, existing work on access control systems do not consider dynamic enforcement of authorisation policies [6].

Moreover, a secure task delegation model has to separate various aspects of delegation within a workflow, where the interactions between workflow invariants (e.g., users, tasks and data) are triggered by delegation events. These delegation events will imply appropriate authorisation requests from access control systems. At present, responses arising from access control requests are stateless such that a response is given to an access request depending on predefined policies during the planning phase. If, however, this response changes due to a policy adaptation for delegation, no mechanism currently exists that allows the new response to be generated in the authorisation policy dynamically. Currently, when delegating a task, often the reasoning behind this is dependent on transient conditions called events. When one of these conditions changes during execution, our access policy decision may change. We do believe that delegation events define dynamic constraints for authorisation policies that should not be neglected in advanced security mechanisms supporting delegation.

The scope of the paper is to investigate the potential of delegation events to ensure a secure task delegation within a workflow. Securing delegation involves the definition of authorisation policies which are compliant with the workflow policy. In order to tackle these problems we need to address two important issues, namely allowing the delegation task to complete, and having a secure delegation within a workflow. Allowing task delegation to complete requires a task model that forms the basis of what can be analysed during the delegation process within a workflow. Secure delegation implies the controlled propagation of authority during task execution. Based on specific events, we define delegation policies in an automatic manner. In order to control the delegation behaviour and to specify its authorisation policies dynamically, we gather relevant events that will define both the task execution path and the generated policies for the delegation of authority. Using *Event Calculus* (EC), we will present a technique to monitor the delegation execution. Afterwards, we will briefly discuss the event calculus formalism to generate delegation policies, and finally, we explain how generated policies will change dynamically in response to task delegation events.

The remainder of this paper is organised as follows. Section 2 presents a workflow scenario to motivate our work. In section 3, we give an overview of our approach to reason about events to specify authorisation policies for delegation. In section 4, we present our task delegation model and explain how events will control its execution. Section 5 focuses on security requirements for delegation and its authorisation policy specifications using EC. In section 6, we motivate our technique to support delegation automation. Section 7 discusses related work. Finally, we conclude and outline several topics of potential future work.

2 Motivating Example: A Use Case Requiring Delegation Policies Integration

To understand the motivation of our research, we present a real world processes from an e-government case study requiring delegation. Mutual Legal Assistance (MLA) defines a workflow scenario involving national authorities of two European countries. Here we describe the MLA process part in the Eurojust organisation A. Users with roles *Prosecutor* and *Assistant* are assigned to execute the MLA process and activities that are part of the process are represented as tasks (see Figure [11](#)).

In this scenario, the task "Translate Documents" T3 is originally only accessible by the user member of role *Prosecutor*, a fact defined in the workflow policy. We define a workflow policy as a level of defining access to task resources. We denote P an authorisation policy for the MLA process. This task is a long-running task and is expected to take 5 working days to complete. The *Prosecutor* is unavailable to execute this task due to illness, and will delegate it to a subordinate involved in the MLA process. *Assistant* is a subordinate to *Prosecutor* in the organisational role hierarchy. During delegation, the policy P is updated so that user with role *Assistant* is now allowed to complete task T3. To that end, he issues an access control request to the policy P to grant the access, and executes the task T3. As such, users with roles *Prosecutor* and *Assistant* are here the delegator and the delegatee, respectively.

The authorisation policy P needs to reflect the new requirements for delegation. In order to derive a delegation policy from the existing policy, we have to specify additional authorisation rules to support delegation, where a rule defines the policy decision effect (e.g., Permit, Deny). Considering a user-to-user delegation, we motivated that such delegation is done in ad-hoc manner, thereby supporting a negotiation protocol. We consider negotiation as a fundamental step for delegation. It involves all the principals (delegator and delegatee) and negotiation specifications (e.g., time, evidence). Our intention is to envisage a wide-ranging request that gives flexibility for the delegation request. Subsequently, such specifications have to be included in the delegation policy to define specific conditions to validate the policy decision effect.

Returning to our example, the delegator *Prosecutor* sends a delegation request for all users members of role "Assistant". This defines a *push* delegation mode, where a delegatee is chosen dynamically based on the negotiation step. An acceptance of delegation inquires a new access control enforcement in the existing policy, thereby adding a new authorisation rule for the delegatee under defined conditions (i.e., time) and/or obligations (i.e., evidence) agreed between the delegation principals. The *Prosecutor* may need to review all the translations done by his *Assistant* for validation. Validation is done based on evidence defined during negotiation. Evidence can be related to the language of translated documents or the number of translated documents within 5 day. To that end, an authorisation rule permitting the access (e.g. *read*, *write*, *execute*) to the legal document in the MLA Information Service, is constrained by an obligation allowing to investigate whether evidence were satisfactorily met. If however,

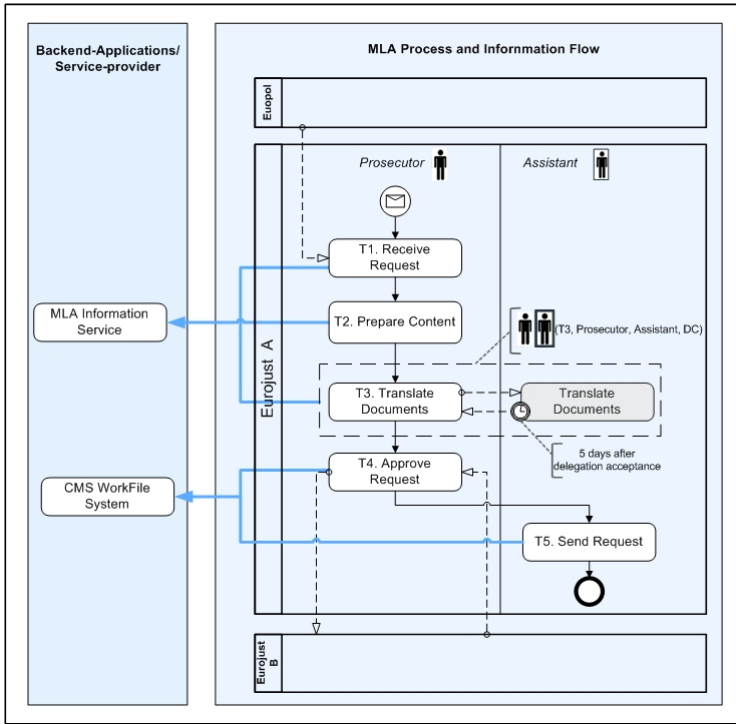


Fig. 1. MLA delegation scenario

evidence are not satisfied, a revoke action may be triggered including a deny result for the previous policy effect.

In traditional access control frameworks no mechanism exists that would support such delegation constraints. Delegation constraints are meant to automate delegation policies from existing policy specifications. Accordingly, it is not possible to foresee a deny rule for revocation during the policy definition. Moreover, a manual review of the current access control rights and task executions is costly, labor intensive, and prone to errors. With an automated mechanism, when the policy changes to reflect delegation, the delegation policy will be derived automatically based on specific facts related to the delegation process. A delegation process defines a task delegation life cycle within the existing process. It is enriched with additional constraints to be compliant with the organisational policies. Organisational policies establish a set of security policies, that regulate how the business process and resources should be managed. Delegation constraints will inquire the need to support specific interactions that would be automatically captured, and specified in the delegation policies for appropriate actions. We do believe that such interactions are intermediate states in the delegation process driven by specified events to control the delegation behaviour within a workflow.

3 The Proposed Framework

The scope of the paper is to investigate the potential of delegation events to ensure a secure task delegation within a workflow. Securing delegation involves the definition of authorisation policies which are compliant with the global policy of the workflow. Therefore, these delegation events will imply appropriate authorisations on the delegatee side for further actions as well as contain specific constraints for those actions (e.g., mode, time, evidence). In order to tackle these problems we need to address two important issues, namely allowing the delegation task to complete, and having a secure delegation within a workflow. To that end, we introduce a delegation model that forms the basis of what can be analysed during the delegation process in terms of monitoring and security.

The monitoring of task delegation is an essential step to ensure delegation completion. A delegated task goes through different states to be terminated. States depends on generated events during the delegation life cycle. Events such as *revoke* or *cancel* are an integral part of the delegation behaviour. Revoking a task may be necessary when a task is outdated or an access right is abused. Moreover, additional events such as *validate* may be required when a delegation request is issued under a certain obligation where the delegatee has to perform specific evidence to validate the task execution. For instance, the delegation of T3 can generate evidence related to the number of translated documents within a period of 5 day. Subsequently, evidence validation will be an important step in the delegation process. Dealing with that, we came up with an event-based task delegation model (TDM) that can fulfill all these requirements. Our model aspires to offer a full defined model supporting all kind of task delegation for human oriented-interactions [3].

Additionally, we consider task delegation as an advanced security mechanism supporting policy decision. We define an approach to support dynamic delegation of authority within an access control framework. The novelty consists of reasoning on authorisation based on task delegation events, and specifying them in terms of delegation policies. When one of these events changes, our access policy decision may change implying dynamic delegation of authority. Existing work on access control systems remain stateless and do not consider this perspective. We propose a task delegation framework to support automated enforcement of delegation policies. Delegation policies are defined from existing policy and are specified from triggered events. For instance, T3 evidence are not satisfied and the validation will trigger the event *revoke* for the delegatee. T3 is not anymore authorised to be executed by the delegatee. In this case, another rule has to be integrated in policy with an effect of deny for the authorisation.

In order to control the delegation behaviour and to specify its authorisation policies in an automated manner, we gather specific events that will define both the task execution path and the generated policies for the delegation of authority. Using *Event Calculus*, we will present a technique to monitor the delegation execution. Afterwards, we will briefly discuss the event calculus formalism to generate delegation policies, and finally, we explain how generated policies change dynamically in response to task delegation events.

4 Task Delegation Model (TDM)

In this section, we present our task delegation model to monitor the delegation execution. Our model is based on events that covers different aspects of delegation. It defines how delegation request is issued and then executed depending on delegation constraints. The idea is to offer a technique to monitor delegation execution based on the triggered events. Using *Event Calculus*, we can foresee the delegation behaviour within its process.

4.1 Introduction to TDM

First, we present a detailed model of task execution that illustrates the delegation process. The task life cycle is based on additional events. The figure below depicts a state diagram of our TDM from the time that a task is created through its final completion, cancellation or failure. It can be seen that there are series of potential states that comprise this process. A task, once created, is generally assigned to a user. The assigned user can choose to start it immediately or to delegate it. Delegation depends on the assignment transition, where the assigned user has the authority to delegate the task to a delegatee in order to act on his behalf.

Our model is based on events that covers different aspects of delegation. It defines how a delegation request is issued. *Pull* mode assumes that a delegator has at his disposal a pool of delegatees to be selected to work on his behalf. *Push* mode assumes that a delegator is waiting for an acceptance from a potential delegatee [4]. Moreover, delegation of privileges can be classified into grant or transfer [7]. A *grant* delegation model allows a delegated access right (privileges) to be available for both delegator and delegatee. As such, the delegator is still having the control to *validate* or *revoke* the task, and the delegatee to *execute* it. However, in *transfer* delegation models, the ability to use a delegated access right is transferred to the delegatee; in particular, the delegated access right is no longer available to the delegator. There is no validation required and the task is terminated (*complete/fail*) by the delegatee.

Each edge within the diagram is prefixed with either an *S* or *U* indicating that the transition is initiated by the workflow system or the human resource respectively, with $(u_1, u_2) \in U$ where *U* is a set of users, u_1 the delegator and u_2 the delegatee. In the following, we define a task delegation relation as follows:

Definition 1. We define a task delegation relation $RD = (t, u_1, u_2, DC)$, where t is the delegated task and $t \in T$ a set of tasks that composes a workflow, and DC the delegation constraints.

For instance, delegation constraints can be related to time or evidence specifications. Moreover, a role hierarchy (RH) defines the delegation relation condition in a user-to-user delegation. Returning to the example, the delegation relation $(T3, Prosecutor, Assistant, (RH, 5 \text{ days})) \in RD$.

4.2 Modelling Task Delegation in Event Calculus

Background and motivations: The proposed approach for the representation of task delegation process relies on the *Event Calculus* (EC) [8]. The choice of EC is motivated by several reasons for delegation. Actually, given the representation of the task delegation model, policies and the corresponding events that trigger policy changes specified in the EC, an event calculus reasoner can be used to reason about them.

Event Calculus is a logic programming formalism for representing events and is being widely used for modeling different aspects such as flexible process design, monitoring and verification [9]. It comprises the following elements: \mathcal{A} is the set of *events* (or actions), \mathcal{F} is the set of *fluents* (fluents are *reified* [1]), \mathcal{T} is the set of time points, and \mathcal{X} is a set of objects related to the particular context. In EC, events are the core concept that triggers changes to the world. A fluent is anything whose value is subject to change over time. EC uses predicates to specify actions and their effects. Basic event calculus predicates used for modelling the proposed framework are:

- *Initiates*(e, f, t) - fluent f holds after timepoint t if event e happens at t .
- *Terminates*(e, f, t) - fluent f does not hold after timepoint t if event e happens at t .
- *Happens*(e, t) is true iff event e happens at timepoint t .
- *HoldsAt*(f, t) is true iff fluent f holds at timepoint t .
- *Initially*(f) - fluent f holds from time 0.
- *Clipped*(t_1, f, t_2) - fluent f was terminated during time interval $[t_1, t_2]$.
- *Declipped*(t_1, f, t_2) - fluent f was initiated during time interval $[t_1, t_2]$.

The reasoning modes provided by event calculus can be broadly categorised into abductive, deductive and inductive tasks. In reference to our proposal, given a TDM and authorisation policies one may be interested to find a plan for task delegation, that allows to identify what possible actions (policy changes) will result from the task delegation and may opt to choose the optimal plan in terms of minimal policy changes, this leads to the "abduction reasoning". Then, one may also be interested to find out the possible effects (including policy changes) for a given set of actions (a set of events that will allow task delegation), this leads to the choice of "deduction reasoning" and using the event calculus is thus twofold.

The event calculus models discussed in this paper are presented using the discrete event calculus language [10] and we will only present the simplified models that represent the core aspects. In the models, all the variables (such as task, time) are universally quantified and in case of existential quantification, it is represent with variable name within curly brackets, {variablename}.

Event calculus based model: The basic entities in the proposed model are tasks. In terms of discrete event calculus terminology they can be considered

¹ Fluents are first-class objects which can be quantified over and can appear as the arguments to predicates.

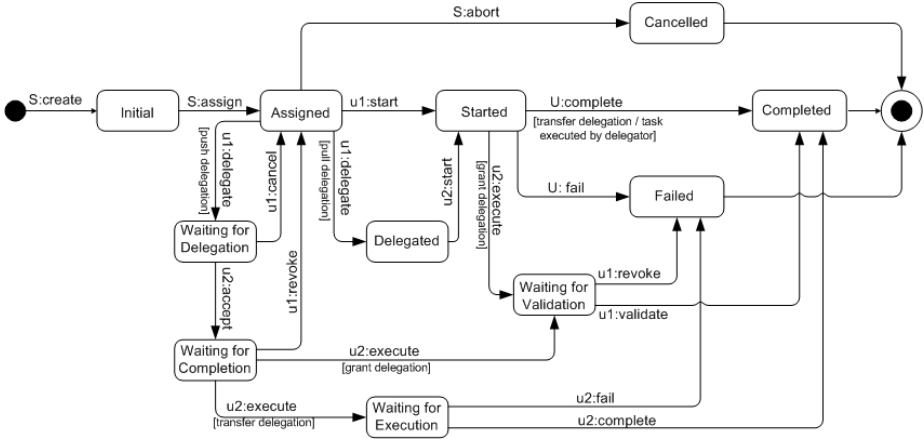


Fig. 2. Task delegation model

as sorts, of which instances can be created. Then, each task can be in different states during the delegation execution. In reference to the task delegation model presented earlier (see Figure 2), the possible task states include Initial, Assigned, Delegated, Completed and others. As task states change over time, they can thus be regarded as fluents in event calculus terminology. Further, the state change is governed by a set of actions/events and in relation to task delegation model, the task state changes from Initial to Assigned as a result of *assign* event occurring. Finally the task delegation model introduces a set of orderings, such as the state of a task cannot be assigned, if it is not created earlier. In reference to event calculus model, we will introduce a set of axioms to handle these dependencies. The event calculus model below introduces the fluents, basic events and dependency axioms:

The event calculus model presented above, first defines sort and fluents that marks the different task states. Then we define an event *Create(task)*, and an Initiates axiom that specifies that the fluent *Initial(task)* continues to hold after the event happens at some time. Similarly, we define the event/axiom for the

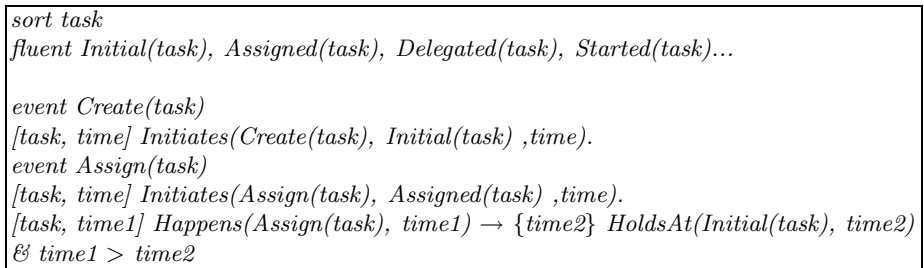


Fig. 3. Event calculus based task delegation model

assignment event and fluent. We further introduce an axiom that specifies that in order to assign some task at time1, that task must already be created and thus in Initial state at time2, and time1 is greater than time2. In a similar fashion, we can define events and associated Initial axioms for the complete TDM model, space limitations restrict us to discuss them further.

For the basic event calculus model above, the solutions (plans) returned by the reasoner may also include the trivial plans which does not enforce the delegation and directly *start* or *abort* the task once assigned. In order to give the user ability to choose the delegation mode once the task is assigned (see Figure 2), we enrich the model to include the following axioms:

$[task, time] \text{ !Happens}(Abort(task), time).$
 $[task, time] \text{ !Happens}(Start(task), time).$
 $[task, time] \text{ !Happens}(PullDelegate(task), time).$

Fig. 4. Delegation mode choice

The event calculus model above, specifies that the task does not either Start, Abort or requires PullDelegation once assigned (and thus the only option for the reasoner is to conclude that the model requires a PushDelegation mode). We can similarly restrict the delegation permission (Grant/Transfer), once the task is in the WaitingCompletion state.

5 Authorisation Policies for TDM

In this section, we analyse security requirements that need to be taken into account to define delegation policies based-events. Additional requirements such as *pull/push* mode and *grant/transfer* type may be a source to a policy change during delegation. Using *Event Calculus*, we present a technique capable of computing and generating new policy rules automatically.

5.1 Building Policies for Delegation

We define delegation transitions as events ruling delegation behaviour. The internal behaviour is based on events defined in our TDM, and may be a source to a policy change, thereby requiring the integration of additional authorisation rules.

Definition 2. We define a policy $P = (target, rule, C)$, where *target* defines where a policy is applicable, *rule* is a set of rules that defines the policy decision result, and *C* the policy constraints set that validates the policy rule. A delegation policy is a policy $P_D = (target_D, rule_D, C_D)$, where $target_D = RD$, $rule_D \subseteq rule$ and $C_D \subset C$ and $C_D = DC \cup events$.

A policy rule may include conditions and obligations which are used to identify various conditions or cases under which a policy may become applicable. Based on

Table 1. Push delegation policy rules-based events

| Delegation Events | Push Delegation | | Adding Policy Rule |
|-----------------------|-----------------|----------|-----------------------------------|
| | Grant | Transfer | |
| $u1:delegate$ | ✓ | ✓ | No add |
| $u2:accept$ | ✓ | ✓ | Add rules based on execution type |
| $u1:cancel$ | ✓ | ✓ | No add |
| $u2:execute/Grant$ | ✓ | | (Permit,Push,Grant:Evidence) |
| $u2:execute/Transfer$ | | ✓ | (Permit,Push,Transfer:NoEvidence) |
| $u1:validate$ | ✓ | | No add |
| $u1:revoke$ | ✓ | | (Deny,Push,Grant) |
| $u2:fail$ | | ✓ | (Deny,Push,Transfer) |
| $u2:complete$ | | ✓ | No add |

the result of these rules different policies can become applicable. We define a rule as a tuple (*effect, condition, obligation*), where effect returns the policy decision result (permit, deny), condition defines the delegation mode (push, pull) and obligation checks evidence. In the following, we analyse security requirements that need to be taken into account in a *push* mode to define delegation policy rules based on our TDM (see Figure 2). We present a table that gathers specific events for *push* delegation and analyse them in terms of policy rules. Adding rules in the workflow policy will ensure the delegation of authority, thereby adding the required effect (permit, deny) to the delegation policy rules (see Table 1).

Returning to the example, we can observe a dynamic policy enforcement during delegation. Initially, T3 is delegated to the *Assistant* u_2 and the delegation policy for T3: $P_D = (RD, Permit, (Push, 5\ days))$ (see Table 1/ $u2:execute/Grant$). In the meanwhile, the *Prosecutor* u_1 is back to work before delegation is done and is not satisfied with the work progress and would revoke what was performed by his assistant so far. The *Prosecutor* is once again able to claim the task and will revoke the policy effect (permit) for the *Assistant*. The event *revoke* will be updated in the policy, and a deny rule is then updated in the policy. Thus, the delegation policy for T3 needs to generate a new rule and the delegation policy is updated to: $P_D = (RD, Deny, (Push, Grant))$ (see Table 1/ $u1:revoke$). Note that the generated rule depends on the *RD* relation to check access rights conflicts. We determined access rights granting based on the current task status and its resources requirements using a task-based access control model for delegation presented in previous work [3]. Our access control model ensures task delegation assignments and resources access to delegates corresponding to the global policy constraints.

5.2 Modelling Delegation Policies in Event Calculus

In order to model the delegation policy rules-based events, we introduce new sorts called *effect, condition, obligation* to the event calculus model for the table defined above and specify instances of each sort to be the possible effects, conditions and obligations. Possible effects include *Deny* and *Permit* results, and

```

sort task, effect, condition, obligation
effect Permit, Deny    condition Push, Pull    obligation Grant, Transfer, ..

fluent RuleAdded(effect, condition, obligation)
event AddPolicyRule(effect, condition, obligation)
[event, condition, obligation, time]
Initiates(AddPolicyRule(effect, condition, obligation),
RuleAdded(effect, condition, obligation), time).

;policy change
[task, time] Happens(PushDelegateAcceptExecuteGrant(task), time) →
Happens(AddPolicyRule(Permit, Push, Evidence), time)
[task, time] Happens(PushDelegateAcceptFailTransfer(task), time) →
Happens(AddPolicyRule(Deny, Push, Transfer), time)

```

Fig. 5. Delegation policy

conditions define the *Push* and *Pull* mode. The possible instances for obligations include *Grant*, *Transfer*, *Evidence* and *NoEvidence* which are constraints related to delegation type and mode. We further add an action *AddRule(effect, condition, obligation)* and corresponding axiom and enrich the model to specify the policy changes as a result of events (see Figure 5).

The policy change axioms presented above specify that once certain actions happen, they cause policy change and thus we add a new rule to the global policy. The name of actions/events depicts their invocation hierarchy, *PushDelegateAcceptExecuteGrant* is the *execute* event with a grant permission once the *PushDelegation* request has been accepted by a delegatee and has to be validated before completion (see Table 1).

6 Delegation Automation

In this section, we motivate our event-based approach supporting delegation automation. Automation is necessary for both the task completion and the policy specification. Reasoning on delegation events using *Event Calculus* offers a solution to foresee the delegation execution and increase the control and compliance of all delegation changes.

6.1 Benefits

Through this paper, we motivated the event-based approach for monitoring and securing task delegation. We observed that based on specific assumptions we are able to control any delegation request. We defined different valid orders of executions for delegation. The order of executions are computed automatically based on events. Events can distinguish between the order of execution by checking the delegation mode and type. For instance, an execution expects a validation transition if and only if we are in a *grant* delegation.

Additionally, we defined a technique to specify delegation policies automatically. By reasoning on specific events, we are able to address the policy stateless issue. We can compute delegation policies from triggered events during task execution. Policy automation offers many benefits. Actually, it reduces efforts for users and administrators. Administrator efforts can be related to processes definition and policies specification. Moreover, it increases control and compliance of all delegation changes. Subsequently, a delegation request is accomplished under constraints which are compliant to the global policy. For instance, time constraint has to be taken into account when granting a temporal access for delegation (i.e., T3 deadline in Figure 2).

6.2 Reasoning

In our study, we utilise the Discrete Event Calculus Reasoner (DECReasoner²) for performing automated commonsense reasoning using the event calculus, a comprehensive and highly usable logic-based formalism. It solves problems efficiently by converting them into satisfiability (SAT) problems.

The event calculus based on the task delegation model and policy specifications can be used to reason about the delegation process. As we discussed earlier, the reasoning task can either be abductive or deductive. For the abductive reasoning, a plan is sought for the specified goal. In reference to our proposal, the goal is to have either the task in completed, cancelled or failed state, so we add the goal $[task] \text{ HoldsAt}(\text{Completed}(task), 15) \mid \text{ HoldsAt}(\text{Failed}(task), 15) \mid \text{ HoldsAt}(\text{Cancelled}(task), 15)$ to the event calculus model presented above and add an instance of the delegated task T3. The invocation of the event calculus reasoner will then give us a set of possible solutions (called plans) for achieving the goal. Let us first consider the case, when the chosen delegation mode is PushDelegation with the grant of permissions to the delegatee, the event calculus reasoner returns the plan as follows.

The execution plan follows the delegation of T3 described in the use case. It shows the action that need to be taken for delegation and most importantly, it shows the possible policy changes as a result of delegation. Steps from 1 to 11 depicts the delegation process execution. Having a *push* mode as a condition, we derive the relevant rules to add in the policy. For instance, step 8 and 9 show when a delegatee request the task T3 for execution. Delegatee acceptance went through the "WaitingDelegation", "WaitingAcceptance" and "WaitingCompletion" states (see Figure 2). Based on those events, we deduce that an authorisation rule is added at this stage under a certain obligation (evidence validation), and finally a task validation complete the delegation execution (see steps 10 and 11 in Figure 6).

All the defined axioms using the DECReasoner language can be given to the reasoner for finding a solution (if exists) to support policy changes, which automatically orients these axioms into delegation rules. Then, given as inputs the specification of the conditions and obligations expressed when adding rules

² For more details: <http://decreasoner.sourceforge.net/>

```

1389 variables and 7290 clauses
relsat solver
1 model
—
model 1:
0 Happens(Create(T3), 0).
1 +Initial(T3).
2 Happens(Assign(T3), 2).
3 +Assigned(T3).
4 Happens(PushDelegate(T3), 4).
5 +WaitingDelegation(T3).
6 Happens(PushDelegateAccept(T3), 6).
7 +WaitingCompletion(T3).
8 Happens(PushDelegateAcceptExecuteGrant(T3), 8).
Happens(AddPolicyRule(Permit, Push, Evidence), 8).
9 +RuleAdded(Permit, Push, Evidence).
+WaitingValidation(T3).
10 Happens(PushDelegateAcceptExecuteGrantValidate(T3), 10).
11 +Completed(T3).
—
;DECReasoner execution details
0 predicates, 0 functions, 12 fluents, 20 events, 90 axioms
encoding 3.1s - solution 0.7s - total 5.8s

```

Fig. 6. Delegation plan

(using EC), the generated plan by the reasoner shows that either the authorisation rules result in a permit or a deny decision³.

In concrete policy changes, there are two possible scenarios. The first scenario is the integration of a new authorisation policy because the conjectures (conditions or obligations) are valid. The second one concerns cases corresponding to an overriding of this rule to a deny result. Note that, we can leverage the trace of DECReasoner to give all necessary information (events, fluents and time-points) to help designer (policy administrator) to detect policies problems in the deployed process. Finally, the event calculus model can further be enriched to ensure minimal policy changes in the execution plans using auditing techniques. Space limitations restrict us to discuss them further.

7 Related Work

The Workflow Authorisation Model (WAM) presents a conceptual, logical and execution model that concentrates on the enforcement of authorisation flows in task dependency and transaction processing [5]. In addition, Atluri *et al.* discussed the specification of temporal constraints in a static approach, which is

³ For space reasons, verification results and encoding details can be found at <http://webloria.loria.fr/~kgaaloul/caise2010/DelegReasoner.rar>

not sufficient to support workflow security in general and task delegation in particular. This is due to workflows needing a more dynamic approach to synchronise the flow of authorisations during the workflow execution. The Workflow Authorisation Model does not discuss the order of operation flow such as our task delegation process. In a workflow, we need to investigate the delegation control in different aspects such as tasks, events and data by leveraging delegation constraints to support authorisation policies.

Role-based Access Control (RBAC) is recognised as an efficient access control model for large organisations. Most organisations have some business rules related to access control policy [11]. In [12][13], authors extend the RBAC96 model by defining some delegation rules. They proposed a flexible delegation model named Permission-based Delegation Model (PBDM), where users may want to delegate a piece of permission from a role [13]. However, neither RBAC nor PBDM support offer a suitable solution to support task delegation constraints. We do believe that constraints such as *Push/Pull* mode or *Grant/Transfer* privileges are essential for delegation and have an impact on the security requirements during policies specification [3].

The eXtensible Access Control Markup Language (XACML) is an XML-based, declarative access control policy language that lets policy editors specify rules about who can do what and when. Policies comprising rules, possibly restricted by conditions, may be specified and targeted at subjects, resources and actions. Subjects, resources, actions and conditions are matched with information in an authorisation request context using attribute values and a rich set of value-matching functions. The outcome or effect of a policy evaluation may be Permit, Deny, NotApplicable or Indeterminate. In [14], Seitz and Firozabadi added new structured data-types to express chains of delegation and constraints on delegation using XACML. The main result of their research is an administrative approach that does not support ad-hoc delegation and lacks of explicit support for task delegation constraints.

8 Conclusion and Future Work

Providing access control mechanisms to support dynamic delegation of authority in workflow systems, is a non-trivial task to model and engineer. In this paper we have presented problems and requirements that such a model demands, and proposed a technique for delegation to specify delegation policies automatically. The motivation of this direction is based on a real world process from an e-government case study, where a task delegation may support changes during execution. Delegation policies may change according to specific events. We defined the nature of events based on task delegation constraints, and described their interactions with policies decisions. When relevant events occur, we define how delegation will behave and how policy rules change dynamically in response to this change. Using *Event Calculus* formalism, we implemented our technique and deployed a use case scenario for task delegation ensuring the required authorisation policy changes.

The next stage of our work is the implementation of our framework using XACML standard. We propose an extension supporting task delegation constraints with regards to the XACML conditions and obligations specifications. Future work will look also at enriching our approach with additional delegation constraints supporting historical records. Delegation history will be used to record delegation that have been made to address administrative requirements such as auditing.

References

1. Venter, K., Olivier, M.S.: The delegation authorization model: A model for the dynamic delegation of authorization rights in a secure workflow management system. In: CCITT Recommendation X.420, Blue Book (2002)
2. Vijayalakshmi, A., Janice, W.: Supporting conditional delegation in secure workflow management systems. In: SACMAT 2005: Proceedings of the Tenth ACM Symposium on Access Control Models and Technologies, pp. 49–58. ACM, New York (2005)
3. Gaaloul, K., Charoy, F.: Task delegation based access control models for workflow systems. In: I3E 2009: Proceedings of Software Services for e-Business and e-Society, 9th IFIP WG 6.1 Conference on e-Business, e-Services and e-Society, Nancy, France, September 23-25. IFIP, vol. 305. Springer, Heidelberg (2009)
4. Gaaloul, K., Miseldine, P., Charoy, F.: Towards proactive policies supporting event-based task delegation. In: The International Conference on Emerging Security Information, Systems, and Technologies, pp. 99–104 (2009)
5. Atluri, V., Huang, W., Bertino, E.: An execution model for multilevel secure workflows. In: Proceedings of the IFIP WG11.3 Eleventh International Conference on Database Security, pp. 151–165. Chapman & Hall, Ltd., London (1998)
6. Bertino, E., Castano, S., Ferrari, E., Mesiti, M.: Specifying and enforcing access control policies for xml document sources. *World Wide Web* 3(3), 139–151 (2000)
7. Crampton, J., Khambhammettu, H.: Delegation in role-based access control. In: Gollmann, D., Meier, J., Sabelfeld, A. (eds.) ESORICS 2006. LNCS, vol. 4189, pp. 174–191. Springer, Heidelberg (2006)
8. Kowalski, R.A., Sergot, M.J.: A logic-based calculus of events. *New Generation Comput.* 4(1), 67–95 (1986)
9. Zahoor, E., Perrin, O., Godart, C.: A declarative approach to timed-properties aware Web services composition, INRIA internal report 00455405 (February 2010)
10. Mueller, E.T.: *Commonsense Reasoning*. Morgan Kaufmann Publishers Inc., USA (2006)
11. Sandhu, R., Coyne, E.J., Feinstein, H.L., Youman, C.E.: Role-based access control models. *IEEE Computer* 29(2), 38–47 (1996)
12. Barka, E., Sandhu, R.: Framework for role-based delegation models. In: ACSAC 2000: Proceedings of the 16th Annual Computer Security Applications Conference, Washington, DC, USA, p. 168. IEEE Computer Society, Los Alamitos (2000)
13. Zhang, X., Oh, S., Sandhu, R.: PBDM: a flexible delegation model in RBAC. In: SACMAT 2003: Proceedings of the Eighth ACM Symposium on Access Control Models and Technologies, pp. 149–157. ACM Press, New York (2003)
14. Seitz, L., Rissanen, E., Sandholm, T., Firozabadi, B., Mulmo, O.: Policy administration control and delegation using xacml and delegent. In: Proceedings of 6th IEEE/ACM International Conference on Grid Computing (GRID 2005), Seattle, Washington, USA, November 13-14, pp. 49–54 (2005)

A New Approach for Pattern Problem Detection

Nadia Bouassida¹ and Hanène Ben-Abdallah²

Mir@cl Laboratory,

¹ Institut Supérieur d'Informatique et de Multimédia

² Faculté des Sciences Economiques et de Gestion

Sfax University, Tunisia

Nadia.Bouassida@isimsf.rnu.tn, Hanene.BenAbdallah@fsegs.rnu.tn

Abstract. Despite their advantages in design quality improvement and rapid software development, design patterns remain difficult to reuse for inexperienced designers. The main difficulty consists in how to recognize the applicability of an appropriate design pattern for a particular application. Design problems can be found in a design with different shapes, unfortunately, often through poor solutions. To deal with this situation, we propose an approach that recognizes pattern problems in a design and that assists in transforming them into their corresponding design patterns. Our approach adapts an XML document retrieval technique to detect the situation necessitating a pattern usage. Unlike current approaches, ours accounts for both the structural and semantic aspects of a pattern problem. In addition, it tolerates design alterations of pattern problems.

Keywords: Design pattern identification, design pattern problem, pattern instantiation.

1 Introduction

It is irrefutable that the quality of a design highly impacts the quality of the code and, possibly, the performance and maintenance of software products. This vital importance of design quality justifies several efforts invested to define design techniques based on reuse of *proven solutions*, *e.g.*, components, design patterns [6], frameworks and models [23]. Reusing such solutions allows a designer to profit from prior experiences to produce a higher quality design more rapidly. Design patterns are among the highly advocated reuse techniques, which are also exploited in all recent reuse techniques; they can be instantiated and composed by a designer in order to produce a component, a framework and/or an application model.

However, to benefit from design patterns, a designer must have a thorough understanding of and a good practice with design patterns in order to identify the pattern problem and thereafter the appropriate patterns to instantiate for its application. The absence of such high-level of expertise motivated several researchers to propose assistance through the detection of patterns (*cf.*, [13]), spoiled patterns or anti-patterns (*cf.*, [1], [2]) and pattern problems (*cf.*, [10]) in a design. The proposed approaches differ mainly in the pattern concepts they consider (*i.e.*, only the structure, the structure and

the method invocations/declarations [21]) and the degree of structural discordance they tolerate: exact match [12] or partial match [16], [13] [21]. These differences can be justified by the objectives behind each approach: code improvement through re-engineering (*cf.*, [12],[9],[14]) *vs.* design improvement (*cf.*, [8],[10]).

In our work, we are interested in improving design during its construction. In this context, a designer (inexperienced with design patterns) may inadvertently specify a design fragment that “resembles” a design pattern. The resemblance can be manifested as: 1) a correct, but extended and/or reduced, structural instantiation of a pattern; 2) a close representation of a pattern problem; or 3) a poor/spoiled instantiation of a pattern. In fact, while a design pattern is considered as a “good” design solution for a frequent design problem, the second and third cases are considered as “bad” design solutions. In addition, a pattern problem can be found in different shapes and forms; unfortunately, they are often represented in designs through poor solutions [10]. Hence, we argue that a design pattern problem identification technique would help designers in all three resemblance cases. One limit of existing identification approaches is that they can handle only one of the above three resemblance cases with no tolerance to variability in the design.

Furthermore, contrary to design patterns, the detection of design pattern problems requires an analysis of the semantics in addition to the structure. Indeed, the names of the classes and their operations distinguish one pattern problem from another. For instance, even though the *State* and *Strategy* patterns have the same structure [3], they represent different problems through different class and operation names. Hence, to detect and correct a pattern problem, one needs to identify first the structure of the problem (or one that is similar to it) and validate the identification through the problem’s semantics.

In this paper, we illustrate how our approach based on an XML document retrieval technique (presented in [21]) can be fine-tuned to identify all three design resemblance cases. Being based on an XML retrieval technique, our approach identifies the similarities between the structure of the pattern problem (*i.e.*, pattern misuses) and a given application design. It offers two advantages over graph/matrix matching and constraint propagation based approaches used by other researchers. First, it can detect similarities between a *group* of classes in the application and the pattern problem. Secondly, it can tolerate controlled alterations of the pattern problem in the design.

The remainder of this paper is organized as follows. Section 2 overviews currently proposed approaches for pattern and pattern problem identification. Section 3 presents our technique for pattern problem identification which adapts an XML retrieval approach. Section 4 presents the pattern problem identification through examples of pattern problems or bad solutions and their transformation into a pattern solution. Section 5 summarizes the paper and outlines future work.

2 Related Work

2.1 Current Works for Pattern Detection

Several works have been interested in pattern identification but for different purposes. The main purpose of detecting design patterns in a reverse engineering context is to

obtain information to understand and possibly restructuring a program. As an example, the work of Tansalis [13] aims at the identification of design patterns as part of the reengineering process. It proposes a design pattern detection methodology based on similarity scoring between graph vertices. For this, it encodes the pattern and design information as matrixes and relies on a graph matching algorithm to detect the pattern. Source code analysis is applied too late in the development process to offer a valuable assistance in the design phase.

On the other hand, for design analysis purposes, Dong *et al.* [20] also use matrixes to encode the pattern information. They use a template matching method to calculate the normalized cross correlation between the two matrices. A normalized cross correlation shows the degree of similarity between a design pattern and an analyzed part of the design.

Albin-Amiot *et al.* [9] present a set of tools and techniques to help OO software practitioners design, understand, and re-engineer a piece of software, using design-patterns. A first prototype tool, Patterns-Box, provides assistance in designing the architecture of a new piece of software, while a second prototype tool, Ptidej, identifies design patterns used in existing software.

2.2 Current Works for Pattern Problem Detection

The work of Mili and Boussaidi [10] [15] represents the design problem that the pattern is meant to solve explicitly. This explicit representation is one step towards the automatic detection and application of patterns. In fact, this work aims at recognizing occurrences of the problem solved by the design patterns which are then transformed according to the solution proposed by the design pattern. It uses an MDA approach where it identifies instances of pattern problems in a given model using a meta-model of the problem, marks the appropriate entities, and finally applies an appropriate transformation to get the pattern solution. The models are considered as graphs and model transformations are a special kind of graph transformations. The transformation is seen as a rule-based implementation of graph grammars. The current implementation is in the ECLIPSE Modeling Framework.

El-Boussaidi *et al.* [10] consider that design problems solvable by design patterns are sometimes badly understood, which often produces poor solutions to modeling requirements. To limit bad designs, this work proposes a semi-automatic tool for marking models (with the variable parts), identifying pattern problems using constraint satisfaction techniques, and then transforming the design by instantiating the appropriate design patterns. The tool relies on an explicit representation of design problems solved by design patterns. It uses a meta-model of the pattern problem *structure* to look for an *exact* match of the pattern problem. In other words, this work does not handle incomplete designs or designs similar to the pattern problem. Moreover, since it does not capture the behavioral aspect of patterns, it may not be as efficient for behavioral patterns.

Alikacem *et al.* [19] propose an approach to detect design problems using metrics to evaluate the source code quality. Their objective is to detect violations of quality rules in object-oriented programs. On the other hand, Ciupke [20] model the quality rules using fuzzy rule based systems. This approach is used to detect automatically design problems in an OO reengineering context using logic programming. In both of

these works, design problems are defined as violations of OO code-quality rules and, hence, are of a lower level of abstraction than the design problems solved by design patterns [10].

Bouhours *et al.* [2] propose a detection approach for “bad smells in a design” that can be remodeled through the use of design patterns. The automatic detection and the explanation of the misconceptions are performed thanks to *spoiled* patterns. A spoiled pattern is a pattern that allows to instantiate inadequate solutions for a given problem: requirements are respected, but architecture is improvable. Thus, this work considers only the structural information, while the behavior is very important.

Moha *et al.* [4][5] are interested in design smells which are poor solutions to recurring design problems. They present detection algorithms for design smells and validate the detection algorithms in terms of precision and recall.

Overall, none of the proposed approaches and tools allows pattern problem detection and its similar structures using structural and behavioral information.

3 Pattern Problem Identification

As argued previously, the structural similarity to a pattern problem is insufficient to decide upon the necessity of a particular design pattern. Semantics, in terms of pattern class names and method declarations within the classes, is also required to confirm the presence of a pattern problem and, hence, the necessity of a corresponding design pattern.

This justifies the operation of our approach in three steps:

- The structural pattern problem identification: As mentioned in the introduction, in order to tolerate structural variations of the pattern problem, we adapt an XML document retrieval approach: we consider a pattern problem as an XML query and the design as the target XML document where the pattern problem is searched. This adaptation is feasible since the transformation of UML diagrams into XML documents is straightforward and can be handled by all existing UML editors.
- The semantic correspondences of class names: it relies on linguistic and typing information to identify the classes of the pattern problem. It confirms the first step’s results and resolves any non deterministic identification when necessary.
- The identification of method declaration patterns: one characteristic of pattern problems is the repeated declaration of certain methods, for instance, by redefining abstract versions, combining abstract methods, etc.

3.1 Structural Pattern Problem Identification

For this first step of pattern problem identification, we use the same XML technique for the identification of a design pattern and which we presented in [21]. For the sake of completeness of this paper, we next briefly review it and we will illustrate it through an example in the next section.

In XML document retrieval, the document can be considered as an ordered, labeled tree. Each node of the tree represents an XML element. The tree is analyzed as a set

of paths starting from the root to a leaf. In addition, each query is examined as an extended query – that is, there can be an arbitrary number of intermediate nodes in the document for any parent-child node pair in the query. Documents that match the query structure closely by inserting fewer additional nodes are given more preference.

A simple measure of the similarity of a path c_q in a query Q and a path c_d in a document D is the following context resemblance function [12]:

$$C_R(c_q, c_d) = \begin{cases} \frac{1+|c_q|}{1+|cd|} & \text{if } c_q \text{ matches } c_d \\ 0 & \text{if } c_q \text{ does not match } c_d \end{cases}$$

where:

- $|c_q|$ and $|c_d|$ are the number of nodes in the query path and document path, respectively, and
- c_q matches c_d if and only if we can transform c_q into c_d by inserting additional nodes.

Note that the value of $C_R(c_q, c_d)$ is 1 if the two paths are identical. On the other hand, the more nodes separate the paths, the less similar they are considered, *i.e.*, the smaller their context resemblance value will be.

In XML document retrieval in general, the context resemblance function C_R is calculated based on an exact match between the names of the nodes in the query and the document paths. However, for pattern problem detection, the nodes representing the classes in the design are application domain dependent, while those in the design are generic. Thus, in our identification algorithm [21], we operate as follows:

- First, we need to calculate the resemblance values for the various matches between the class nodes in the query (pattern problem) and those in the design. In addition, for a given path, we consider that the match between the pattern problem path and the design path may not necessarily start at the root node; for this, we need to consider all possible sub-paths of the design.
- Secondly, we need to take into account: 1) the number of times a given match between two class nodes is used to calculate C_R ; and 2) the importance of each relation in the pattern.

The structural resemblance between a pattern problem and a design starts by calculating the resemblance scores between each path of the pattern problem to all the paths in the design (stored as matrix called *CRMMatrix*). In this calculation, we assume that the structural variability should be limited between the pattern problem and a potential instantiation in the design. That is, we assume that a design path may differ from a pattern problem path by adding at most N nodes compared to the longest path of the pattern problem. The larger the N , the more scattered the pattern problem would be in the design.

In addition, to account for multiple matches of a class, the *CRMMatrix* is normalized with respect to the total number of classes in the design. Thus, the final result of the structural identification algorithm is the matrix *NormalizedCRMMatrix* whose columns are the pattern problem classes and whose rows are the classes of the design. Given this matrix, we can decide upon which correspondence better represents the pattern

problem instantiation: For each pattern problem class, its corresponding design class is the one with the maximum resemblance score in the *NormalizedCRMMatrix*. There might be more than one such class. This non-deterministic correspondence could be resolved through the following step, which is the semantic correspondence of class names.

Note that in a worst case instantiation, each pattern problem class must be matched to at least one class in the design; thus, on average, the sum of the normalized resemblance scores of the matched classes should not be less than the number of classes in the pattern divided by the number of classes in the design.

3.2 Semantic Correspondences of Class Names

In order to identify different pattern problems, we have to determine the correspondences between the class names. This part differs from our previous pattern identification approach [21]. For design pattern identification, recognizing the structure and the behavior of the pattern is sufficient. However, in a pattern problem, the class names reflect semantic relationships and the presence of methods in general and in some cases the presence of the same method in different classes is very important.

Thus, we propose to determine semantic correspondences between the class names by using an adapted version of our class name comparison criteria to construct frameworks [22]. These latter express linguistic relationships between class names. We define five types of relations between classes (however, the list can be extended):

- $N_equiv(C_1, \dots, C_n)$: implies that the names are either identical or synonym, *e.g.*, Person-Person and Individual-Person.
- $N_variation(C_1, \dots, C_n)$: implies that the names are a variation of a concept, *e.g.*, employee-contractual, employee-permanent, employee-vacationer.
- $N_comp(C_1; C_2, \dots, C_n)$: implies that the name C_1 is a composite of the components C_2, \dots, C_n , *e.g.*, House- Room.
- $Gen_Spec(C_1; C_2, \dots, C_n)$: implies the name C_1 is a generalization of the specific names C_2, \dots, C_n , *e.g.*, Person-Employee.
- $Str_extension(C_1; C_2)$: implies that the name C_1 is a string extension of the name of the class C_2 , *e.g.*, XWindow- IconXWindow.

The determination of the above linguistic/semantic relations can be handled through either a dictionary (*e.g.*, Wordnet [24]), or a domain ontology when available.

3.3 Method Declaration Pattern Identification

The structural resemblance and the name comparison steps managed in the previous sections are insufficient to identify the design problems. To determine the overall resemblance, we combine these steps with method identification. In fact, in some pattern problems, the presence of methods repeated within a subset of classes in a particular pattern is essential. This last step should reinforce the quality of the identification results.

To determine the presence of methods in a design D and a pattern problem P_b , we will compare the method names for each pair of classes that were already identified as similar during the structural and semantic identification steps. First, note that,

according to our DTD, each XML path for the class methods is composed of the class name node, the XML tag “method” node and the method name. Thus, the resemblance score function is not very useful.

Instead, to compare the method declaration pattern, we proceed as follows: First, for each class C_p in P_b that is matched to a class C_d in D , we derive from the path starting at C_p a set of paths representing all possible matches with the methods in C_d ; these paths will be directly compared to all the paths in D . Secondly, the comparison results are collected into a matrix where the columns are the derived pattern problem paths and the rows are the design paths. A perfect match occurs when each column of this matrix contains at least one. A column with all zeros indicates a missing method; that is, the design lacks the redefinition of this pattern problem method and the pattern problem should therefore be reconsidered.

Note that, in addition to detecting missing re-declaration of methods, our approach tolerates the declaration of additional methods in the design.

4 Pattern Problem Identification Examples

To illustrate the steps of our approach for pattern problem identification, let us consider two examples: the bridge pattern problem and an altered version of this problem.

4.1 The Bridge Problem Case

The Bridge pattern (Figure 1) generally consists of three roles: Abstraction, Implementor, and ConcreteImplementor. The Implementor class provides an interface for the Abstraction class. Its children, ConcreteImplementor, need to implement the interface.

The Bridge pattern applies when an abstract class defines the interface to the abstraction and concrete subclasses implement it in different ways. Consider the implementation of a window abstraction in a user interface toolkit (Figure 2). This

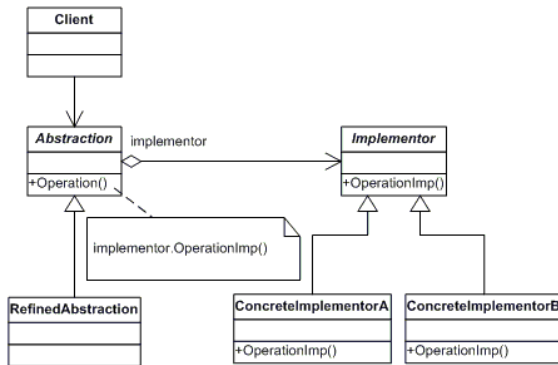


Fig. 1. The Bridge pattern[6]

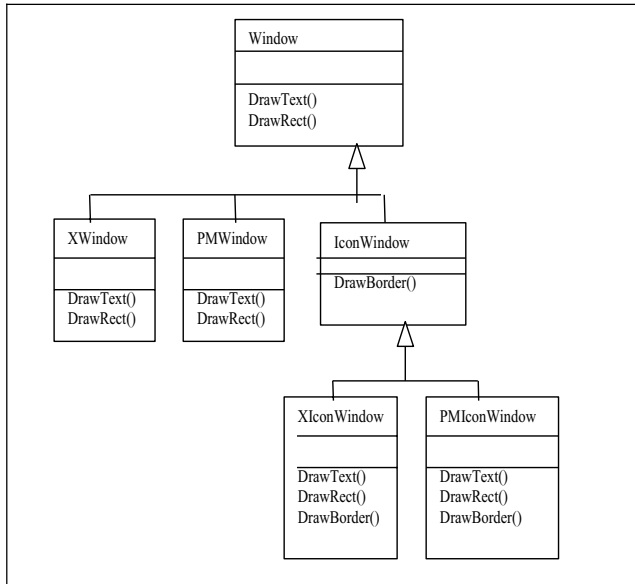


Fig. 2. A design that could be improved by the Bridge pattern

abstraction should enable us to write applications that work on both the XWindow system and IBM's presentation manager (PM), for example. Using inheritance, we could define an abstract class `Window` and subclasses `XWindow` and `PMWindow` that implement the window interface for different platforms. However, it is inconvenient to extend the window abstraction to cover different kinds of windows or new platforms. In the example, the `IconWindow` is a subclass of `Window` that specializes the window abstraction for icons. To support `IconWindows` for both platforms, we have to implement two new classes `XIconWindow` and `PMIconWindow`. The bridge pattern solves this problem by putting the window abstraction and its implementation in separate class hierarchies as shown in Figure 3.

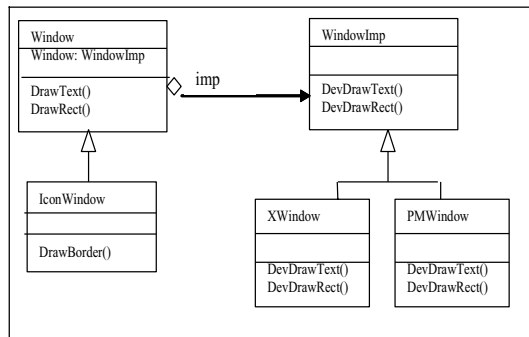


Fig. 3. The solution with the bridge pattern

Note that, at an abstract level, the Bridge pattern problem applies when : 1) there is an abstract class *A* with at least two levels of inheritance underneath it, and 2) one of the inheriting classes, say *B*, adds specific methods and has one or more inheriting classes (say *C* and *D*) redefining the methods of *A* and *B*.

An abstraction of the Bridge pattern problem is illustrated in Figure 4. We next illustrate the steps of our identification approach, which can be applied to improve the design fragment shown in Figure 2.

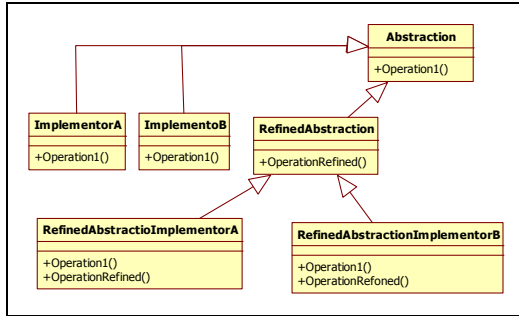


Fig. 4. Abstraction of the pattern problem

A) The structural identification step

The first step relies on the computation of resemblance function scores between the XML paths of the pattern problem and the design. The XML tree of the design is illustrated in a graphical format in Figure 5.

A sample of the context resemblance scores of the paths of the design (Figure 2) with the Bridge pattern problem paths (Figure 4) are shown in Table1. The class names are abbreviated because of space limitations.

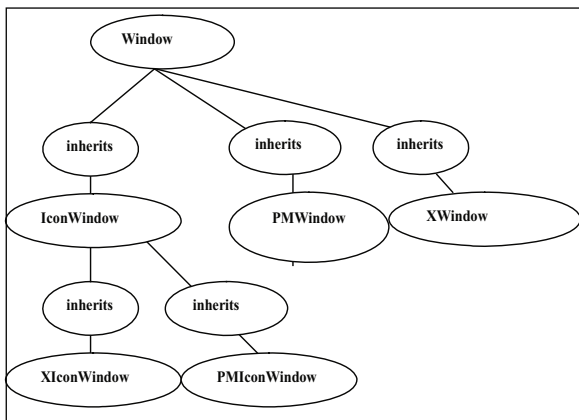


Fig. 5. XML trees for the design (Figure 2)

Table 1. Context similarity function scores

| | inherits <i>Abst</i> → <i>RefAbst</i> | inherits <i>RefAbst</i> → <i>RefAbstImpA</i> | inherits <i>RefAbst</i> → <i>RefAbstImpB</i> |
|---|--|---|---|
| <i>Window</i> inherits → <i>IconWindow</i> | $CR(c_q, c_d) = 1$ if <i>Abst</i> = <i>Window</i> <i>RefAbst</i> = <i>IconWindow</i> | $CR(c_q, c_d) = 1$ if <i>RefAbst</i> = <i>Window</i> <i>RefAbstImpA</i> = <i>IconWindow</i> | $CR(c_q, c_d) = 1$ if <i>RefAbst</i> = <i>Window</i> <i>RefAbstImpB</i> = <i>IconWindow</i> |
| <i>IconWindow</i> inherits → <i>XIconWindow</i> | $CR(c_q, c_d) = 1$ if <i>Abst</i> = <i>IconWindow</i> <i>RefAbst</i> = <i>XIconWindow</i> | $CR(c_q, c_d) = 1$ if <i>RefAbst</i> = <i>IconWindow</i> <i>RefAbstImpA</i> = <i>XIconWindow</i> | $CR(c_q, c_d) = 1$ if <i>RefAbst</i> = <i>IconWindow</i> <i>RefAbstImpB</i> = <i>XIconWindow</i> |
| <i>IconWindow</i> inherits → <i>PMIconWindow</i> | $CR(c_q, c_d) = 1$ if <i>Abst</i> = <i>IconWindow</i> <i>RefAbst</i> = <i>PMIconWindow</i> | $CR(c_q, c_d) = 1$ if <i>RefAbst</i> = <i>IconWindow</i> <i>RefAbstImpA</i> = <i>PMIconWindow</i> | $CR(c_q, c_d) = 1$ if <i>RefAbst</i> = <i>IconWindow</i> <i>RefAbstImpB</i> = <i>PMIconWindow</i> |

Once the above context resemblance scores are computed, we find the normalized similarity matrix which sums up the values of the context resemblance scores for each class in the design with respect to a class in the pattern problem and divides it by the number of classes in the design. In our example, we obtain the following normalized similarity matrix:

$$\text{Normalized CRMatrix(PatternPb, Design)} = \begin{matrix} & \begin{matrix} \text{Abst} & \text{RefAbst} & \text{RefAbstImpA} & \text{RefAbstImpB} & \text{ImpA} & \text{ImpB} \end{matrix} \\ \begin{matrix} \text{Window} \\ \text{IconWindow} \\ \text{XIconWindow} \\ \text{PMIconWindow} \\ \text{PMWindow} \\ \text{XWindow} \end{matrix} & \begin{bmatrix} 15 & 10 & 0 & 0 & 0 & 0 \\ 6 & 5 & 1 & 1 & 0 & 0 \\ 0 & 1.75 & 2.75 & 1.75 & 1.75 & 1.75 \\ 0 & 1.75 & 1.75 & 2.75 & 1.75 & 1.75 \\ 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \end{matrix} /6$$

The normalized CR matrix identifies the Bridge pattern problem and indicates that the class *Window* matches the *Abst* class, the class *IconWindow* is the *RefAbst* class, *XIconWindow* is *refAbstImpA*, *PMIconWindow* is *refAbstImpB*. However, note that the class *PMWindow* and *XWindow* were identified as either *ImpA* or *ImpB* since the resemblance score of the classes is equal to 1, while their resemblance score to *refAbstImpA* is equal to 1.75 which was already identified as *XIconWindow*. The next, semantic-based correspondence step will assist in deciding the best match and validating the determined matches.

B) Semantic correspondence

In the pattern problem abstraction, the semantic correspondence of class names uses the following relations:

- Str_extension(implementorA; RefinedAbstractionImplementorA),
- Str_extension (RefinedAbstraction; RefinedAbstractionImplementorA),
- Str_extension (implementorB; RefinedAbstractionImplementorB),
- Str_extension (RefinedAbstraction; RefinedAbstractionImplementorB)

Now, we verify that these semantic relations exist for the classes of the design that were identified by the structural step. In fact, in the design fragment (Figure 2), we noticed that the class names of the inheritance hierarchy at the second level

(i.e., *XIconWindow* and *PMIconWindow*) are composed of the class names of the hierarchy at the first level (i.e., *IconWindow* et *PMWindow*). In addition, the following semantic relations hold:

- Str_extension(*XWindow*; *XIconWindow*),
- Str_extension(*IconWindow*; *XIconWindow*),
- Str_extension (*PMWindow*; *PMIconWindow*), and
- Str_extension (*IconWindow*; *PMIconWindow*)

Thus, we can deduce that this is a Bridge pattern problem.

C) Method declaration pattern identification

The third step in our pattern problem identification examines the respect of the method presence pattern among the identified classes. That is, this step identifies if the methods are present in the design as required in the pattern problem abstraction.

In the Bridge problem abstraction, the classes *XIconWindow* and *PMIconWindow* implement the methods of the abstract class *IconWindow* and those of *XWindow* and *PMWindow*, respectively.

Due to space limitation, Table 2 shows a sample of the resemblance function scores comparing the Design of Figure 2 and the Bridge pattern problem abstraction (Figure 4).

Recall that this step uses the identification results of the previous two steps. For example, since the class *Window* was identified as *Abstraction*, then we have either *operationI=DrawText()* or *operationI=DrawRect()*. Also since *RefinedAbstraction* is *IconWindow*, then we have *Operationrefined= DrawBorder()*.

Table 2. Identification of method declaration patterns

| | <i>Abstraction</i> contains → <i>DrawText</i> | <i>Abstraction</i> contains → <i>DrawRect</i> | <i>ImplementorA</i> contains → <i>DrawText</i> | <i>ImplementorB</i> contains → <i>DrawText</i> | ... |
|--|---|---|--|--|-----|
| <i>Window</i> contains → <i>DrawText()</i> | 1 | 0 | 0 | 0 | |
| <i>Window</i> contains → <i>DrawRect()</i> | 0 | 1 | 0 | 0 | |
| <i>IconWindow</i> contains → <i>DrawBorder()</i> | 0 | 0 | 1 | 0 | |
| <i>XWindow</i> contains → <i>DrawText()</i> | 0 | 0 | 0 | 1 | |
| <i>PMWindow</i> contains → <i>DrawText()</i> | 0 | 0 | 0 | 0 | |
| <i>XWindow</i> contains → <i>DrawRect()</i> | 0 | 0 | 0 | 0 | |
| <i>PMWindow</i> contains → <i>DrawRect()</i> | 0 | 0 | 0 | 0 | |

Now, to conduct this last identification step, we substitute the method names of the problem abstraction by their potentially equivalent methods in the design (the various paths derivation step). Then, we verify that the presence of the methods in the design is conforming to the pattern problem abstraction: For a design to cover all the methods of the pattern problem, we need to find, in each column, at least one entry in this table that has the value of 1. Otherwise, the column that is missing a one indicates that the corresponding message exchange is missing in the design and the pattern problem identification is reconsidered.

On the other hand, our approach tolerates additional, methods in the design. In our example, we tolerated the additional method `DrawRect()`.

At the end of the method declaration pattern identification step, the designer has more confidence in the results of the structural and semantic name identification steps.

4.2 Identification of an Altered Bridge Pattern Problem

In this section, we show how our approach (unlike others) can detect altered/similar pattern problems. Let us consider the altered version of the Bridge pattern problem as abstracted in Figure 4; and let us consider the design fragment of Figure 6 to analyze. In fact, the classes *Emptystyle*, *WithIconstyle* and *Applicativestyle* are added, yet it remains an instance of the Bridge pattern problem. In addition, we omitted the methods in Figure 6 and we illustrate the pattern problem identification using the structural determination and the semantic correspondences of class names.

After collecting the context resemblance scores, the normalized similarity matrix is computed.

| | Abst | RefAbst | RefAbstImpA | RefAbstImpB | ImpA | ImpB |
|--------------------|-------|---------|-------------|-------------|------|------|
| Window | 26.75 | 12.5 | 0 | 0 | 0 | 0 |
| EmptyWindow | 11.5 | 8 | 1 | 1 | 1 | 1 |
| WindowwithIcon | 11.5 | 8 | 1 | 1 | 1 | 1 |
| ApplicativeWindow | 11.5 | 8 | 1 | 1 | 1 | 1 |
| EmptyStyle | 6 | 8 | 2.75 | 2.75 | 1.75 | 1.75 |
| WithIconStyle | 6 | 8 | 2.75 | 2.75 | 1.75 | 1.75 |
| ApplicativeStyle | 6 | 8 | 2.75 | 2.75 | 1.75 | 1.75 |
| XWindowEmpty | 0 | 1.75 | 2.75 | 2.75 | 1.75 | 1.75 |
| PMEmpty | 0 | 1.75 | 2.75 | 2.75 | 1.75 | 1.75 |
| XWindowWithIcon | 0 | 1.75 | 2.75 | 2.75 | 1.75 | 1.75 |
| PMWithIcon | 0 | 1.75 | 2.75 | 2.75 | 1.75 | 1.75 |
| XWindowApplicative | 0 | 1.75 | 2.75 | 2.75 | 1.75 | 1.75 |
| PMApplicative | 0 | 1.75 | 2.75 | 2.75 | 1.75 | 1.75 |

NormalizedCRMatrix(PatternPb, Design) =

We noticed in the normalized *CRMatrix* that the class *Abstraction* matches *Window* since it has the most important score (26.75). Moreover, the match score of the classes *EmptyWindow*, *WindowWithIcon*, *ApplicativeWindow*, *EmptyStyle*, *WithIconStyle*, *ApplicativeStyle* to *RefAbst* is equal to (8), while the score matching these classes to *Abstraction* is 11.5; however these six classes are identified as *RefAbst*, since *Window* has been already identified as *Abstraction* with a greater matching score. Finally, the classes *PMEmpty*, *PMWIcon*, *PMApplicative* are identified as *RefAbstImpB* and the classes *XwindowEmpty*, *XWindowWIcon*, *XWindowApplicative* are identified as *RefAbstImpA*.

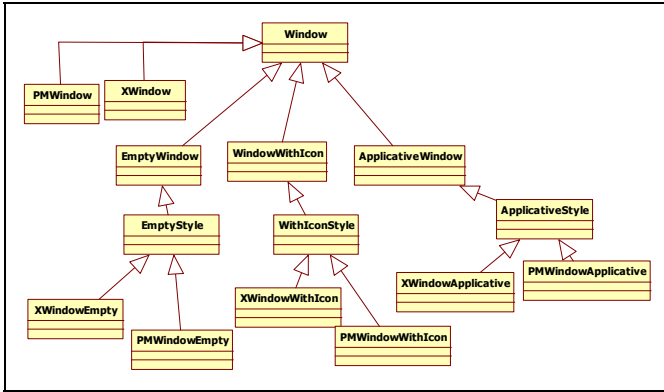


Fig. 6. Altered Bridge pattern problem

The results of the structural determination will be confirmed by the semantic correspondence step. For the semantic correspondence of class names, we remark that the following semantic relations hold:

- Str_extension(EmptyWindow;XWindowEmpty),
- Str_extension (EmptyWindow;PMWindowEmpty)
- Str_extension(WindowWithIcon;XWindowWithIcon)
- Str_extension (WindowWithIcon;PMWindowWithIcon)
- Str_extension(ApplicativeWindow;PMWindowApplicative) and
- Str_extension(ApplicativeWindow;XWindowApplicative)

Thus, the class names of the inheritance hierarchy at the fourth level (*i.e.*, XWindowEmpty) are *composed* of the class names of the hierarchy at the second level (*i.e.*, EmptyWindow, XWindow). As a conclusion, this step confirms that the design is also a representation of the Bridge pattern problem.

Note that, even though the above examples use the inheritance relation, our approach works in the same way for all types of UML class relations. The reader is referred to [21] where the Composite pattern is illustrated with the composition relation.

6 Conclusion

The paper first overviewed existing works for pattern problem detection. Secondly, it presented a new approach based on a structural identification followed by a semantic identification of class names (based on linguistic correspondences) and method presence pattern verification. This approach is applicable for all patterns and pattern problems that can be modeled through UML class diagrams or any UML profile language whose class diagrams can be stored as XML documents.

The paper illustrated the proposed approach through the Bridge pattern problem. In addition, to emphasize one advantage of our approach, it was applied to show its capacity to detect also an altered version of the Bridge pattern problem. The illustrated examples were automatically treated through our prototype toolset.

The current version of our structural identification algorithm runs in an exponential time in terms of the number of relations among the classes of the design. One practical way to manage this complexity is to decompose a large design and treat it one fragment at a time; a simple heuristic decomposition strategy is to consider the connected components of the class diagram, starting from an abstract class.

We are examining how to add more intelligence in the structural identification algorithm: how to alleviate the analysis of paths by adding priorities, and how to reduce the number of possible method correspondences in the last step.

References

1. Bouhours, C., Leblanc, H., Percebois, C.: Structural variants detection for design pattern instantiation. In: 1st International Workshop on Design Pattern Detection for Reverse Engineering, Benevento, Italy (October 2006)
2. Bouhours, C., Leblanc, H., Percebois, C.: Bad smells in design and design patterns. *Journal of Object Technology* 8(3) (May-June 2009)
3. Kampfmeier, H., Zschaler, S.: Finding the Pattern You Need: The Design Pattern Intent Ontology. In: Engels, G., Opdyke, B., Schmidt, D.C., Weil, F. (eds.) *MODELS 2007*. LNCS, vol. 4735, pp. 211–225. Springer, Heidelberg (2007)
4. Moha, N., Guéhéneuc, Y.G., Leduc, P.: Automatic generation of detection algorithms for design defects. In: Uchitel, S., Easterbrook, S. (eds.) *Proceedings of the 21st Conference on Automated Software Engineering*, September 2006, pp. 297–300. IEEE Computer Society Press, Los Alamitos (2006)
5. Moha, N., Guéhéneuc, Y.-G., Le Meur, A.-F., Duchien, L.: A domain analysis to specify design defects and generate detection algorithms. In: Fiadeiro, J.L., Inverardi, P. (eds.) *FASE 2008*. LNCS, vol. 4961, pp. 276–291. Springer, Heidelberg (2008)
6. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: *Design patterns: Elements of reusable Object Oriented Software*. Addison-Wesley, Reading (1995)
7. Florijin, G., Meijers, M., Van Winsen, P.: Tool support for object oriented patterns. In: Aksit, M., Matsuoka, S. (eds.) *ECOOP 1997*. LNCS, vol. 1241, pp. 472–495. Springer, Heidelberg (1997)
8. Bergenti, F., Poggi, A.: Improving UML design pattern detection. In: *Proceedings of the 12th International Conference on Software Engineering and Knowledge Engineering SEKE (2000)*
9. Albin Amiot, H., Cointe, P., Guéhéneuc, Y.G.: Un meta-modele pour coupler application et détection des design patterns. *L'objet* 8, 1–18 (2002)
10. El Boussaidi, G., Mili, H.: Detecting patterns of poor design solutions by using constraint propagation. In: *MODELS, Proceedings of the 11th International Conference on Model Driven Engineering Languages and Systems (September 2008)*
11. Pagel, B.U., Winter, M.: Towards pattern-based tools. In: *Proceedings of EuropLop (1996)*
12. Brown, K.: Design reverse-engineering and automated design pattern detection in Smalltalk. Technical Report TR- 96-07, University of Illinois at Urbana-Champaign (1996)
13. Tsantalis, N., Chatzigeorgiou, A., Stephanides, G., Halkidis, S.T.: Design pattern detection using similarity scoring. *IEEE Transactions on Software Engineering* 32(11) (2006)

14. Lee, H., Youn, H., Lee, E.: A design pattern detection technique that aids reverse engineering. *International Journal of Security and Applications* 2(1) (January 2008)
15. Mili, H., El Boussaidi, G., Salah, A.: Représentation et Mise en oeuvre de patrons de conception par représentation explicite des problèmes. In: *LMO, Suisse* (2005)
16. Dong, J., Sun, Y., Zhao, Y.: Design pattern detection by template matching. In: *SAC 2008, Ceara, Brazil, March 16-20* (2008)
17. XML Metadata Interchange: *OMG Document ad/98-07-03* (July 6, 1998)
18. Manning, C.D., Raghavan, P., Schütze, H.: *An introduction to information retrieval*. Cambridge University Press, England (2008)
19. Alikacem, E., Sahraoui, H.A.: Détection d'anomalies utilisant un langage de description règle de qualité. In: *Rousseau, R., Urtado, C., Vauttier, S. (eds.) LMO 2006*, pp. 185–200 (2006)
20. Ciupke, O.: Automatic Detection of Design Problems in Object-Oriented Reengineering. In: *TOOLS 30*, pp. 18–32. IEEE Computer Society Press, Los Alamitos (1999)
21. Bouassida, N., Ben-Abdallah, H.: Structural and behavioral detection of design patterns. In: *International Conference on Advanced Software Engineering & Its Applications (ASEA)*, Jeju Island, Korea, December 10-12. LNCS Proceedings. Springer, Heidelberg (2009)
22. Bouassida, N., Ben-Abdallah, H., Gargouri, F.: Stepwise framework design by application unification. In: *IEEE International Conference on system man and Cybernetics, Tunisia* (2003)
23. <http://www.omg.org/mda/>
24. <http://wordnet.princeton.edu/>

Comparing Safety Analysis Based on Sequence Diagrams and Textual Use Cases

Tor Stålhane¹, Guttorm Sindre¹, and Lydie du Bousquet²

¹ Dept of Computer and Info. Science, Norwegian Univ. of Sci. and Tech (NTNU)

² Laboratoire d'Informatique de Grenoble (LIG)

{stalhane, guttors}@idi.ntnu.no, lydie.du-bousquet@imag.fr

Abstract. Safety is of growing importance for information systems due to increased integration with embedded systems. Discovering potential hazards as early as possible in the development is key to avoid costly redesign later. This implies that hazards should be identified based on the requirements, and it is then useful to compare various specification techniques to find out the strengths and weaknesses of each with respect to finding and documenting hazards. This paper reports on two experiments in hazards identification – one experiment based on textual use cases and one based on systems sequence diagrams. The comparison of the experimental results reveal that use cases are better for identifying hazards related to the operation of the system while system sequence diagrams are better for the identification of hazards related to the system itself. The combination of these two techniques is therefore likely to uncover more hazards than one technique alone.

Keywords: safety analysis, sequence diagrams, misuse cases, experiment.

1 Introduction

Systems safety is concerned with the avoidance or mitigation of harmful accidents, as opposed to security which addresses malicious attacks against the system. Safety has traditionally been more of a concern for embedded systems than for information systems. However, increasing convergence and integration between different kinds of systems, as well as trends towards ubiquitous information systems and “internet of things” [1] means that the border between information systems and embedded systems is getting more blurred. An increasing number of safety-critical systems are to a large degree realized in software. Thus, safety analysis of software gets more and more important. Safety analysis needs to include a wide variety of participants – not just the safety analysis specialists. Two important groups that always need to be involved are the customers with their domain knowledge and the developers who have a firm understanding of all the decisions needed to develop a software system.

As a consequence of this, the safety analysis must apply methods and documents that are easy to understand for everybody involved. In our opinion, both sequence diagrams and textual use cases belong to this category. It might, however, not be without consequences to choose one over the other. It might be the case that one of

these methods is uniformly better than the other or they may each have their strong and weak sides. In any case, one or more experiments are needed in order to see how these two methods behave when used as the starting point for a safety analysis. In order to shed some light over this, we have run an experiment partly in Grenoble – system sequence diagrams – and partly at NTNU – textual use cases.

The rest of this paper is structured as follows: Section 2 goes through related work. Section 3 presents the two safety analysis techniques to be compared. Section 4 describes the experimental design and section 5 the analysis of the results. Section 6 discusses threats to validity, whereupon section 7 concludes the paper.

2 Related Work

As far as we have found out, there are few experiments comparing textual (mis)use cases and sequence diagrams. This is no surprise, since there are few controlled experiments about use cases to be found in the literature at all. There are experiments comparing the effectiveness of representation languages, for instance ER or EER diagrams with relational tuples [2]. However, the two forms of representation compared contained the same information but were on different life-cycle levels, ER being an analysis stage language, while relational tuples belong to the design stage. The relationship between use case diagrams and systems sequence diagrams is different: both are requirements stage representations but the textual use case and the systems sequence diagram have different focus. The textual use case focuses on the exchange of messages between system and user, while the systems sequence diagram also includes the exchange of messages between system components.

An experiment more directly targeting the comparison of diagrams and tables and textual representations is [3], confirming claims about advantages of diagrams previously made analytically in [4]. Experiments have also found diagrams superior for presenting procedural instructions [5], which is closely related to conceptual modeling. More generally, the empirical evidence is not so consistently in favor of diagrams. As reported in [6], some experimental studies have shown advantages for diagrams, but others have shown advantages for tables, or given mixed results. One particular study also showed the outcome to be dependent on the subject group, business students solving problems more effectively with tables while engineering students solved problems more effectively with graphs [7].

There are several papers published on the use of UML diagrams as a basis for safety analysis. A good example is [8], where Guiochet and Vilchis use the full range of UML diagrams – use case diagrams, sequence diagrams, class diagrams, state diagrams and deployment diagrams. In [9] Long and Jinglun use event sequence charts for safety analysis. C. Ren also uses several UML diagrams – among them the sequence diagram – for safety inspection of mine equipment [10]. Unfortunately, none of authors have performed experiments. All three papers, and others related to the same area, are case studies to see if the chosen diagrams can be used in a safety analysis. Similarly, Allenby and Kelly propose a technique for safety analysis based on scenarios [11].

A more comparative work is done by T.A. Alspaugh et al [12]. Their focus is unfortunately not safety problems but defects. What makes their paper interesting is that they compare the ease of identifying defects using use cases, sequence diagram and scenarios formulated in ScenarioML. Their conclusion is clear – scenarios are better than both sequence diagrams and use cases. Their experiment had only four participants and no statistically significant results were thus obtained. On the other hand, they ran a series of interviews with the participants and thus gained a lot of insight into the reasons behind the results. The reason why use cases did not work well was that “the numbering scheme for the steps can be confusing”. For the sequenced diagrams, some of the participants had problems with the notation – especially branches and options. On the other hand, the participants had no problems with the ScenarioML documents which they found easy to read and understand.

In addition to the safety and reliability related experiments cited above, there has been a lot of research on other facets of use cases such as authoring – see for instance the well-known work of Achour et al [21] and the follow-up work of Cox et al. [22].

3 The Techniques to Be Compared

3.1 Textual Use Cases – TUC

TUC can be used to identify possible failure modes, using the textual misuse case analysis. The misuse case was originally proposed for collecting security requirements. They can, however, also be used for safety analysis where focus is on accidents causing harm to the system and its environment. A simple example is an automated system used to keep the water level in a tank constant while delivering steam to an industrial process. This is done by filling the tank through one valve and emptying it through another valve when needed. If the pressure in the tank exceeds the critical pressure pre-set by the operator, a relief valve should open automatically. The operator may also manually empty the tank (for instance if the relief valve fails to work when the pressure becomes too high) or manually fill the tank (if the automatic adjustment of water level does not work).

A misuse case such as “Set too high pressure” may have a “threatens” relationship to one or more use cases – in this case “Set critical pressure”. It is also possible that one misuse case may “aggravate” the effect of another, or that a use case may have a “mitigates” relationship to that misuse case. An example of a textual representation is shown in Table 1. Here, threats corresponding to misuse cases can be added in a third column called “Threats”, and mitigations can be added in a fourth column. An essential idea of misuse case analysis is that the representation format causes only a limited overhead if use cases are already applied in a project, which is often the case in mainstream software projects. In addition, the informality and simplicity of the technique makes it suitable for supporting brainstorming on threats at an early stage of development. For a more complete coverage of misuse case analysis, the reader is referred to [13] and [14].

Table 1. Column format use case with safety issues for the boiler

| Use case name | “Empty tank manually” | | |
|--|---|--|--|
| User actions | System response | Threats | Mitigations |
| | System alarms operator of high Pressure | System fails to raise alarm; Operator fails to notice alarm | 2 independent alarms; Use both sound and blinking lights |
| Operator issues command to empty tank | | Operator fails to react (e.g., incapacitated?) Operator gives wrong command, e.g., filling tank | Alarm backup operator; Auto sanity check, disallow filling at high Pressure |
| | System opens valve to sewer | System fails to relay command to valve; Valve is stuck | |
| Operator reads pressure | | Operator misreads and stops tank emptying too soon | Maintain alarm blinking until situation normal |
| | Pressure returns to normal | This is not achieved, see exceptions | |
| Operator stops tank emptying and logs the event. This ends the use case. | | | |
| Exceptional paths | | | |
| | Opening valve is insufficient to normalize pressure | | |
| Operator issues command to reduce temperature | | Operator gives wrong command, e.g., increase temperature | Automatic sanity check, disallow temp increase at high pressure |
| | Pressure returns to normal | | |
| Operator logs the event. This ends the use case. | | | |

3.2 System Sequence Diagrams - SSD

The sequence diagram is an UML diagram used to show the dynamic properties of a system. A sequence diagram shows the messages that are passed between two components in a system. In order to be a strict sequence diagram, these two parts have to be objects – instances of classes. In the early phases of system development, however, the components used in the diagram can be e.g. GUI, database servers, internets or subsystems. Sequence diagrams applied in this way are called system sequence diagrams or SSD [15]. Even though they have no formal standing in the UML 2.0 definition they are widely used in the industry. One of the factors that make them both useful in an industrial context and important for our research on safety analysis is that the SSDs, just as the textual use cases, can be constructed early in the development process.

An SSD is close to a textual use case and can be constructed using the same structure and words as a textual use case. The SSD has, however, one advantage over the textual use cases. With the advent of UML 2.0 [16], sequence diagrams in general and thus also the SSDs can contain alternatives – if...then...else... (alt or opt) and loops – for... (loop). These constructs enable us to insert info into an SSD that it might be difficult to include in a textual use case. The diagram below contains the same info as the use case in table 1 except for the last two rows.

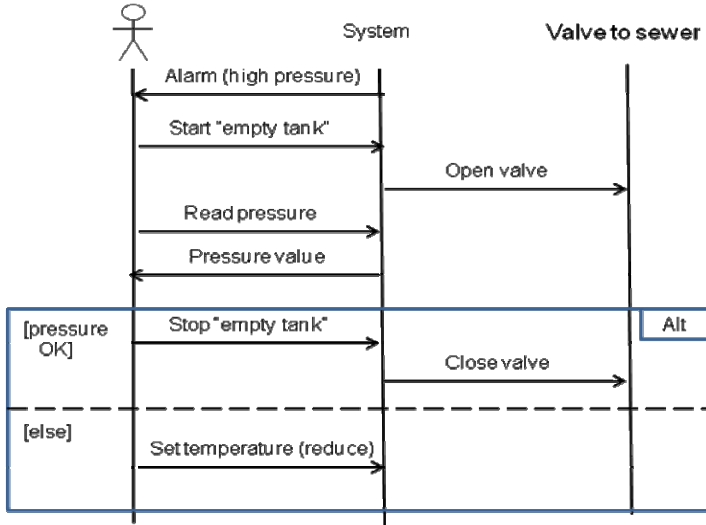


Fig. 1. SSD for the function shown in table 1

The SSD can be used for safety analysis by going through each message sent or received and ask questions like “what will happen if the message is not sent” and “what will happen if the message has the wrong content”?

4 The Experiment

4.1 Research Approach

Both our previous experiments have focused on the question “Is representation A better than representation B for the identification of failure modes?” In both cases it turned out that the answer was not a simple yes or no. Even though we could conclude that in general representation A was better than representation B, there were still areas where representation B was better and areas where they were equal. Thus, we will look at both whether textual use cases are better than sequence diagrams and, if this is the case, are there still areas where sequence diagrams are better? We will try to throw some light over this by answering the following research questions:

- RQ1: Is TUC in general better than SSD for identifying failure modes?
- RQ2: If we conclude in RQ1 that one of the representations in general is better than the other, are there still any problem areas where this does not hold?

- RQ3: If we conclude in RQ2 that in general representation A is better than representation B, then:
 - RQ3-1: Are there still failure modes where representation B is better than representation A?
 - RQ3-2: What can be the reasons for this?

The two first of these research questions can be reformulated as hypotheses. In order to test these hypotheses, we will use the t-test for RQ1 and proportion test for RQ2. The reasons for these choices are that for RQ1 we need to compare the number of failure modes identified while for RQ2 we need to compare the proportion of participants that have identified a single failure mode.

To use the proportion test, we define the following parameters:

- x_i is the number of successes observed in sample i .
- π_i is the portion of success events in sample i .
- n_i is the size of sample i .
- Δ is the difference that we want to test for. In our case Δ is set to zero.
- π is the portion of successes in both samples viewed together – see equation (1).

$$\pi = \frac{x_1 + x_2}{n_1 + n_2} \quad (1)$$

The proportion test is based on the following statistics, where z is assumed to be normally distributed (0, 1):

$$z = \frac{\pi_1 - \pi_2 - \Delta}{\sqrt{\pi(1-\pi)\left(\frac{1}{n_1} + \frac{1}{n_2}\right)}} \quad (2)$$

RQ3-1 is answered by studying the results from the t-tests used to answer RQ2. However, when trying to answer RQ3-2 we will use the areas defined in our experiment coding as a starting point. RQ3-2 cannot be answered using any statistical approach since the focus is why – not what. Assuming that representation A in general is better than representation B, we will do as follows:

1. For each failure mode where representation B is better than representation A, write down the characteristics of the failure mode that can be related to both representations.
2. Identify what it is in representation B that makes it easy to discovery the failure mode and what it is in representation A that makes it difficult to discover.
3. If we are able to identify a convincing set of factors in the representations that respectively hide and reveal the failure mode under consideration, we will claim that the reason why the failure mode was found by the participants using representation B and not by the participants using representation A stem from these factors.

4.2 Experiment Design

4.2.1 The Preparation

The experiment using SSDs were done at the University J. Fourier - Grenoble I (UJF), France by the Laboratoire d'Informatique of Grenoble (LIG) and the experiments on

TUC were done in Trondheim at the department of Information and Computer Science at NTNU. Since the participants in the two sites had different background, the preparation phase for the two experiments differed while the rest of the experiment design was the same at both sites.

At UJF, the participants were in the first year of a Master in Computer Sciences, where courseware is offered in English¹. Students were gathered in an auditorium, where they received a short introduction on safety issues and safety analysis. An example of safety analysis was illustrated. This part lasts 1 hour and 30 minutes. Then participants were asked to fill in the pre-experiment questionnaire, read the document and fill in the post-experiment questionnaire. They then had 1 hour and 30 minutes to do it but most of the participants left the room before the end of the period...

At NTNU, the participants were gathered in an auditorium where they receive a short introduction to safety analysis and why it was important to be able to do this at an early stage. They then had 15 minutes to refresh their knowledge of textual use cases and misuse cases before filling in the pre-experiment questionnaire.

When the pre-experiment questionnaire was finished the participants worked on their task at identifying failure modes based on the textual use cases and a system diagram, showing the main components of the train control system and how the components – radio, operators, trains and track side equipment – cooperated in order to run the railway in a safe and efficient manner.

For both experiments, each participant documented the identified failure modes in free text on the sheet of paper where the TUC or SSD was shown.

4.2.2 The Coding Process

The researchers read the failure mode descriptions and related it to one of the failure modes used in the coding schema. If several failure modes could apply, the researchers reached a common understanding of the category that was more close to the participant's intentions. In order to reduce variation in the application of the coding schema, all the forms turned in by the experiment participants were coded by the same researcher and controlled by another. The coding schema was designed in advanced, based on an analysis of the failure mode categories used by the respondents in pervious experiments. The schema has three levels and the top level contains the following categories:

- OP – Operator problems
- CS – Computer system problems
- EP – Engineer problems
- MP – Maintenance personnel problems
- TC – Technical communication problems
- TE – Track-side equipment problems

Each of these categories has two or more subcategories. The subcategories also have a set of sub-subcategories, which is not shown here. This coding schema has 104 failure modes and gave us a flexible way to categorise the experiment results. The subcategories for operator problems – OP – are shown below:

¹ <http://mosig.imag.fr/MainEn/HomePage>

- OP100 Incoming messages
- OP200 Operator action
- OP300 Operator scheduling
- OP400 Operator equipment problems
- OP500 Operator knowledge
- OP600 Operator overload, e.g. due to panic
- OP700 Communication problems

The responses were registered in an Excel worksheet – each failure mode category at the lowest level has a line in the worksheet and each participant has a column. The entries for each participant are binary – found a certain failure mode or not. I.e. there is no “bonus” for identifying the same failure mode several times.

5 Experiment Results and Analysis

The best participant in the TUC experiment found 25 failure modes while the one that did worst found only 8. Thus, the identification probability ranges from 24% to 8%. For those who used the SSD, the best participant found 19 failure modes while the one who did worst found only 4. In percentages this gives us a range of 18% to less than 4%.

In order to test whether a difference between two representations is important, we need to look at the significance level α and the effect size ES. Even though the significance level – the p-values – are below 0.05, the difference may still be uninteresting if the effect size is low. Thus, we will base our discussions both on the significance level and the effect size. Let t be the value of the t statistics and df the number of degrees of freedom. We will use the following expression to calculate the effect size ES:

$$ES = \frac{2t}{\sqrt{df}} \quad (3)$$

RQ1 is answered using the t-test. The result is as follows:

Table 2. t-test for total number of failure modes identified

| t-Test: Two-Sample Assuming Unequal Variances | | |
|---|------------|------------|
| | <i>TUC</i> | <i>SSD</i> |
| Mean | 15.07 | 11.70 |
| Variance | 10.92 | 24.46 |
| Observations | 29.00 | 10.00 |
| Hypothesized Mean Difference | 0.00 | |
| Df | 12.00 | |
| t Stat | 2.01 | |
| P(T<=t) one-tail | 0.03 | |
| t Critical one-tail | 1.78 | |
| P(T<=t) two-tail | 0.07 | |
| t Critical two-tail | 2.18 | |

Based on this, the answer to RQ1 is that the experiment participants using TUCs found significantly more failure modes than those that used SSDs. The effect size is 1.16 which indicates a large effect.

Of the 104 failure modes, 70 were related to human behaviour, 28 were related to the technical systems and six were related to operational conditions. If we look at each category separately we find that the difference is significant only for the human related failure modes with a p-value less than 0.01 and an effect size of 1.1. For the two other categories, we find no significant differences.

To answer RQ2, we first need to consider the differences in proportion of experiment participants that identify the failure modes for each main area.

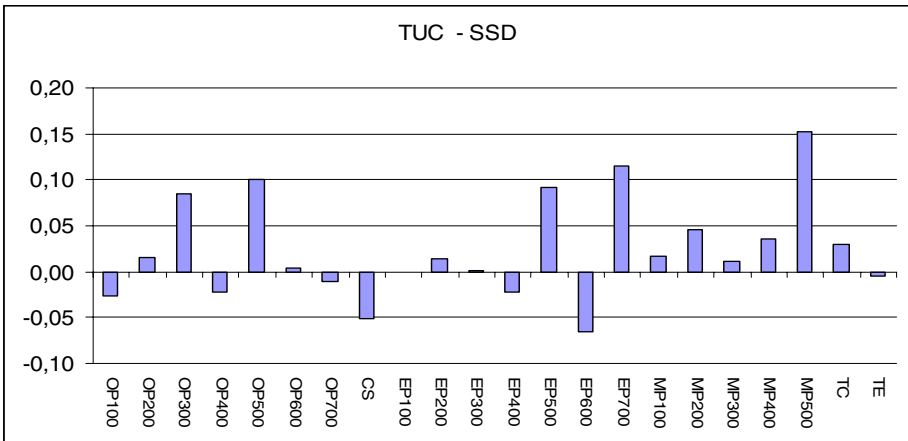


Fig. 2. Frequency differences between TUC and SSD

Our answer to RQ2 is that everywhere where there is a statistically significant difference between the areas, failure mode identification based on textual use cases is better or just as good as the one based on SSD with $\alpha = 0.10$. TUC is significantly better than SSD for the areas listed below:

Table 3. t-test for number of failure modes identified for each main area

| Failure mode | p-value | ES |
|--|---------|------|
| OP 300 Operator scheduling | 0.02 | 0.88 |
| OP 500 Operator knowledge | 0.02 | 0.90 |
| EP 500 Engineer knowledge | 0.04 | 0.92 |
| EP 700 Communication problems | 0.02 | 1.15 |
| MP 200 Maintenance action | 0.01 | 1.01 |
| MP 500 Maintenance personnel knowledge | 0.00 | 1.22 |

We see that all the effect sizes are moderate to large. Note that a large part of the areas where TUC is significantly better than SSD are related to knowledge and actions – factors that relate to how personnel use the system.

In order to get a clearer picture of the experimental results related to RQ2, we need to look at the results for each single failure mode. These results are shown in table 4 below, while the plot of individual failure modes shows the more detailed picture shown in figure 3. We see that half of the frequency effect sizes are small – 0.10 and that roughly a third are moderate to large – 0.30 to 0.50.

Table 4. Proportion tests for each failure mode

| Failure mode | p value | Δ_{freq} |
|--|---------|-----------------|
| OP 204 Wrong ack | 0.067 | 0.10 |
| OP 212 Do not save changes to schedule etc. | 0.002 | 0.24 |
| OP 302 Wrong train scheduling * | 0.007 | 0.42 |
| OP 501 Wrong situation analysis * | 0.013 | 0.39 |
| OP 503 Wrong interpretation of system’s functionality | 0.067 | 0.10 |
| CS 100 Does not save or deletes info | 0.006 | -0.46 |
| EP 209 Enters wrong info to operator | 0.094 | 0.29 |
| EP 504 Lack of training * | 0.006 | 0.21 |
| EP 700 Communication problems * | 0.067 | 0.10 |
| MP 209 Enters wrong info * | 0.003 | 0.46 |
| MP 212 Do not save changes to schedule etc. * | 0.067 | 0.10 |
| MP 401 Lack of person-to-person trust or personnel conflicts | 0.067 | 0.10 |
| MP 402 Problems with telephone or radio | 0.067 | 0.10 |
| MP 501 Wrong situation analysis * | 0.000 | 0.49 |
| MP 504 Lack of training * | 0.031 | 0.14 |
| TC 400 Other technical communication problems | 0.067 | 0.10 |

In order to answer RQ3, we start with the table above. As we see, the only failure mode where SSD is significantly better than TUC is for CS100 – Does not save or delete info. Thus, the answer to RQ3-1 is that for CS100, SSD is doing significantly better than TUC. In addition we see that the frequency difference is 0.46, indicating a large effect.

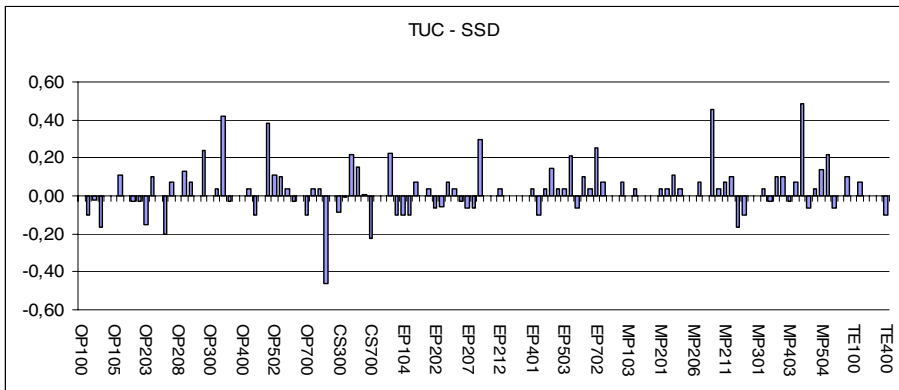


Fig. 3. Frequency differences for all failure modes

An alternative way to look at the results of the experiment is to consider all failure modes that have been identified by more than 50% of the participants in each group. This gives us the results shown in table 5.

Table 5. Top ten failure modes identified for each representation

| | Percentage of participants | | |
|-------|----------------------------|-------|-------|
| | TUC | SSD | |
| OP209 | 82.76 | 80.00 | CS400 |
| CS400 | 79.31 | 70.00 | OP209 |
| EP209 | 79.31 | 60.00 | CS100 |
| EP102 | 72.41 | 60.00 | CS700 |
| EP702 | 65.52 | 50.00 | OP203 |
| MP209 | 65.52 | 50.00 | CS300 |
| OP302 | 62.07 | 50.00 | EP102 |
| OP501 | 58.62 | 50.00 | EP209 |
| MP501 | 58.62 | | |

The table shows that for TUC there is only one computer system (CS) failure mode among the top 10 failure modes, while we have four CS failure modes for the SSD. These failure modes are:

- CS100 Does not save or deletes info
- CS300 Reacts wrongly to command or do not react at all
- CS400 Shows wrong info, including false alarms – both TUC and SSD
- CS700 Other software errors

For all of these failure modes SSD is better than TUC although the significance level and frequency differences are small, except for CS700 where we have a difference of -0.22. The p-value, however, is 0.22. We should also note that while there is 11% computer system related failure modes among the top ten found for TUC, the corresponding value for SSD is 50%. The same effect has been observed in previous experiments. We have found that FMEA outperform misuse case diagrams [17] when it comes to network problems and that misuse cases based on both diagrams and text score low on the failure modes “software unavailable”, “network down” and “delete files” [18]. Thus, as a general conclusion, we will claim that TUC is better than SSD for all failure modes except those pertaining to the inner working of the computer system.

We will use the process suggested for RQ3-2 in chapter 4.1 in order to understand why we get this result.

1. Characteristics of CS failure modes: The main characteristic of a CS failure mode is that it describes computer system failure modes. The user and his behaviour are not involved.

2. What in SSD helps us to discover CS failure modes and what in TUC hinders this discovery? The answer to this question is that SSD makes explicit the sending and receiving of messages – e.g. commands – between actors and one or more subsystems. The TUC, on the other hand has all its focus on the user and how the system reacts to his commands. Sending and receiving messages is only included indirectly.
3. Based on this, we will conclude that TUC helps the analyst to focus on user / system interactions, while SSD helps the user to focus on how the user / system interactions are performed – e.g. via sending and receiving messages.

6 Threats to Validity

We will use the categories defined in [19] as a starting point for our discussion on threats to validity. We will look at each threat in a short section before giving a sum-up of our validity claims.

6.1 Conclusion Validity

Conclusion validity is concerned with our ability to draw the right conclusions about the relationship between the treatment and the outcome. An important question here is sample size. For the main failure mode categories, we have seen a medium to large effect – ES values from 0.88 to 1.22.

If we denote the type I error probability by α and the type II error probability by β , and let N be the sample size, the following relationship holds:

$$N = \frac{4(u_{\alpha/2} + u_{\beta})^2}{ES^2} \tag{4}$$

If we use $\alpha = 0.05$ and $\beta = 0.20$, we get for smallest effects $N = 26/0.9^2$ which gives use an N-value of 32. Since we have a total of 39 participants, we have a sufficient number of observations for our conclusion.

For proportions, a large effect size is 0.40 or more. The only single failure mode which departs from the general picture – TUC better than SSD – is failure mode CS100. In this case the ES is large – 0.46. If we use the same notation as in equation (4) except that N now is the sample size of each part of the experiment, we have for proportions that:

$$N = (u_{\alpha/2} + u_{\beta})^2 \frac{p_1(1 - p_1) + p_2(1 - p_2)}{d^2} + \frac{2}{d} + 2 \tag{5}$$

For CS100 we have $p_1 = 0.60$ and $p_2 = 0.14$ and thus $d = 0.46$. If we, as before use $\alpha = 0.05$ and $\beta = 0.20$, we get $N = 19.7$ Thus, we need 40 observations in order to observe the difference for CS100. We have 39 observations, which is so close to 40 that we will accept the result as valid. We only have to increase the risks of a type II error from 0.20 to 0.21 in order to reduce the number of observations necessary to 39.

6.2 Internal validity

Internal validity is concerned with the relationship between treatment and outcome – was it the treatment that caused the outcome?

The main discussion point is whether the two groups of experiment participants are equal. In order to answer this question we will look at results from the pre-experiment questionnaire. Out of the eight questions in pre-experiment questionnaire, there are significant differences between the two groups in the answers to only three of them. The two groups score the same for knowledge of sequence diagrams, analysis of safety, reliability and misuse cases plus the number of months of practical experience with IT work.

The three questions where the two groups differ are questions PQ1, where the SSD group has less experience with class diagrams, PQ5, where the SSD group has less experience with writing use cases and PQ7, where the SSD group on the average has completed six semesters of their studies while the TUC has completed only 4.8. The important difference, however, is the difference between the two groups' knowledge of their respective methods – the SSD group's knowledge of sequence diagrams and the TUC group's knowledge of use cases. On a Likert scale from 1 (no knowledge) to 5 (good knowledge) the SSD group has an average score of 2.1 for sequence diagram experience while the TUC group has an average score of 1.6 for use cases. The difference is significant at the 0.05 level. The effect size, however, is just 0.5 – a small to moderate effect. As an aside, we know there are purists who do not like to use arithmetic on Likert scale values. We will not take up this discussion here but point the reader to John W. Tukey's article [20] on this topic.

Besides the factors mentioned above, cultural differences may have influenced the results. There is, however, no way that we can assess the size of this influence.

6.3 Construct Validity

Construct validity is concerned with the relationship between theory and observations. The theory is in our case about the relationship between the number of failure modes identified and the documents used in this process – TUC or SSD. Counting the number of failure modes enables us to observe any differences in the effect of the two ways to represent the situation – TUC or SSD. Since we have already shown that there are no or just small differences between the participants' capabilities, we will claim that the observed effect stems from the different representations.

6.4 External Validity

External validity is concerned with generalization – where and when are the conclusions applicable and can we generalize from our experiments to industrial practice? External validity is threatened in two ways: the experiment participants received little training in the new method and the quality of the result has no influence, neither on a real product nor on the participant's working situation. However, these threats will influence both methods in the same way. Since we are only looking for differences and not for any absolute measure this will not influence our conclusions on the relative merits of the two methods.

6.5 Our Claims to Validity

Based on the discussions above, and besides a possible caveat for cultural differences, we claim that there are no serious threats to validity for our conclusions on the number of identified failure modes.

7 Conclusion

Based on our data analysis we will conclude that TUC is better than SSD when it comes to identifying failure modes related to required functionality and operator behaviour. The systems sequence diagrams outperform textual use cases when it comes to failure modes pertaining to the system's internal working. Based on this, we recommend that, for hazard analysis in the early phases, we should provide a set of documents that enables the analysts to focus on each area under consideration.

A natural direction for further work would be to do more experiments, for instance using industry practitioners in addition to students. It would also be interesting to compare various techniques in bigger case studies, since controlled experiments necessarily limit the size of the tasks that can be performed, thus lacking the realism and complexity of information systems development projects.

References

1. Gershenfeld, N., Krikorian, R., Cohen, D.: The Internet of Things. *Scientific American* 291(44), 76–81 (2004)
2. Batra, D., Hoffer, J.A., Bostrom, R.P.: Comparing Representations with Relational and EER Models. *Communications of the ACM* 33, 126–139 (1990)
3. Cheng, P.C.-H.: Why Diagrams Are (Sometimes) Six Times Easier than Words: Benefits beyond Locational Indexing. In: Blackwell, A.F., Marriott, K., Shimojima, A. (eds.) *Diagrams 2004*. LNCS (LNAI), vol. 2980, pp. 242–260. Springer, Heidelberg (2004)
4. Larkin, J.H., Simon, H.A.: Why a Diagram is (Sometimes) Worth Ten Thousand Words. *Cognitive Science* 11 (1987)
5. Boekelder, A., Steehouder, M.: Selecting and Switching: Some Advantages of Diagrams over Tables and Lists for Presenting Instructions. *IEEE Transactions on Professional Communication* 41, 229–241 (1998)
6. Allmendinger, L.: Diagrams and Design Tools in Context. *ACM SIGDOC Asterisk Journal of Computer Documentation* 18, 25–41 (1994)
7. Coll, R.A., Coll, J.H., Thakur, G.: Graphs and tables: a four factor experiment. *Communications of the ACM* 37, 77–84 (1994)
8. Guiochet, J., Vilchis, A.: Safety Analysis of a Medical Robot for Tele-echography
9. Long, Z., Jinglun, Z.: Analysis and Study of System Safety Based on Event Sequence Diagram. *International Journal of Computer Science and Network Security* 8(2) (February 2008)
10. Ren, C.: A Safety Inspection Management System for Mine equipment Based on UML. In: 2009 International Conference on Signal Processing Systems (2009)
11. Allenby, K., Kelly, T.: Deriving Safety Requirements Using Scenarios. In: *Proc. RE 2001*, Toronto, Canada, August 27-31. IEEE, Los Alamitos (2001)

12. Alspaugh, T.A., et al.: Clarity for Stakeholders: Empirical Evaluation of ScenarioML, Use cases and Sequence Diagrams. In: Fifth International Workshop on Comparative Evaluation in Requirements Engineering (2007)
13. Sindre, G., Opdahl, A.L.: Eliciting Security Requirements with Misuse Cases. *Requirements Engineering* 10, 34–44 (2005)
14. Alexander, I.F.: Misuse Cases, Use Cases with Hostile Intent. *IEEE Software* 20, 58–66 (2003)
15. Larman, C.: *Applying UML and Patterns – An Introduction to Object-Oriented Analysis and Design and Iterative Development*, 3rd edn. Pearson Education Inc., London, ISBN 0-13-148906-2
16. Pender, T.: *UML Bible*, Wiley Publishing Inc., Indianapolis, Indiana, US (2003), ISBN 0-7645-2604-9
17. Stålhane, T., Sindre, G.: A comparison of two approaches to safety analysis based on use cases. In: Parent, C., Schewe, K.-D., Storey, V.C., Thalheim, B. (eds.) *ER 2007*. LNCS, vol. 4801, pp. 423–437. Springer, Heidelberg (2007)
18. Stålhane, T., Sindre, G.: Safety Hazard Identification by Misuse Cases: Experimental Comparison of Text and Diagrams. In: Czarnecki, K., Ober, I., Bruel, J.-M., Uhl, A., Völter, M. (eds.) *MODELS 2008*. LNCS, vol. 5301, pp. 721–735. Springer, Heidelberg (2008)
19. Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., Wesslén, A.: *Experimentation in Software Engineering: An Introduction*. Kluwer Academic, Norwell (2000)
20. Tukey, J.W.: Data analysis and behavioral science or learning to bear the quantitative man burden by shunning badmandments. In: Jones, L.W. (ed.) *The Collected Works of John W. Tukey*, Wadsworth, Monterey, CA, vol. III, pp. 187–389 (1986)
21. Achour, C.B., et al.: Guiding Use Case Authoring: Results of an Empirical Study. In: *Proceedings of the 4th IEEE International Symposium on Requirements Engineering – RE 1999*, Limerick, Ireland (1999)
22. Cox, K., Phalp, K.: Replicating the CREWS use case Authoring Guidelines Experiment. *Empirical Software Engineering* 5, 245–267 (2000)

Feature-Based Entity Matching: The FBEM Model, Implementation, Evaluation*

Heiko Stoermer, Nataliya Rassadko, and Nachiket Vaidya

University of Trento,
Dept. of Information and Communication Tech.,
Trento, Italy
{stoermer, rassadko, vaidya}@disi.unitn.it

Abstract. Entity matching or resolution is at the heart of many integration tasks in modern information systems. As with any core functionality, good quality of results is vital to ensure that upper-level tasks perform as desired. In this paper we introduce the FBEM algorithm and illustrate its usefulness for general-purpose use cases. We analyze its result quality with a range of experiments on heterogeneous data sources, and show that the approach provides good results for entities of different types, such as persons, organizations or publications, while posing minimal requirements to input data formats and requiring no training.

Keywords: Entity resolution, record linkage, information integration.

1 Introduction

The question whether two pieces of structured or semi-structured data refer to the same object (or entity) has been the objective of substantial work over several decades, and good solutions are the prerequisites for high-quality automatic integration or linkage of data, information and knowledge. This integration topic and the related quality issues are receiving increased attention by work performed in the context of the Semantic Web, or more generally, the linkage of entity-related information in more general-purpose web information systems.

The problem, which appears in literature under names such as record linkage, entity linkage, entity resolution, etc., poses substantial challenges to automated approaches, particularly when faced with a heterogeneous environment. In open, de-centralized environments that are emerging on the Web, we can find an increasing amount of (semi-) structured information. But especially this lack of centralization – which is one of the reasons for the dynamic development of the web and thus one of its strengths – leads to difficult situations where for example creators of structured information published on the web refer to the entities they describe with arbitrary, self-issued identifiers, making it impossible for an automated system to perform efficient, syntactical integration of information from different sources [4].

* The authors would like to thank Barbara Bazzanella for her excellent work on [2], and for the permission to reproduce Table 1. This work is partially supported by the FP7 EU Large-scale Integrating Project **OKKAM – Enabling a Web of Entities** (contract no. ICT-215032). For more details, please visit <http://www.okkam.org>

Nonetheless, the demand for high-quality integration comes naturally from the side of users and producers of progressive information systems alike. The cost-efficient creation of a news and article dossier about a certain person (which can then be sold against a micropayment), the linkage of contacts between social networks, the collection of opinions about a certain product from different sources, and many more examples all require good-quality, easy-to-use automated approaches for entity resolution.

In this paper we present a novel approach for entity resolution, named FBEM (feature-based entity matching). The approach is a generalization of earlier work [17], and combines probabilistic as well as ontological methods for deciding whether two records describe the same entity, taking into account intensional and extensional aspects of the entities at hand. The approach aims at general-purpose usefulness with special focus on web information systems, and bases on empirical findings about what are commonly used entity types, and how they are usually described.

The rest of this article is organized as follows: the following section makes an attempt at giving a concise account of the vast amount of related work from several fields. Section 3 explains the background knowledge underlying the FBEM approach and its representation in an ontology which serves as input to the algorithm. Sect. 4 then introduces the algorithm, proposing a formal model for entity similarity and describing its implementation. In Sect. 5 we perform a thorough investigation of the usefulness of the algorithm, presenting a series of experiments in entity resolution which cover entity types such as organizations, persons and scientific publications in datasets of different sizes. The article closes with a discussion of the results and a description of further work that is planned for improving the approach.

2 Related Work

In contrast to schema-level integration, entity-level integration deals with the actual individuals, not with integration of class structures or entity types. Entity level integration has to deal with deciding whether two entity descriptions refer to the same individual. Approaches to the problem of entity resolution can be broadly categorized into two main categories: one requires training data to adapt the matching procedure with machine learning techniques, the other depends on domain knowledge for matching. The approach presented in this paper falls in the latter category.

Approaches of entity-level integration have been proposed under several names, ranging from duplicate detection [9], entity resolution [3, 11], merge/purge [12], object identification [18], reference reconciliation [8]. Extensive surveys are available [9, 5, 19]. A related group of algorithms are the ones that aim at matching entity names by computing the distance between the string values of corresponding entity names. The algorithms included in this group suggest general-purpose methods for computing the similarity between strings [14]. These algorithms are considered important since they are currently used as the basic metric on which more sophisticated approaches are based on. [7] describes and provides an experimental comparison of various string distance metrics. Other approaches involve schema matching [16] in cases where the entities to be matched are described with a different schema, also employing domain knowledge from ontologies, where available [15].

3 Knowledge about Entities

3.1 Background Knowledge

In her work on the foundations of entity representation [2], Bazzanella establishes six top-level entity types that are both specialized and generic enough to serve as a basic model for entity representation and matching in general-purpose environments such as the (Semantic) Web. These are: PERSON, ORGANIZATION, EVENT, ARTIFACT, LOCATION, OTHER.

In a feature-listing experiment with more than 350 participants, [2] compiles background knowledge about how humans describe entities, which – for the sake of completeness – we cite in Table 1.

Table 1. Relevance of features for describing basic entity types. (cf. [2]).

| Entity Type (<i>e</i>) | Attribute type (<i>a</i>) | $p(e a)$ | Entity Type (<i>e</i>) | Attribute type (<i>a</i>) | $p(e a)$ |
|--------------------------|-----------------------------|----------|--------------------------|-----------------------------|----------|
| <i>Person</i> | surname | 0.97 | <i>Artifact</i> | artifact type | 0.97 |
| | first name | 0.96 | | artifact name | 0.94 |
| | full name | 0.96 | | brand | 0.93 |
| | affiliation | 0.85 | | model | 0.91 |
| | occupation | 0.83 | | features | 0.83 |
| <i>Organization</i> | organization name | 0.98 | <i>Location</i> | location name | 0.98 |
| | activity | 0.85 | | location type | 0.89 |
| | organization type | 0.85 | | use | 0.63 |
| | part of | 0.49 | | place:province | 0.59 |
| | place:country | 0.15 | | attraction | 0.55 |
| <i>Event</i> | event type | 0.97 | | | |
| | event name | 0.96 | | | |
| | date:year | 0.92 | | | |
| | date:month | 0.84 | | | |
| | protagonist:surname | 0.79 | | | |

From these data we develop the following working hypothesis: we have two types of features (i) generic ones (“name” and “type”), that say nothing about the entity type but have an average importance in the model that is still higher than the one of an “unknown” feature, and (ii) type-discriminative ones that help us infer an entity type from its description; these features and their relevance values can be used beneficially to perform general-purpose entity resolution, by attempting to map two given records describing the same entity to the above model, and using the relevance values to influence the calculation of record similarity between the two.

3.2 An Ontology of Entity Description

In order to make use of the results described in the previous section in an actual algorithm, the FBEM OWL-ontology has been created which captures the aspects of features and feature weights in entity descriptions. The ontology defines concepts for Entity and Feature, as well as the necessary relations between them, as depicted in Fig. 1.

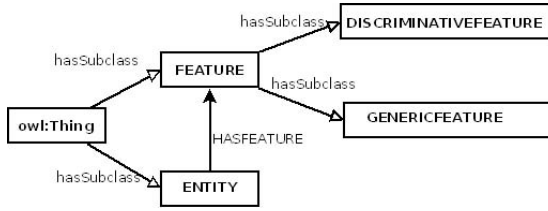


Fig. 1. Class structure of the background KB

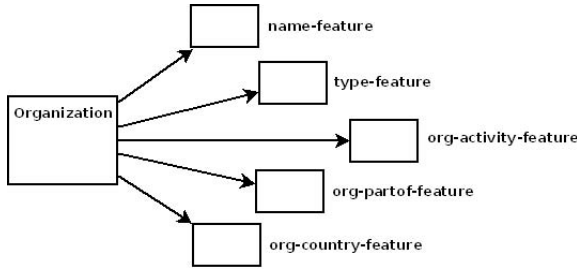


Fig. 2. Example individual of class Entity describing the Artifact type

The actual content of the ontology is in the individuals of the class Entity and its subclasses. One example is given in Fig. 2 which represents the entity type ORGANIZATION and its features such as *activity*, *type* or *partof*. Each of those feature individuals has a set of datatype properties containing the feature weights of type float, and – in order to support synonymity and multilinguality – also a set of so-called *label patterns* which contain the primary name of a feature as established in Sect. 3.1 as well as possible synonyms and natural language versions.

These label patterns are used by the FBEM implementation to attempt establishing a mapping of feature names of the given entities into the FBEM ontology, which is in fact a coarse, but very efficient way of schema matching, which we have implemented for different natural languages (currently German, Italian and English).

4 A Feature-Based Entity Similarity Model

4.1 The FBEM Similarity Score

To present a ranked list of candidate entities that match a reference entity (or a query), we require a score that serves as parameter for ranking, which expresses the closeness of a candidate entity to the reference entity, relative to all other candidates.

In our setting, the representation of a reference entity Q and candidate entity E is modeled as the set of features F plus an optional type t . The type information is not required to follow any form or (natural) language, but is free text:

$$Q \equiv E \equiv \langle F, t \rangle;$$

The first part of the representation of Q and E is in the form of a set F of *features*, which are represented as $\langle \text{name}, \text{value} \rangle$ pairs that are independent in content and size (i.e. they don't necessarily share a vocabulary or schema, or even a natural language):

$$f = \langle n, v \rangle;$$

$$F = \{f_1, f_2, \dots\};$$

We define the following functions:

$n(f)$: returns the *name* part of a feature f of an E or Q ;

$v(f)$: returns the *value* part of a feature f of an E or Q ;

$typeof(E)$: returns the ontological type of an E or Q .

In order to establish similarity between two entities, we require additional operators and functions that are later used for the computation.

$$mapsto(f) =_{def} \exists x, f(\text{FEATURE}(x) \wedge f \in F \wedge \text{HASLABELPATTERN}(x, n(f))) \quad (1)$$

Axiom 1 provides the definition of a function that maps a feature of Q or E into the background KB. The function must return the feature into which f has been mapped, or NIL.

$$typeDesc(f) =_{def} \exists x(\text{DISCRIMINATIVEFEATURE}(x) \wedge mapsto(f) \equiv x) \quad (2)$$

Axiom 2 provides the definition of a boolean function that determines whether a feature of Q or E is type-discriminative. It does so by determining whether a feature can be mapped into the set of type-discriminative features of the background KB.

We furthermore define three boolean operators that describe whether Q and E are type-compatible (c_E), incompatible (ci_E), or of unknown compatibility (cu_E):

$$c_E(Q, E) =_{def} typeof(Q) \equiv typeof(E) \quad (3)$$

Axiom 3 states that Q and E are considered compatible if they have the same ontological type.

$$ci_E(Q, E) =_{def} typeof(Q) \neq typeof(E) \quad (4)$$

Axiom 4 defines that Q and E are considered incompatible if they have different ontological types in the background KB.

$$cu_E(Q, E) =_{def} \neg ci_E(Q, E) \wedge \neg c_E(Q, E) \quad (5)$$

Axiom 5 defines that Q and E are considered of unknown compatibility if they are neither compatible nor incompatible.

In order to involve background knowledge into the calculation of entity similarity, we also need to define a set of operators that indicate whether a pair of features f_Q and f_E are name-identical (cid_f), name-compatible (c_f), name-incompatible (ci_f) or of unknown compatibility (cu_f):

$$\begin{aligned}
cid_f(f_Q, f_E) &=_{def} \\
& (c_E(Q, E) \vee cu_E(Q, E)) \wedge ((mapsto(n(Q)) \equiv mapsto(n(E))))
\end{aligned} \tag{6}$$

Axiom [6](#) defines that two features are known to be name-identical with respect to the feature name, if their respective records Q and E are not incompatible, and if the features map into the same ontological feature of the background KB.

$$\begin{aligned}
c_f(f_Q, f_E) &=_{def} \\
& (c_E(Q, E) \vee cu_E(Q, E)) \wedge (n(Q) \equiv n(E))
\end{aligned} \tag{7}$$

Axiom [7](#) defines that two features are name-compatible if their respective records Q and E are not incompatible, and the respective feature names are identical strings (but unknown with respect to our background KB).

$$\begin{aligned}
ci_f(f_Q, f_E) &=_{def} \\
& (ci_E(Q, E)) \vee (mapsto(n(Q)) \neq mapsto(n(E)))
\end{aligned} \tag{8}$$

Axiom [8](#) defines that two features are name-incompatible if their respective records Q and E are incompatible, or if they map into different ontological features of the background KB.

$$\begin{aligned}
cu_f(f_Q, f_E) &=_{def} \\
& \neg cid_f(f_Q, f_E) \wedge \neg c_f(f_Q, f_E) \wedge \neg ci_f(f_Q, f_E)
\end{aligned} \tag{9}$$

Axiom [9](#) defines that two features are of unknown compatibility if they are neither name-identical, name-compatible or name-incompatible.

Now, we define $fsim(f_Q, f_E)$, a function that computes the similarity of two features f_Q, f_E , taking into account the similarity of the value parts of f_x , as well as our background knowledge base:

$$\begin{aligned}
fsim(f_Q, f_E) &=_{def} \\
sim(v(f_Q), v(f_E)) * & \begin{cases} w_c + p(E|f_E), & \text{for } cid_f(f_Q, f_E); \\ w_c, & \text{for } c_f(f_Q, f_E); \\ w_u, & \text{for } cu_f(f_Q, f_E); \\ 0, & \text{otherwise .} \end{cases}
\end{aligned} \tag{10}$$

Equation [10](#) relies on the following functions and parameters:

$sim(x, y)$: a suitable string similarity measure between x and y .

$p(E|f_E)$: the relevance of the feature to describe a given entity type, as defined in the background KB (see Sect. [3.1](#)); please note that $p(Q|f_Q) = p(E|f_E)$ because the model requires that the features map into the same element of the background KB;

w_c : the importance which is given to the fact that a pair of features is compatible;

w_u : the importance which is given to the fact that a pair of features is of unknown compatibility.;

At this point we are able to establish the similarity between individual features. To compute the complete feature-based entity similarity, which finally expresses to which extend E is similar to Q , we proceed as follows.

Let $maxv(V)$ be a function that computes the maximum value in a vector¹. We then span the matrix M of feature similarities between Q and E , defined as

$$M := (fsim(Q, E))_{|Q| \times |E|} \rightarrow \mathbb{Q} \geq 0$$

with $fsim$ as defined above, $|Q|, |E|$ being the number of elements of the vectors Q and E , respectively, and \mathbb{Q} is set of rational numbers.

The feature-based entity similarity score fs is defined as the sum of all the *maximum similar* feature combinations between Q and E :

$$fs(Q, E) = \sum_{i=1}^{|Q|} maxv(M_i) \quad (11)$$

Here M_i is i^{th} row of the matrix M . Please note that when type of entity E and that of query Q are given or can be inferred and found to be not matching, E and Q will be incompatible. Our approach is modeled in such a way that when Q and E is incompatible, the similarity score for them will be zero.

Taking into account that M_i is a weighted value, we use a dot-notation denote its weight w as $M_i.w$. Using a mathematical normalization of the similarity score, the final entity similarity measure $esim(Q, E)$ can be defined as follows:

$$esim(Q, E) = \frac{fs(Q, E)}{\sum_{i=1}^{|Q|} maxv(M_i).w} \quad (12)$$

In the last formula, we divided a sum of weighted values on a sum of corresponding weights. This allows us to normalize similarity score within the range of $[sim(x, y)_{min}, sim(x, y)_{max}]$, e.g., $[0, 1]$.

4.2 Implementation

For implementing an algorithm that can provide the entity similarity defined in Eq. 12, there are several variables to set and operators to implement which can not be trivially derived from the definitions in the previous section. In the following we give details about their implementation.

typeof(E): An implementation of the the *typeof* operator is required to return the ontological type of the entity E or the intended ontological type of the desired entity that is described by Q . Compatibility of types will later influence the similarity that is established between a Q and an E (especially type-incompatibility will lead to a significantly lower score). The ontological type of Q or E can be inferred from a successful mapping of a given feature into an individual of class Discriminative-Feature in the FBEM ontology, or by an explicit specification of the type. Please

¹ Trivially defined as $maxv(V) = max_{i=1}^{|V|} (V_i)$, with $|V|$ being the number of elements of V .

note that due to the flexibility of this model, the involvement of a sophisticated schema-matching component from such a freely specified type to one of the types in our knowledge model may be of benefit. For the results presented in this article, the implementations of various functions are given below.

$mapsto(f)$: The *mapsto* function establishes a mapping between a feature f and an instance of the class `FEATURE` of our background ontology. It does so by checking whether it is possible to establish a string match between $n(f)$ and one of the values of the `HASLABELPATTERN` property of the instance.

$sim(x, y)$: This operator returns a similarity measure between the strings x and y , in the range of $[0, 1]$.

$p(E|f_E)$: The relevance of a feature for a certain entity is retrieved from the background KB, using the *mapsto* operator. If the *mapsto* function fails, this value will be unknown.

w_c : This weight can be empirically optimized, default is 0.5.

w_u : This weight can be empirically optimized, default is 0.25.

The algorithm implements certain optimizations, e.g. the reasoning that is required to derive the necessary facts from the background ontology is being performed at loading time of the algorithm, materialized and then cached in very efficient memory structures because direct interaction with the OWL model has proven to be unsustainable in terms of runtime performance.

5 Evaluation

5.1 Evaluating String Similarity Measures

As explained in the previous sections, the FBEM algorithm performs string similarity matching between a selection of values of the entities that are to be compared. To understand what is the impact on the selection of a string similarity measure on the overall performance of the algorithm, an experiment has been performed that runs FBEM on the Eprints and Rexa datasets (see Sect. 5.5) using four different metrics: the well-known Levenshtein [13] and Soundex algorithms, as well as Monge-Elkan [14] and TagLink [6]. The results reported in Table 2 illustrate quite drastically that a poor choice of string similarity measure has a negative impact on the quality of matching results. While Levenshtein delivers good results, it does so in very few cases. Other approaches such as Soundex or Monge-Elkan fail our requirements completely. Only the TagLink measure provides acceptable results, and a possible reason may be that the algorithm establishes similarities between tokens of strings which makes it less vulnerable to difference in token sequence (as is the case for names, such as “Barack Obama” vs. “Obama, Barack. T.”).

5.2 Record Identity Tests

To perform a baseline evaluation of the FBEM algorithm, a test of record identity has been performed to measure the quality of the algorithm when no difference between records exist. The dataset that was used is a collection of 50,000 *entity profiles* taken

Table 2. FBEM performance using different string similarity measures

| | Levenshtein | Soundex | Monge-Elkan | TagLink |
|-----------|-------------|---------|-------------|---------|
| Precision | 0.95 | 0.00 | 0.01 | 0.72 |
| Recall | 0.05 | 0.16 | 0.48 | 0.77 |
| F-Measure | 0.10 | 0.01 | 0.02 | 0.75 |

Table 3. Experimental results analyzing the matching of identical records (full duplicates)

| Experiment 1 | | Experiment 2 | |
|--------------------|-----|--------------------|--------|
| Dataset A size | 100 | Dataset A size | 100 |
| Dataset B size | 100 | Dataset B size | 50,000 |
| Overlap $A \cap B$ | 100 | Overlap $A \cap B$ | 100 |
| Precision | 1 | Precision | 1 |
| Recall | 1 | Recall | 1 |

from the Entity Name System², which contains a majority of geographic entities, as well as some organizations and persons. The records are represented in a flat list of free-form name/value pairs that follow no particular schema³.

Starting from this basic dataset, we performed two experiments. One is a random selection of 100 samples that were matched against each other (cartesian product). A second experiment evaluates result quality of 100 random samples when matched against the complete dataset of 50,000 records. The results are reported in Table³.

5.3 Aligning Restaurant Records

The restaurant dataset⁴ is composed of about 864 restaurant records from two different data sources (Fodor’s and Zagat’s restaurant guides), which are described by name, street, city, phone and restaurant category. Among these, 112 record pairs refer to the same entity, but usually display certain differences. An example is given in Table⁴.

The objective of this experiment was to use real-world data that do not cover the “usual suspects” such as scientific articles or authors. In detail, the data were organized in a way that the 112 records from Fodor’s which had a counterpart in Zagat’s were added to a dataset A, and all the others were merged into a dataset B. Then we performed an evaluation using the FBEM algorithm without any modifications, to measure how many entries in dataset A could be successfully found in dataset B. The quality of results is very promising, both recall and precision are above 0.95; more details are reported in Table⁵.

5.4 Aligning Person Records

The *people2* dataset⁴ contains two files, *A* with original records of people and *B* another with modified records from the first file. *B* contains maximum 9 modified entries for

² <http://www.okkam.org>

³ Originally provided by Sheila Tejada, downloaded from

<http://www.cs.utexas.edu/users/ml/riddle/data.html>

⁴ Febr project: <http://sourceforge.net/projects/febr1/>

Table 4. Example records of the restaurants data set

| Fodor's | Zagat's |
|---|----------------------|
| name carmine's | name carmine's |
| street 2450 broadway between 90th and 91st sts. | street 2450 broadway |
| city new york | city new york city |
| phone 212/362-2200 | phone 212-362-2200 |

Table 5. Evaluation results of FBEM on the restaurant dataset

| Dataset details | Results |
|------------------------|-------------------|
| Dataset A size 112 | False positives 2 |
| Dataset B size 744 | False negatives 4 |
| Overlap $A \cap B$ 112 | Recall 95% |
| | Precision 98% |

an original records in A , with maximum 3 modifications per attribute, and maximum 10 modifications per record. The original file contains 600 records, while B contains 400 records which are modifications of 95 original records from A . The attributes of the records are record id, given name, surname, street number, address, suburb, postcode, state, date of birth, age, phone number, social security number. An example is given in Table 6. It is evident from the example that there substantial modifications per record. Still, the results show precision and recall both above 0.77; more details are reported in Table 7.

Table 6. Example records of the people2 data set

| A | | B | |
|---------------|--------------|---------------|--------------|
| rec_id | 2280 | rec_id | 2285 |
| given_name | kate | given_name | kath |
| surname | peat | surname | peat |
| street_number | 111 | street_number | 1 |
| ... | | | |
| address_1 | duffy street | address_1 | street duffy |
| suburb | robina | suburb | robivna |
| date_of_birth | 19450303 | date_of_birth | 19450033 |
| phone_number | 02 90449592 | phone_number | 04 03014449 |
| ... | | | |

As evident in table 8 there are 505 entries which are not present in the alignment. This dataset was used to test for true negatives, i.e. the ability of the algorithm to decide that a searched entity is *not* present in the target data. This experiment is important for scenarios where concrete decisions have to be taken, instead of delivering only a list potential matches without giving further information about their quality. The results for the experiment are promising, with accuracy at 95% for the given dataset. Table 8 gives the details of the experiment.

Table 7. Evaluation results of FBEM on the people2 dataset

| Dataset details | Results |
|------------------------|--------------------|
| Dataset A size 600 | False positives 71 |
| Dataset B size 400 | False negatives 93 |
| Overlap $A \cap B$ 400 | Recall 77% |
| | Precision 81% |
| | F-Measure 79% |
| | Fallout 19% |

Table 8. Results characterizing the ability to decide about true negatives

| | |
|--|-----|
| Total number of entities in dataset A size | 600 |
| Number of entities which have no correspondence in B | 505 |
| Total number of entities which have no duplicate in our result | 488 |
| Number of missing entities | 26 |
| Number of falsely detected entities | 9 |
| Accuracy | 95% |

5.5 Aligning Bibliographic Databases

For the experiments described in this section we used benchmarks provided by the instance matching contest of the Ontology Alignment Evaluation Initiative 2009 (OAEI)⁵. Each benchmark is accompanied with a gold standard in alignment format [10]. The gold-standard alignment is the set of pairs of entities that are known to match, which is then used to evaluate precision and recall of the alignment produced by the experiment.

The benchmark consists of three datasets containing instances from the domain of scientific publications, i.e. information about authors and their publications in proceedings, journals and books.

Eprints contains data about papers produced for the AKT research project. The dataset contains around 1000 entities.

Rexa is generated from the search results of the search server Rexa. The dataset contains over 20000 entities.

SWETO-DBLP is a version of DBLP modeled against the SWETO ontology. The dataset contains over 1000000 entities.

For this experiment, the data are homogenized into the flat name/value pair format that the FBEM algorithm accepts as input. Note that the homogenization is not an integral part of the FBEM algorithm, and thus can be customized. However, a simple, generic homogenizer for RDF data was used for the described experiments in order to produce comparable results. This homogenizer performs no particular reasoning or encodes any kind of knowledge about the underlying data, it only converts the long WWW-style URIs into short names, and removes relations that point to other classes or individuals, thus maintaining only datatype properties directly associated to the entity at hand.

⁵ All data are available from <http://oei.ontologymatching.org/2009/>

Table 9. Matching results for the Eprints-Rexa-DBLP benchmark

| | Precision | Recall | F-Measure | Fallout |
|----------------|-----------|--------|-----------|---------|
| Eprints – Rexa | 0.72 | 0.78 | 0.75 | 0.28 |
| Eprints – DBLP | 0.91 | 0.78 | 0.84 | 0.09 |
| Rexa – DBLP | 0.83 | 0.94 | 0.88 | 0.17 |

Table 10. Type-specific matching results between the Eprints and Rexa datasets

| | Precision | Recall | F-Measure | Fallout |
|---------------|-----------|--------|-----------|---------|
| Person | 0.70 | 0.81 | 0.73 | 0.30 |
| Article | 0.77 | 0.71 | 0.74 | 0.23 |
| Inproceedings | 0.96 | 0.69 | 0.80 | 0.04 |

In the experiment, the datasets Eprints, Rexa and DBLP were aligned pairwise. In order to achieve a result within a reasonable timeframe, the large amount of data contained in DBLP was reduced to the entities that are actually contained in DBLP-related gold standard, and then aligned w.r.t. eprints and rexa datasets. Finally, the resulting alignments were evaluated against the corresponding golden standard provided with the datasets.

The matching results are reported in Table 9 and give a positive overall picture. Important to note are the values for Fallout, which is defined as $(1 - \text{Precision})$ and reflects the amount of wrong mappings.

In order to gain a better understanding of the detailed strengths and weaknesses of FBEM w.r.t. entity types that are being matched, an additional experiment was performed that does typed matching between the Eprints and Rexa data. The three types that were selected from the datasets are “person”, “article” and “inproceedings”. The results in Table 10 does however not exhibit a very clear trend that could lead to a solid identification of entity types for which FBEM is particularly (un)usable. In terms of precision it behaves better on the larger records that describe publications, and less well on Person data, whereas recall behaviour is the opposite.

6 Discussion and Future Work

One important result presented in this paper is the strong influence that a string similarity metric has on the quality of the matching results. In Sect. 5 we have experimented with a selection of metrics which showed quite drastic differences in outcome. For this reason, one of the next steps will be to work on a component for FBEM that implements heuristics which will allow to select a suitable similarity measure for a given feature value. Such heuristics can base on aspects such as single-token vs. multi-token strings, well-known patterns, and the detection of specific names, e.g. for persons or companies, for which highly specialized algorithms exist.

A second goal is to further broaden the scope of the evaluation of the approach. While this aspect has been substantially expanded since the publication of the predecessor approach, the analyses we have performed have not yet shown results diverse enough

to understand which kind of data and/or entity types FBEM is particularly suitable for. For this reason, more heterogeneous benchmarks will be performed.

A further, mid-term goal is to address more in detail the aspects of large-scale, high-performance entity resolution. Several performance aspects have already been kept in mind during the development of the FBEM implementation. Nonetheless, several additional aspects have to be addressed. First, a good selection of stopping techniques needs to be compiled which will allow the algorithm to cease comparing features when a highly relevant match has been found (e.g. an identical social security number). Finally, blocking techniques will be analyzed to limit the amount of entity-to-entity comparisons. For example, the Entity Name System [4] implements a high-performance, index-based preselection of candidates, which are then further compared with more sophisticated, but also more costly methods. This approach will be one of the first to be evaluated.

7 Conclusion

In this paper we have presented a probabilistic, general-purpose approach for entity resolution which bases on background knowledge about entities and their representation. We have illustrated the important role that entity resolution plays in the engineering of applications that require good quality data integration, and have shown in a series of experiments that for common entity types such as people, organizations or locations, the FBEM approach delivers satisfying results, both in recall and in precision. While already performing at a good level of quality, several areas of improvement have been identified and discussed. These will be addressed in future evolutions of the approach.

In preparing this work it has proven quite difficult to collect suitable datasets which include evaluation standards, even though related work has been performed for many years. Publication not only of evaluation results, but also of benchmarks, seems vital in this context. The authors thus plan to compile the data used in this and future experiments in a coherent way, so that benchmarking of related approaches is rendered considerably more easy.

References

- [1] Bazzanella, B., Chaudhry, J.A., Palpanas, T., Stoermer, H.: Towards a General Entity Representation Model. In: Proceedings of the 5th Workshop on Semantic Web Applications and Perspectives (SWAP 2008), Rome, Italy (December 2008)
- [2] Bazzanella, B., Stoermer, H., Bouquet, P.: Top Level Categories and Attributes for Entity Representation. Technical Report 1, University of Trento, Scienze della Cognizione e della Formazione (September 2008)
- [3] Benjelloun, O., Garcia-Molina, H., Menestrina, D., Su, Q., Whang, S.E., Widomr, J., Jonas, J.: Swoosh: A Generic Approach to Entity Resolution. Technical report, Stanford InfoLab (2006)
- [4] Bouquet, P., Stoermer, H., Niederee, C., Mana, A.: Entity Name System: The Backbone of an Open and Scalable Web of Data. In: Proceedings of the IEEE International Conference on Semantic Computing, ICSC 2008, August 2008, pp. 554–561. IEEE Computer Society, Los Alamitos (2008), CSS-ICSC 2008-4-28-25

- [5] Brizan, D.G., Tansel, A.U.: A Survey of Entity Resolution and Record Linkage Methodologies. *Communications of the IIMA* 6(3), 41–50 (2006)
- [6] Camacho, H., Salhi, A.: A string metric based on a one to one greedy matching algorithm. In: *Research in Computer Science* number, pp. 171–182 (2006)
- [7] Cohen, W.W., Ravikumar, P., Fienberg, S.E.: A Comparison of String Distance Metrics for Name-Matching Tasks. In: *Proceedings of the IJCAI 2003 Workshop IIWeb*, Acapulco, México, August 9-10, pp. 73–78 (2003)
- [8] Dong, X., Halevy, A., Madhavan, J.: Reference Reconciliation in Complex Information Spaces. In: *SIGMOD 2005: Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, pp. 85–96. ACM Press, New York (2005)
- [9] Elmagarmid, A.K., Ipeirotis, P.G., Verykios, V.S.: Duplicate Record Detection: A Survey. *IEEE Transactions on Knowledge and Data Engineering* 19(1), 1–16 (2007)
- [10] Euzenat, J.: An api for ontology alignment. In: McIlraith, S.A., Plexousakis, D., van Harmelen, F. (eds.) *ISWC 2004*. LNCS, vol. 3298, pp. 698–712. Springer, Heidelberg (2004)
- [11] Garcia-Molina, H.: Pair-wise entity resolution: overview and challenges. In: Yu, P.S., Tsostras, V.J., Fox, E.A., Liu, B. (eds.) *Proceedings CIKM 2006*, Arlington, Virginia, USA, November 6-11, p. 1. ACM, New York (2006)
- [12] Hernández, M.A., Stolfo, S.J.: The merge/purge problem for large databases. *SIGMOD Rec.* 24(2), 127–138 (1995)
- [13] Levenshtein, V.I.: Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady* 10, 707–710 (1966)
- [14] Monge, A.E., Elkan, C.: An Efficient Domain-Independent Algorithm for Detecting Approximately Duplicate Database Records. In: *DMKD* (1997)
- [15] Noy, N.F.: Semantic Integration: a Survey of Ontology-based Approaches. *SIGMOD Rec.* 33(4), 65–70 (2004)
- [16] Rahm, E., Bernstein, P.A.: A Survey of Approaches to Automatic Schema Matching. *VLDB Journal: Very Large Data Bases* 10(4), 334–350 (2001)
- [17] Stoermer, H., Bouquet, P.: A Novel Approach for Entity Linkage. In: Zhang, K., Alhajj, R. (eds.) *Proceedings of IRI 2009*, the 10th IEEE International Conference on Information Reuse and Integration, Las Vegas, USA, August 10-12. IRI, vol. 10, pp. 151–156. IEEE Systems, Man and Cybernetics Society (2009)
- [18] Tejada, S., Knoblock, C.A., Minton, S.: Learning object identification rules for information integration. *Inf. Syst.* 26(8), 607–633 (2001)
- [19] Winkler, W.E.: The State of Record Linkage and Current Research Problems. Technical report, Statistical Research Division, U.S. Census Bureau, Washington, DC (1999)

Dealing with Matching Variability of Semantic Web Data Using Contexts

Silvana Castano, Alfio Ferrara, and Stefano Montanelli

Università degli Studi di Milano,
DICO, via Comelico 39, 20135 Milano, Italy
{castano,ferrara,montanelli}@dico.unimi.it

Abstract. Goal of this paper is to propose a reference modeling framework to explicitly identify and formalize the different levels of variability that can arise along all the involved dimensions of a matching execution. The proposed framework is based on the notion of *knowledge chunk*, *context*, and *mapping* to abstract the variability levels and related operations along the *source-dataset*, the *matching-dataset*, and the *mapping-set* dimensions, respectively. An application of the proposed framework with instantiation in the HMatch 2.0 systems is illustrated.

Keywords: Variability modeling, matching, semantic web, knowledge management.

1 Introduction

Techniques for matching semantic web data are an essential component of modern information management and evolution architectures, to correctly compare and integrate disparate resource descriptions and to promote effective resource sharing and integration on the global scale [1]. Several tools and approaches have been proposed in the literature for performing ontology and schema matching [2,3], and, more recently, also for performing instance matching [4,5]. A key demand for effective matching tools and techniques is the capability to deal with different kinds of variability that can emerge during a matching execution on a certain dataset. For example, one must be able to deal with variability of data representations both from a linguistic and a structural point of view, with the purpose of dynamically isolating only the subset of data relevant for a certain matching goal. As another example, the output of the matching process can result in mappings at different levels of granularity, providing just a similarity measure expressing that pairs of concepts or instances match as a whole, or more detailed information about mapping rules to convert their matching properties one to another. For effective matching of semantic web data, a systematic analysis and modeling of the various kinds of variability that influence a matching execution are required, and a conceptual framework for their classification is still missing in the field.

Goal of this paper is to propose a reference modeling framework to explicitly classify and formalize the different levels of variability that can arise along all

the involved dimensions of a matching execution. The proposed framework is based on the notion of *knowledge chunk*, *context*, and *mapping* to abstract and formalize the variability levels and related operations along the *source-dataset*, the *matching-dataset*, and the *mapping-set* dimensions, respectively. An application of this framework to show its instantiation in the HMatch 2.0 system is also illustrated. We want to stress that we do not propose yet another matching approach, rather we want to focus on the problem of modeling the matching variability per se'. To the best of our knowledge, this is a novel contribution in the field of matching, with can be taken as a reference for i) providing a disciplined guidance for the design of new matching tools/techniques, flexibly customizable to operate in different situations at both schema and instance level and ii) defining a conceptual framework to analyze and compare existing systems/approaches with respect to the degree of variability actually supported.

The paper is organized as follows. After discussing related work in Section 2, in Section 3, we introduce the basic notions of the proposed Matching Variability Framework. Details about the formalization of contexts for modeling matching variability along each dimension are presented in Section 4, 5, and 6. In Section 7, we discuss an example of instantiation in the HMatch 2.0 system. Finally, concluding remarks are provided in Section 8.

2 Related Work

Relevant work on matching variability exists in the literature on ontology and schema matching [2,3,6]. However, we observe that in most of the existing approaches, variability is only partially supported or it is considered but in an implicit way by the various tools and prototypes. In particular, variabilities at the matching-dataset level are “embedded” in the tools through some level of configurability of the matching process. In this direction, a relevant example is provided by tools like ASMOV [7], RiMOM [8], and DSSim [9], as well as in mostly theoretical approaches like [10]. More recent work in the field of ontology matching is mainly focused on investigating the variabilities at the level of the mapping-set. In particular, recent approaches are being proposed to discuss how the mappings produced by different tools can be combined on the basis of different similarity measures. In [11], the variability of the possible mapping combinations is exploited to improve the mapping quality in terms of precision and recall, rather than to work on the modeling aspects of mapping combination. In this respect, interesting work are provided in [12,13], where the focus is more on presenting the possible operations that can be performed over a mapping-set, rather than on discussing the variabilities of the matching execution that originates them. Two other kinds of approaches to the problem of mapping exploitation are given in the framework of query answering [14] and of reasoning-based mapping validation [15,16]. In both the cases, however, the proposed techniques do not take into account the problem of modeling different mapping exploitation/validation strategies in correspondence of different matching contexts and/or goals. We note that all the presented tools/approaches are mainly focused on discussing

the aspects of variability at the level of both matching-dataset and mapping-set, while matching variabilities at the level of the source-dataset have not been formalized yet. With respect to the related work, this paper is a first attempt to give a comprehensive formalization of matching variability, by considering all the different dimensions involved in a matching execution.

3 Modeling Matching Variability

Data and ontology matching is frequently invoked in different scenarios and with different purposes. For this reason, the requirements and conditions under which matching is performed can vary from one case to another according to the specific goal that needs to be satisfied. This means that different types and different levels of variability characterize each matching execution. We represent these levels of variability along the following three dimensions: *variability of the source-dataset*, *variability of the matching-dataset*, and *variability of the mapping-set*.

Variability of the source-dataset. This kind of variability describes the different customizations that can be performed over the initial (i.e., source) dataset considered for matching. In particular, this variability dimension expresses the possible filtering operations that can be applied to the source-dataset to restrict and to better focus on the data of interest for the considered matching execution. We distinguish two different levels of variability along this dimension:

- *Abstract selection* (α). It allows to filter out the source-dataset by selecting for matching only a subset of the data properties according to a given criterion.
- *Concrete selection* (χ). It allows to filter out the source-dataset by selecting for matching only those data that exhibit a certain value for a given (set of) property.

As a result of the application of these levels of variability to the source-dataset, the so-called matching-dataset is produced.

Variability of the matching-dataset. This kind of variability describes the different customizations on the accuracy/deepness of the matching execution that can be performed over the matching-dataset. We distinguish three different levels of variability along this dimension:

- *Constraint matching* (κ). It allows to specify a precondition (i.e., a constraint) that has to be verified by the data in the matching-dataset for being matched. This way, it is possible to subordinate the matching execution to those data that satisfy the given constraint.
- *Scope matching* (π). It allows to specify at which level of deepness the matching execution has to be performed in terms of number and kind of features to consider for the data comparison.

- *Weight matching* (ω). It allows to assign a different level of relevance to the properties of the data to be matched. This way, it is possible to specify that one or more properties have a higher relevance than others within a given matching execution.

Variability of the mapping-set. This kind of variability captures the different customizations on the mappings that are returned as result of a matching execution. We distinguish three different levels of variability along this dimension:

- *Mapping ranking* (ρ). It allows to specify a minimum threshold that should be satisfied by a mapping to consider its corresponding elements as matching elements.
- *Mapping cardinality* (δ). It allows to specify the number of correspondences that are admitted in the matching result for a given element e . The choice spans from the one-to-one option, where only the mapping with the best matching element of e is included in the result, up to the many-to-many option, where all the discovered mappings with the matching elements of e are included in the result.
- *Mapping granularity* (γ). It allows to specify the level of granularity of the mappings produced as a result of the matching execution. The choice spans from a generic element-to-element correspondence, up to a complete mapping table of correspondences between the single properties of two matching elements.

The levels of variability along the three dimensions described above can be differently combined to obtain a specific configuration of the matching execution. All the possible matching variabilities are described by the three-dimension schema of Figure 1(a), where the nature of a matching execution is determined by the variability levels activated (+) or not activated (-) along the three dimensions. A given combination of choices depends on the specific target to satisfy. As an example in a bibliographic scenario, we consider the retrieval of publications of the same authors in the years from 2000 to 2003. This matching target can be satisfied by the combination $\langle +\alpha, +\chi, -\kappa, +\pi, -\omega, +\rho, +\delta, -\gamma \rangle$ (Figure 1(b)). The abstract selection ($+\alpha$) allows to restrict the matching execution to the comparison of authors, while the concrete selection ($+\chi$) allows to consider only publications from 2000 to 2003, respectively. The scope matching ($+\pi$) allows to set the maximum level of deepness for the comparison of authors of two publications by considering all the information available in the corresponding semantic web representations. Finally, the mapping ranking ($+\rho$) and the mapping cardinality ($+\delta$) allow to specify a matching threshold t and a many-to-many option for mapping cardinality, respectively. This way, for a given publication, all the publications with a matching value over t will be returned in the matching result. As another example, the matching configuration for discovering all the bibliographic records referring to the same real publication is shown in Figure 1(c) and it will be discussed in Section 7.

3.1 The Matching Variability Framework

To model matching variabilities modeling, we introduce the *Matching Variability Framework*, based on the notions of *knowledge chunk*, *mapping*, and *context*.

Knowledge chunk. A knowledge chunk kc represents an element of interest for matching, either concept or instance. Given a semantic web resource \mathcal{O} , like a RDF(S) repository or an OWL ontology, let \mathcal{N} be the set of element names in the signature of \mathcal{O} , \mathcal{P} is a set of property/relation names, \mathcal{L} the set of datatypes and literal values in \mathcal{O} , and id the identifier of \mathcal{O} , such as its URI. A knowledge chunk kc provides a synthetic representation of an element $e \in \mathcal{O}$ in terms of its constituent axioms/assertions, both explicitly and implicitly defined in \mathcal{O} . To this end, kc is defined as a set of axioms $kc = \{a_1(kc), a_2(kc), \dots, a_n(kc)\}$ constituting the specification of the corresponding element e . An axiom $a_i(kc)$, with $i \in [1, n]$ has the form $a_i(kc) = \langle n(kc), r(a_i), v(a_i), id \rangle$ where:

- $n(kc) \in \mathcal{N}$ is the name of the knowledge chunk kc , which coincides with the name of e .
- $r(a_i) \in \mathcal{P} \cup \{\equiv, \sqsubseteq\}$ is a semantic relation contained in the definition of e .
- $v(a_i) \in \mathcal{N} \cup \mathcal{L}$ is the value of the corresponding relation $r(a_i)$.
- id is the provenance of kc , namely the identifier of the resource from which kc is generated.

Given \mathcal{O} , a set of knowledge chunks is derived to provide a synthetic representation of concepts and instances contained in \mathcal{O} . In particular, a knowledge chunk kc_u is created for each URI $u \in \mathcal{O}$. An axiom $a_i(kc_u)$ is defined for each path $u \rightarrow v$ between u and a terminal node v in the RDF graph of the resource. The semantic relation $r(a_i)$ is defined as the concatenation of the labels of the properties p_1 and p_n in $u \rightarrow v$, while the value $v(a_i)$ is set to v . An example of knowledge chunk is given in Section 3.2, while a detailed description of the construction of knowledge chunks from RDF(S) and OWL resources is provided in [17].

Mapping. A mapping m denotes a semantic correspondence between two knowledge chunks kc_i and kc_j and it is a tuple of the form $m = \langle n(kc_i), n(kc_j), SA, \mathcal{U} \rangle$, where:

- $n(kc_i)$ and $n(kc_j)$ are the names of the matching knowledge chunks kc_i and kc_j , respectively.
- $SA \in [0, 1]$ is the semantic affinity value denoting the level of matching between kc_i and kc_j .
- \mathcal{U} is a (possibly empty) set of mapping rules, each one denoting the correspondence between pairs of matching axioms of kc_i and kc_j .

Context. A context defines a variability level in terms of operations on knowledge chunks and on mappings between them. The idea of using contexts to model matching variability derives from the field of conceptual modeling where

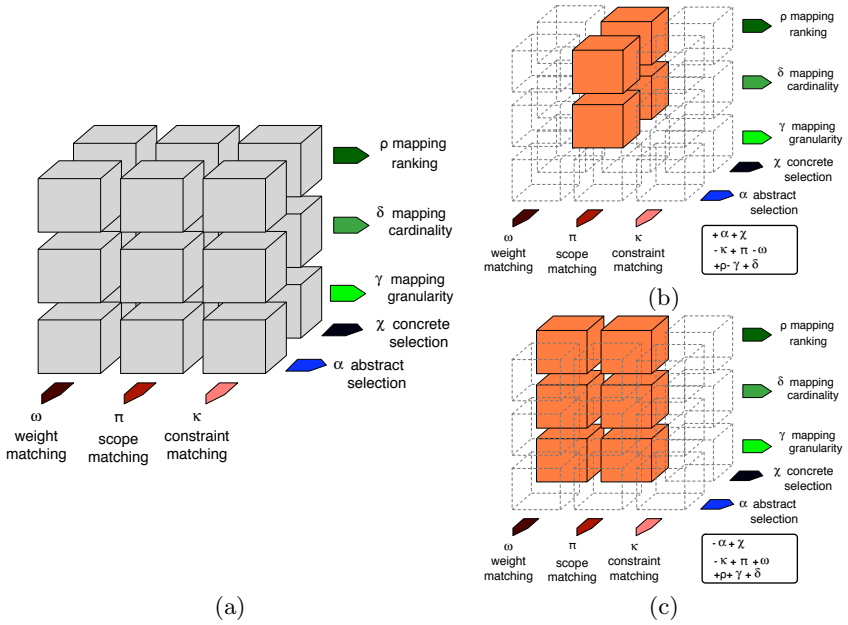


Fig. 1. Graphical representation of variability dimensions of matching (a), and examples of matching configurations (b) (c)

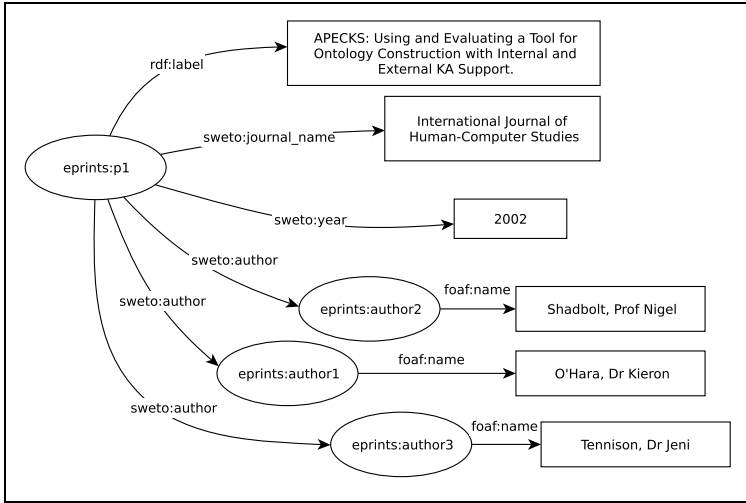
AKT EPrints

| | |
|------------|---|
| eprints:p1 | <p>sweto:author: O'Hara, Dr Kieron, Shadbolt, Prof Nigel, Tennison, Dr Jeni rdf:label: APECKS: Using and Evaluating a Tool for Ontology Construction with Internal and External KA Support. sweto:journal_name: International Journal of Human-Computer Studies sweto:year: 2002</p> |
| eprints:p2 | <p>sweto:author: Alani, Dr Harith, Dasmahapatra, Dr Srinandan, Gibbins, Dr Nicholas, Glaser, Hugh, Harris, Steve, Kalfoglou, Dr Yannis, O'Hara, Dr Kieron, Shadbolt, Prof Nigel rdf:label: Managing Reference: Ensuring Referential Integrity of Ontologies for the Semantic Web. sweto:book_title: 13th International Conference on Knowledge Engineering and Knowledge Management (EKAW'02) sweto:year: 2002</p> |
| eprints:p3 | <p>sweto:author: Carr, Dr Leslie, Kampa, Dr Simon, Miles-Board, Mr Timothy rdf:label: Hypertext in the Semantic Web. sweto:book_title: ACM Conference on Hypertext and Hypermedia 2001 sweto:year: 2001 sweto:pages: pp. 237-238</p> |

Rexa

| | |
|---------|--|
| rexa:p1 | <p>sweto:author: Kieron O'Hara, Nigel R. Shadbolt sweto:journal_name: International Journal of Human-Computer Studies sweto:year: 2002</p> |
| rexa:p2 | <p>sweto:author: Harith Alani, Srinandan Dasmahapatra, Nicholas Gibbins, Hugh Glaser rdf:label: Ensuring referential integrity of ontologies for the semantic web. sweto:book_title: Managing reference sweto:year: 2002 sweto:pages: 317-334</p> |

Fig. 2. A portion of the AKT EPrints and of the Rexa datasets



(a)

| kc_i | $r(a_i)$ | $v(a_i)$ | id |
|------------|--------------|--|---------|
| eprints:p1 | author_name | Shadbolt, Prof Nigel | eprints |
| | author_name | Tennison, Dr Jeni | eprints |
| | author_name | O'Hara, Dr Kieron | eprints |
| | label | APECKS: Using and Evaluating a Tool for Ontology Construction with Internal and External KA Support. | eprints |
| | journal_name | International Journal of Human-Computer Studies | eprints |
| | year | 2002 | eprints |

(b)

Fig. 3. Example of RDF description of a publication (a) and the corresponding knowledge chunk (b)

it has been introduced for the purpose of dealing with domain variability and requirements engineering [18,19].

In the Matching Variability Framework, we abstract a matching execution $Match$ as $Match(\mathcal{D}) \rightarrow \mathcal{M}$, where \mathcal{D} is a *source-dataset*, namely the set of knowledge chunks considered for the matching execution, and \mathcal{M} is a *mapping-set*, namely the set of mappings produced as a result of the matching execution.

3.2 Matching Example

To classify matching variability and to illustrate our proposed framework, we consider bibliographic data based on the ARS benchmark datasets provided by the instance matching track of OAEI 2009¹.

In particular, we focus on a test case that includes two datasets in the domain of scientific publications, namely *AKT EPrints* and *Reax*². A portion of the

¹ Ontology Alignment Evaluation Initiative: <http://oaei.ontologymatching.org/2009>

² Both the datasets are available at <http://www.intancematching.org/>

test case is shown in Figure 2 where we report data extracted from the RDF description of three publications from AKT EPrints and two publications from Rexa, respectively. An example of the RDF representation for the publication `eprints:p1` (see Figure 2) is shown in Figure 3(a) according to the SWETO-DBLP ontology³, an extension of FOAF⁴ for the domain of scientific publications and related authors. In particular, the properties `rdf:label` and `sweto:author` are used to represent the title and the author of publications, respectively. An example of the knowledge chunk corresponding to the bibliographic record `eprints:p1` is shown in Figure 3(b).

In the following sections, we introduce the contexts that formalize the variability levels along each dimension of Figure 1.

4 The Source-Dataset Contexts

Variability along the source-dataset dimension is formalized through the following two contexts.

Definition 1. Abstract context. *The abstract context of a knowledge chunk kc is a unary operation $\alpha_c(kc) \rightarrow 2^{kc}$ that returns the set of axioms $\overline{kc} \subseteq kc$ that satisfy the abstract context condition c associated with $\alpha_c(kc)$. The condition c is an arbitrary combination of conjunctions and/or disjunctions of property names $p \in \mathcal{P}$. Each component of the condition c is satisfied if $\exists r(a_i) \in kc \mid r(a_i) = p$.*

Abstract contexts filter data with respect to the data structure, that is the set of properties and relations associated with a given knowledge chunk. For example, in order to focus only on journal papers and their titles for matching publications, we apply the abstract context $\alpha_{journal_name \wedge label}$ to the knowledge chunks of eprints in Figure 2. As result, we have that only `eprints:p1` satisfies the abstract condition and will be included in the matching operation (see Figure 4).

| kc_i | $r(a_i)$ | $v(a_i)$ | id |
|-------------------------|--------------|--|----------------------|
| <code>eprints:p1</code> | label | APECKS: Using and Evaluating a Tool for Ontology Construction with Internal and External KA Support. | <code>eprints</code> |
| | journal_name | International Journal of Human-Computer Studies | <code>eprints</code> |

Fig. 4. Example of abstract context result

Definition 2. Concrete context. *The concrete context of a knowledge chunk kc is a unary operation $\chi_c(kc) \rightarrow \{kc, \emptyset\}$ that returns the knowledge chunk kc itself if the axioms of kc satisfy the concrete context condition c , and returns the empty set otherwise. The concrete context condition c is an arbitrary combination of conjunctions and/or disjunctions of boolean predicates of the form $\langle v(a_i) \theta k \rangle$, where k is a constant value, and θ is a comparison operator in $\{>, <, =, \geq, \leq, \neq, LIKE\}$, where `LIKE` denotes a pattern matching operator for strings (such as in SQL).*

³ http://lsdis.cs.uga.edu/projects/semdis/swetodblp/august2007/opus_august2007.rdf

⁴ <http://xmlns.com/foaf/spec/>

Concrete contexts filter source data with respect to their contents, that is the property values associated with a given knowledge chunk. Still referring to publications, we would focus the matching execution only on the publications produced after 2001. We apply the concrete context $\chi_{year > 2001}$ to eprints publications of Figure 2, by selecting only eprints:p1 and eprints:p2.

5 The Matching-Dataset Contexts

Variability along the matching-dataset dimension is formalized through three specific contexts as follows.

Definition 3. Constraint context. *The constraint context is a binary operation $\kappa_c(kc, kc') \rightarrow \{(kc, kc'), \emptyset\}$ that, given two knowledge chunks kc and kc' submitted to matching, returns the pair (kc, kc') if kc and kc' satisfy the constraints c , and returns the empty set otherwise. The constraint c is an arbitrary conjunction/disjunction of boolean predicates of the form $\langle p \theta p' \rangle$, where θ is a comparison operator in $\{>, <, =, \geq, \leq, \neq\}$ and p and p' denote two property names in \mathcal{P} , respectively. The predicate $p \theta p'$ is satisfied if $\exists a_i(kc), a_j(kc') \mid r(a_i) = p, r(a_j) = p', (v(a_i) \theta v(a_j)) = \text{true}$.*

A constraint context defines pre-condition(s) that must be satisfied by two knowledge chunks submitted to matching in order to be further considered for the purpose of matching. Conditions under which two knowledge chunks are considered as comparable (and thus can be further matched according to the scope and weight contexts) are expressed by the constraints in c . A very common constraint that could be required is the equality constraint $p = p'$, stating that two knowledge chunks kc_i and kc_j are equality-comparable only if their properties p and p' have the same value. For example, with respect to publications of Figure 2, if we want to match only records of publications appeared in the same year, we apply the constraint context $\kappa_{kc_i.year=kc_j.year}(kc_i, kc_j)$ to pairs of knowledge chunks in (AKT EPrints \times Rexa), resulting in the following set \overline{KC} of comparable knowledge chunk pairs.

$$\overline{KC} = \{(eprints : p1, rexa : p1), (eprints : p1, rexa : p2), \\ (eprints : p2, rexa : p1), (eprints : p2, rexa : p2)\}$$

Definition 4. Scope context. *The scope context is defined as a binary operation $\pi_c(kc, kc')$ that, given a pair of knowledge chunks kc and kc' returns the scope-projection $(\overline{kc}, \overline{kc'})$ of kc and kc' under the scope c , with $c \in \{\text{terminological, structural, full}\}$. The scope-projection $(\overline{kc}, \overline{kc'})$ is defined according to the following rules:*

- If $c = \text{terminological}$ then
 - $\overline{kc} = \{n_i \mid \exists a_i(kc), r(a_i) = n_i \vee \exists a_j(kc), v(a_j) = n_i\}$
 - $\overline{kc'} = \{n'_i \mid \exists a_i(kc'), r(a_i) = n'_i \vee \exists a_j(kc'), v(a_j) = n'_i\}$

- If $c = \text{structural}$ then
 - $\overline{kc} = \{\langle n_j, r(a_j) \rangle \mid \exists a_i(kc), r(a_i) = r(a_j), n(kc) = n_j\}$
 - $\overline{kc'} = \{\langle n'_j, r(a'_j) \rangle \mid \exists a_i(kc'), r(a_i) = r(a'_j), n(kc') = n'_j\}$
- If $c = \text{full}$ then
 - $\overline{kc} \equiv kc$
 - $\overline{kc'} \equiv kc'$

The scope context has the purpose of keeping only the features of the knowledge chunk representation that are of interest for comparison and matching evaluation. In particular, the **terminological** scope limits to the terms appearing in the knowledge chunk. In the Matching Variability Framework, we call this set of terms *terminological equipment* of a knowledge chunk, which represents the (unstructured) terminological information available in a knowledge chunk. The **structural** scope produces the set of properties of kc and kc' by cutting off their values. In this case, only the knowledge chunk structure is considered during matching. Finally, the **full** scope considers all the information available in a knowledge chunk. As an example, if we are interested in matching the publications of Figure 2 `eprints:p1` and `rexa:p1` on the basis of their structure only, we apply the scope context $\sigma_{\text{structural}}(\text{eprints} : p1, \text{rexa} : p1)$ that returns the scope-projection $\overline{\text{eprints} : p1} = \{\langle \text{eprints} : p1, \text{author_name} \rangle, \langle \text{eprints} : p1, \text{label} \rangle, \langle \text{eprints} : p1, \text{journal_name} \rangle, \langle \text{eprints} : p1, \text{year} \rangle\}$ and $\overline{\text{rexa} : p1} = \{\langle \text{rexa} : p1, \text{author_name} \rangle, \langle \text{rexa} : p1, \text{journal_name} \rangle, \langle \text{rexa} : p1, \text{year} \rangle\}$.

Definition 5. Weight context. *The weight context $\omega_c(kc) \rightarrow \overline{KC}$ is defined as a unary operation that, given a knowledge chunk kc returns a weighted knowledge chunk \overline{kc} defined according to the weighting set c . The weighting set c is composed by pairs of the form (p_i, w_i) , where p_i is a property name in \mathcal{P} and w_i is a weight in the range $[0,1]$. For each weighting pair (p_i, w_i) , the resulting weighted knowledge chunk \overline{kc} is defined as $\overline{kc} = \{\langle a_i(\overline{kc}), w_i \rangle \mid \exists a_i(kc) = a_i(\overline{kc}) \wedge r(a_i) = p_i\}$.*

The weight context allows to discriminate the relevance to be assigned to axioms for knowledge chunk comparison. The higher the weight is, the highest the relevance of the axiom is. An important usage of the weight context is to assign more relevance to axioms having capability of identifying objects. Axioms having strong identification power can be set to have higher relevance with respect to the others in determining the final level of matching. For example, considering publications of Figure 2, we want to set titles and authors as more relevant properties for identification than book titles and years, while pages should not be taken into account at all. To this end, we define the weight context condition $(\text{author_name}, 1.0)$, $(\text{label}, 1.0)$, $(\text{book_title}, 0.5)$, $(\text{year}, 0.5)$, $(\text{pages}, 0.0)$.

6 The Mapping-Set Contexts

Variability along the mapping-set dimension is addressed by the following contexts.

Definition 6. Ranking context. *The ranking context $\rho_c(\mathcal{M}) \rightarrow 2^{\mathcal{M}}$ is defined as a unary operation that, given a mapping-set \mathcal{M} returns a ordered list $\overline{\mathcal{L}}$ of mappings of \mathcal{M} filtered according to the threshold $c \in [0,1]$. In particular, $\overline{\mathcal{L}}$ is defined as follows:*

$$\overline{\mathcal{L}} = (m_0, m_1, \dots, m_n) \mid \forall m_i, m_j, j > i \Rightarrow SA_j \geq SA_i \geq c$$

The ranking context is used to cut off mappings whose level of semantic affinity is lower than a given threshold. For example, referring to publications of Figure 2, if we execute matching of eprints against rexa by applying the abstract context α_{author_name} , we obtain the following mappings:

$$m_1 = \langle eprints : p1, rexa : p1, 0.8, \emptyset \rangle$$

$$m_2 = \langle eprints : p2, rexa : p1, 0.4, \emptyset \rangle$$

$$m_3 = \langle eprints : p2, rexa : p2, 0.67, \emptyset \rangle$$

This result can be refined by cutting off mappings with a low semantic affinity value, by applying the ranking context $\rho_{0.5}$ that returns only the mappings m_1 and m_3 .

Definition 7. Cardinality context. *The cardinality context $\delta_c(\mathcal{M}) \rightarrow 2^{\mathcal{M}}$ is defined as a unary operation that, given a mapping-set \mathcal{M} returns a mapping-set $\overline{\mathcal{M}} \subseteq \mathcal{M}$ that contains only mappings compatible with the cardinality constraint $c \in \{\text{one}, \text{many}\}$. The resulting mapping-set $\overline{\mathcal{M}}$ is defined according to the following rules:*

- $c = \text{one}$: $\overline{\mathcal{M}} = \{m_i = \langle n(kc_i), n(kc'), SA_i, \mathcal{U}_i \rangle \mid \exists! k', \langle n(kc_i), n(kc'), SA_i, \mathcal{U}_i \rangle\}$
- $c = \text{many}$: $\overline{\mathcal{M}} \equiv \mathcal{M}$

The cardinality context regulates the maximum number of target knowledge chunks that can match a single source knowledge chunk in a mapping-set. This number (called cardinality) ranges from the value ‘unbounded’ allowing to keep all the mappings discovered during the matching execution to ‘exactly one’, which takes only the “best matching” mapping, where the notion of best matching is defined according to the requirements of the matching task at hand. With respect to the previous example, if we do not apply any ranking context, the knowledge chunk eprints:p2 matches with both rexa:p1 and rexa:p2. This situation can be handled by applying the cardinality context δ_{one} and by selecting as best matching element the one with the highest semantic affinity. This results in keeping only the mapping $\langle eprints : p2, rexa : p2, 0.67, \emptyset \rangle$.

Definition 8. Granularity context. *The granularity context $\gamma_c(\mathcal{M}) \rightarrow 2^{\mathcal{M}}$ is a unary operation that, given a mapping-set \mathcal{M} returns a mapping-set $\overline{\mathcal{M}} \subseteq \mathcal{M}$ defined according to the granularity condition $c \in \{\text{simple}, \text{complex}\}$. The resulting mapping-set $\overline{\mathcal{M}}$ is defined according to the following rules:*

- $c = \text{simple}$: $\overline{\mathcal{M}} = \{m_i = \langle n(kc), n(kc'), SA_i, \mathcal{U}_i \rangle \mid m_i \in \mathcal{M}, \mathcal{U}_i = \emptyset\}$
- $c = \text{complex}$: $\overline{\mathcal{M}} = \{m_i = \langle n(kc), n(kc'), SA_i, \mathcal{U}_i \rangle \mid m_i \in \mathcal{M}, \mathcal{U}_i \neq \emptyset\}$

The granularity context determines the kind of mapping that holds between two knowledge chunks in a mapping-set, ranging from a simple correspondence at the whole knowledge chunk level to more complex mappings specifying also mapping rules, which state how to transform the axioms of one knowledge chunk into the matching ones of the other knowledge chunk. As an example we consider a mapping m_i between `eprints:p2` and `rexa:p2`. A complex granularity mapping specifies the following mapping rules: $(eprints : p2).author_name \Leftrightarrow (rexa : p2).author_name$, $(eprints : p2).label \Leftrightarrow (rexa : p2).book_title + (rexa : p2).label$, $(eprints : p2).book_title \Leftrightarrow NULL$, $(eprints : p2).year \Leftrightarrow (rexa : p2).year$, $NULL \Leftrightarrow (rexa : p2).pages$. The mapping rules state that the label of `eprints:p2` corresponds to the concatenation of `book_title` and `label` of `rexa:p2`; the `book_title` in `eprints:p2` does not have a corresponding value in `rexa:p2`; pages in `rexa:p2` does not have a corresponding value in `eprints:p2`.

7 Matching Semantic Web Data with Contexts

In this section, we show an instantiation of the Matching Variability Framework in HMatch 2.0 [20]. HMatch 2.0 is developed as a flexible matching suite where a number of matching techniques are implemented and organized in different modules providing linguistic (HMatch(L)), concept (HMatch(C)), and instance (HMatch(I)) matching techniques. These HMatch 2.0 modules can be differently combined to provide four matching models, namely *surface*, *shallow*, *deep*, and *intensive*, which allows the implementation of the contexts along the matching-dataset variability dimension. Finally, the mapping-set contexts can be realized by proper configuration of HMatch(M), the mapping-manager module of HMatch 2.0. As an example of matching semantic web data with contexts, we report the HMatch 2.0 performance in the OAEI 2009 instance matching contest, where we exploited HMatch 2.0 for matching the whole AKT Eprints and REXA sources. Goal of this matching execution was to find the bibliographic records referred to the same real publications between 2000 and 2003. This conceptual target corresponds to the contexts $\langle -\alpha, +\chi, -\kappa, +\pi, +\omega, +\rho, +\delta, +\gamma \rangle$, which has been obtained with the combination of the properly configured HMatch 2.0 techniques/modules shown in Figure 5.

As a first step, we translated the original RDF datasets into a collection of knowledge chunks \mathcal{D} . The KC-Wrap tool embeds functionalities for the derivation of knowledge chunks from OWL ontologies and RDF repositories. Moreover, KC-Wrap also implements the abstract and concrete contexts defined in our Matching Variability Framework. We exploited a concrete context in order to limit the matching task to data referred to the years between 2000 and 2003. As a result of this step, the source-dataset \mathcal{D} containing all the records in AKT Eprints and REXA has been transformed into a smaller dataset $\overline{\mathcal{D}}$ (i.e., matching-dataset) containing only the bibliographic records of interest for the considered matching

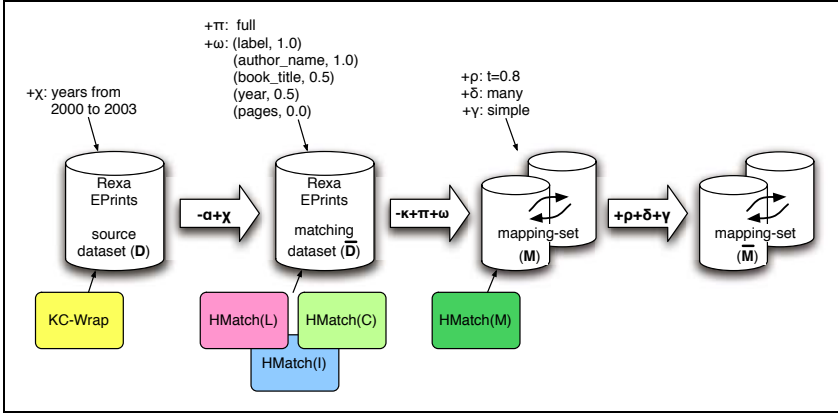


Fig. 5. Instantiation of the Matching Variability Framework in HMatch 2.0

target. Then, we exploited a **full** scope context, corresponding to the intensive matching model of HMatch 2.0, and a weight context in order to configure the matching process. In particular, the full scope-projection has been chosen in order to take into account all the information available about bibliographic records, while a weight context has been enforced to set properties `label` and `author_name` as the most relevant for object identification, followed by `year` and `book_title` (see Figure 5). The value of the property `pages` has not been taken into account, by setting its weight to 0.0. According to this configuration, the matching process is executed, leading to a resulting mapping-set \mathcal{M} . Then, HMatch(M) is exploited to select the mappings with a semantic affinity value greater than or equal to 0.8 (i.e., mapping ranking context). Moreover, we adopted a **many** cardinality context and a **simple** granularity context. The cardinality represents the fact that we can have more than one record representing the same publication in the original source-dataset, while the choice of the granularity context was indicated by OAEI 2009 regulations. The resulting mapping-set $\bar{\mathcal{M}}$ contains the bibliographic records referred to the same real publications and it has been validated against the set of expected mapping provided by OAEI 2009, obtaining a precision of 0.95 and a recall of 0.46, that is the second best performance of OAEI 2009⁵.

8 Concluding Remarks

In this paper, we discussed the notion of matching variability and we presented the Matching Variability Framework for its formal representation and classification. An example of instantiation of this framework in our matching system HMatch 2.0 has been described by considering a test case of bibliographic records of OAEI 2009. Ongoing work is mainly devoted to the integration in the HMatch

⁵ <http://islab.dico.unimi.it/content/oeai2009>

2.0 system of the wrapping tools and related contexts at the source-dataset level. Moreover, we plan to use the Matching Variability Framework for a comparative analysis of recently developed instance matching tools.

References

1. Nikolov, A., Uren, V., Motta, E., Roeck, A.D.: Handling Instance Coreferencing in the KnoFuss Architecture. In: Proc. of the 1st ESWC Int. Workshop on Identity and Reference on the Semantic Web (IRSW 2008), Tenerife, Spain (2008)
2. Rahm, E., Bernstein, P.A.: A survey of approaches to automatic schema matching. *The VLDB Journal* 10(4), 334–350 (2001)
3. Euzenat, J., Shvaiko, P.: *Ontology Matching*. Springer, Heidelberg (2007)
4. Isaac, A., van der Meij, L., Schlobach, S., Wang, S.: An Empirical Study of Instance-Based Ontology Matching. In: Aberer, K., Choi, K.-S., Noy, N., Allemang, D., Lee, K.-I., Nixon, L.J.B., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber, G., Cudré-Mauroux, P. (eds.) *ASWC 2007 and ISWC 2007*. LNCS, vol. 4825, pp. 253–266. Springer, Heidelberg (2007)
5. Bleiholder, J., Naumann, F.: Data Fusion. *ACM Computing Surveys* 41(1) (2008)
6. 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)
7. Jean-Mary, Y.R., Shironoshita, E.P., Kabuka, M.R.: Ontology Matching with Semantic Verification. *Web Semantics: Science, Services and Agents on the World Wide Web* 7(3), 235–251 (2009)
8. Li, J., Tang, J., Li, Y., Luo, Q.: RiMOM: A Dynamic Multistrategy Ontology Alignment Framework. *IEEE Transactions on Knowledge and Data Engineering* 21(8), 1218–1232 (2009)
9. Nagy, M., Vargas-Vera, M., Motta, E.: Managing Conflicting Beliefs with Fuzzy Trust on the Semantic Web. In: Gelbukh, A., Morales, E.F. (eds.) *MICAI 2008*. LNCS (LNAI), vol. 5317, pp. 827–837. Springer, Heidelberg (2008)
10. Doshi, P., Thomas, C.: Inexact Matching of Ontology Graphs Using Expectation-Maximization. In: Proc. of the 21st National Conference on Artificial Intelligence (AAAI 2006), Boston, Massachusetts, pp. 1277–1282 (2006)
11. Cruz, I.F., Antonelli, F.P., Stroe, C.: AgreementMaker: Efficient Matching for Large Real-World Schemas and Ontologies. In: Proc. of the 35th Int. Conference on Very Large Data Bases (VLDB 2009), Lyon, France, pp. 1586–1589 (2009)
12. Zimmermann, A., Krötzschand, M., Euzenat, J., Hitzler, P.: Formalizing Ontology Alignment and its Operations with Category Theory. In: Proc. of the 2006 Conference on Formal Ontology in Information Systems, Amsterdam, The Netherlands, pp. 277–288 (2006)
13. Euzenat, J.: Algebras of Ontology Alignment Relations. In: Sheth, A.P., Staab, S., Dean, M., Paolucci, M., Maynard, D., Finin, T., Thirunarayan, K. (eds.) *ISWC 2008*. LNCS, vol. 5318, pp. 387–402. Springer, Heidelberg (2008)
14. Gal, A., Martinez, M.V., Simari, G.I., Subrahmanian, V.S.: Aggregate Query Answering under Uncertain Schema Mappings. In: Proc. of the IEEE Int. Conference on Data Engineering (ICDE 2009), Washington, DC, USA, pp. 940–951 (2009)
15. Meilicke, C., Völker, J., Stuckenschmidt, H.: Learning Disjointness for Debugging Mappings between Lightweight Ontologies. In: Gangemi, A., Euzenat, J. (eds.) *EKAW 2008*. LNCS (LNAI), vol. 5268, pp. 93–108. Springer, Heidelberg (2008)

16. Castano, S., Ferrara, A., Lorusso, D., N ath, T.H., M oller, R.: Mapping Validation by Probabilistic Reasoning. In: Bechhofer, S., Hauswirth, M., Hoffmann, J., Koubarakis, M. (eds.) *ESWC 2008*. LNCS, vol. 5021, pp. 170–184. Springer, Heidelberg (2008)
17. Castano, S., Ferrara, A., Montanelli, S.: The iCoord Knowledge Model for P2P Semantic Coordination. In: *Proc. of the 6th Conference of the Italian Chapter of AIS*, Costa Smeralda (Nu), Italy (2009)
18. Lapouchnian, A., Mylopoulos, J.: Modeling Domain Variability in Requirements Engineering with Contexts. In: *Proc. of the 28th Int. Conference on Conceptual Modeling (ER 2009)*, Gramado, Brazil. Springer, Heidelberg (2009)
19. Van Lamsweerde, A.: Goal-Oriented Requirements Engineering: A Guided Tour. In: *Proc. of the 5th IEEE Int. Symposium on Requirements Engineering (RE 2001)*, Washington, DC, USA (2001)
20. Castano, S., Ferrara, A., Montanelli, S.: Matching Ontologies in Open Networked Systems: Techniques and Applications. *Journal on Data Semantics, JoDS V* (2006)

GRUve: A Methodology for Complex Event Pattern Life Cycle Management

Sinan Sen and Nenad Stojanovic

FZI Research Center for Information Technology
Haid-und-Neu-Straße 10-14, 76131 Karlsruhe, Germany
{sinan.sen,nstojano}@fzi.de
<http://www.fzi.de/ipe>

Abstract. With the rising importance of recognizing a complex situation of interest near real-time, many industrial applications have adopted Complex Event Processing as their backbone. In order to remain useful it is important that a Complex Event Processing system evolves according to the changes in its business environment. However, today's management tasks in a Complex Event Processing system related to the management of complex event patterns are performed purely manually without any systematic methodology. This can be time consuming and error-prone.

In this paper we present a methodology and an implementation for the complex event pattern life cycle management. The overall goal is the efficient generation, maintenance and evolution of complex event patterns. Our approach is based on a semantic representation of events and complex event patterns combined with complex event pattern execution statistics. This representation enables an improved definition of relationships between patterns using semantic descriptions of patterns and events.

1 Introduction

Complex Event Processing (CEP) is the analysis of events from different event sources in near real-time in order to generate immediate insight and enable immediate response to changing business conditions [1]. For example *Transportation and Logistics*, *Financial Front Office*, *Telecommunication* or *Customer Risk Management* are successful application areas of CEP. Events are classified into simple (atomic) events and complex events. According to Chandy et. al. [2] an event indicates a significant change in the state of the universe. Sensor data, network signals, credit card transactions or application processing information are examples for simple events. A simple event is atomic and does not contain further events. In comparison to a simple event, a complex event is composed of other simple or complex events. For example, a credit card fraud event is detected from incoherent use of a single credit card in different places by combining these events into a complex event.

To detect a pattern over events properly is the most important capability of CEP systems. This capability enables a just-in-time reaction to occurred situations. In order to recognize these situations so called complex event patterns

(CEPATs) have to be defined. These patterns¹ are used in order to process the events and aggregate them to more high level complex events. A pattern is an expression formed by using a set of events (either simple or complex) and a set of event operators³. They resemble knowledge about the reactivity of the system. For example the pattern²

(A AND B) happen WITHIN 2 Minutes

contains two events *A* and *B*, a logical Operator *AND* and a window operator *WITHIN*.

In order to cope with the evolving nature of business environments we need effective and efficient support for advanced CEPAT management. However, today's management tasks in a CEP system related to the generation, maintenance and evolution of CEPATs are performed manually without any systematic methodology and tool support. So far there exists no academic approach dealing with the management and evolution of CEPATs. Also the vendors providing CEP systems neglect the issue of management and evolution. They are focused rather on runtime than on design time issues. CEPATs can be compared to business rules in the sense that they both are indispensable for today's business. In the business rules domain the management and the evolution of rules have been proven to be crucial in order to optimize the process of rule generation, modification and evolution (see [4],[5],[6],[7]). However, a straightforward application of existing business rule management systems to CEPAT management is not a viable solution. This is due to the nature of CEPATs. Moreover, the existing business rule evolution approaches do not tackle capabilities such as recommendation of new rules, the reusability of existing rules, especially for large rule sets, and finally the usage-driven evolution of rules which is part of our methodology.

In this paper we present a methodology for the management of CEPATs which supports the whole life cycle of a CEPAT: starting from its generation, throughout its usage including its evolution. The methodology treats CEPATs as knowledge artefacts that must be acquired from various sources, represented, shared and evaluated properly in order to enable their evolution. We argue that such a methodology is a necessary condition for making a CEP system efficient and keeping it alive. In a nutshell our approach is a semantic-based representation which enables us to a) make the relationships between CEPATs explicit in order to reuse existing pattern artefacts b) automatically suggest improvements in a CEPAT, based on necessity derived from usage statistics.

The paper is organized as follows: In section 2 we present related work. In section 3 we define requirements for a CEPAT management methodology followed by section 4 where we describe our methodology consisting of different phases including the semantic CEPAT representation and where we outline its importance. In section 5 we present the implementation state of our work and in section 6 we conclude the paper and give an outlook for the continuation of this line of research.

¹ Pattern and complex event pattern are used synonymously in this article.

² For the sake of convenience the pattern is represented in a pseudocode form.

2 Related Work

In the followed section we discuss related work from different fields of research relevant for this paper, namely current approaches in event representation, business rule evolution and rule management. We also take a look at the existing CEP systems.

Event representation: There exist different approaches for event representation. Distributed publish/subscribe architectures like presented in [8], [9] and [10] represent an event as a set of typed attributes. Each individual attribute has a type, name and value. The event as a whole has purely syntactical and structural value derived from its attributes. Attribute names and values are simple character strings. The attribute types belong to a predefined set of primitive types commonly found in programming languages. In our opinion this kind of event model is not sufficient for CEPAT management since it does not provide any explicit semantics. The only justification for choosing this typing scheme is the scalability aspect. The event stream engine AMIT (see [11] and [12]) is targeting the high-performance situation detection mechanism. Via the offered user interface one can model business situations based on event, situations, lifespans and keys. Within AMIT an event is specified with a set of attributes and it is the base entity. All events belong to a certain class (a class contains events with similar characteristics) and have relationships to other events. Attributes can be references to other objects as well. This approach is close to ours in the sense that it describes the importance of generalization, specialization and relationships of events.

Event representation in existing CEP systems: There is a large number of CEP systems, both academic and commercial. The Aurora/Borealis [13] system has the goal to support various kinds of real-time monitoring applications. Data items are composed of attribute values and are called tuples. All tuples on a stream have the same set of attributes. The system provides a tool set including a graphical query editor that simplifies the composition of streaming queries using graphical boxes and arrows. The Cayuga [14] system can be used to detect event patterns in the event stream as well. An event here is a tuple of attribute value pairs according to a schema. The goal of the Stream [15] project is to be able to consider both structured data streams and stored data together. The Esper [16] CEP engine supports events in XML, Java-objects and simple attribute value pairs. There are also many commercial CEP vendors like Tibco, Coral8/Aleri, StreamBase or Progress Apama providing tool suites including development environments and a CEP engine. Most of the described systems use different SQL-like or XML based complex event pattern definitions, which makes it difficult to understand and read them. They do not provide a semantic model for events and event patterns either. The management of event patterns is restricted to create, modify and delete. Furthermore they do not use any underlying management methodology and do not tackle the issues of reusability and evolution of patterns.

(Business) Rule evolution and its importance: The Agile Business Rule Development methodology [17] is aimed at harvesting rules and producing an executable (albeit incomplete) rule set as early as possible. It is based on five cycles: Harvesting, Prototyping, Building, Integration and Enhancing. Within the cycles, the activities are Discovery, Analysis, Authoring, Validation and Deployment. The methodology considers mostly the harvesting of rules and does not consider their evolution. Also the refinement and the reusability of rules are not part of the methodology. There are also many other approaches in the area of (business) rules evolution or management like [4], [5], [6], [7], [15], [17]. They consider either some aspects of (business) rules or describe the necessity of evolution and maintenance. We of course incorporate relevant aspects from these works in our methodology where applicable.

3 Requirements: What Do We Need and Why Is It Important?

Compared to other IT-systems, CEP systems still lack in support of tools allowing users to reconfigure a system easily or to refactor services and components [18]. This is also valid for CEPAT generation, maintenance and management. David Luckham³ considers the validation and the management of event processing rules especially when dealing with large sets of event processing rules as one of the challenges in the area of CEP [19]. Luckham defines CEPAT management as writing correct patterns, their efficient execution, correct changes, ensuring logical consistency and absence of redundancies.

From our point of view the management of CEPATs requires two basic functionalities: efficient generation of patterns, including the incremental development of new patterns and the continual improvement of existing patterns. These functionalities in turn require methods and tools for: checking the validity of new patterns, including the entire repository, reuse of existing patterns and discovery of the need for changes in patterns and realizing them. In the following we briefly describe each of these requirements:

- **Ensure an anomaly-free repository:** Anomalies in general are symptoms of probable errors. Efficient generation of patterns means that the newly created patterns semantically suit to the existing patterns. This can be ensured through resolving two types of anomalies: contradiction and redundancy. In order to ensure an anomaly-free repository a formal model is needed that is expressive enough to detect anomalies.
- **Deal with weak information needs and reuse of existing knowledge:** The creation of new patterns can be augmented by an incremental approach which will enable smooth refinements of existing patterns in order to sustain patterns being as relevant as possible for the problem at hand.

³ David Luckham is one of the most prominent experts in the area of CEP - <http://www-ee.stanford.edu/~luckham/>

- **Enable continual improvement:** As already explained patterns will change in time and there is a need for methods of continually updating patterns in order to ensure their relevance for new situations. CEP systems are designed for near real-time environments. Iterative refinement of the generated CEPATs enables a smooth modification of an initial CEPAT in order to make it as relevant as possible to the given task of the CEPAT. A continual improvement of the CEPATs enables an adaptation of the patterns to the internal or external changes in order to keep the CEP system useful. The methodology should also consider the effective detection of problems in the usage of patterns, to enable suggesting how to evolve them.

These requirements cause some additional requirements for the implementation of the methodology that are described below:

- **High level graphical representation of events and patterns:** In order to express rules succinctly in a way that makes them understandable, we need proper graphical development environments. A business expert (a non-technician) should be able to express her/his need for a complex event pattern by using this environment. The complexity of underlying CEP engines needs to be hidden from the user in order to increase the usability of the tool. Furthermore the tool must support the refinement and the reusability of patterns in a graphical way as well.
- **Independent modeling of events and patterns:** The event and pattern model has a major impact on the flexibility and usability of a CEP system. Nowadays event patterns are mostly realized in XML, Java-Objects or SQL-like CEP languages. However, these representations do not support the management of event patterns. In order to assure that, a model is needed which allows the explicit definition of event and event pattern semantics and which supports reasoning about events and event patterns. This model is used at design time for event and event pattern representation and is transformed into the CEP engine specific languages during the pattern deployment phase.

We believe a methodology and a tool-set based on these requirements will shorten the time of the CEPAT development, improve the quality of CEPATs and enable also non-domain experts to develop, maintain and evolve CEPATs. In the next section we present a methodology implementing these requirements.

4 GRUVe: The Methodology

We define the CEPAT life cycle management as a process covering the generation, the refinement, the execution and the evolution of CEPATs. The focus is on supporting the user by creating new patterns, increasing the reusability of existing pattern artefacts and to give hints for pattern evolution. As illustrated in figure 1, the GRUVe⁴ methodology consists of four main phases. These phases

⁴ Acronym for Generation Refinement Usage eVolution of complex (e)vent patterns.

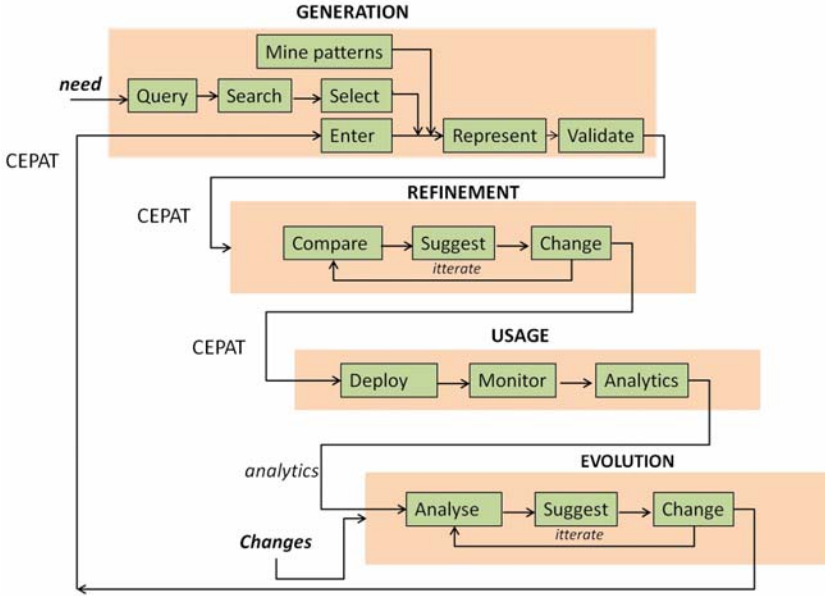


Fig. 1. Phases of the CEPAT life cycle

form a feedback loop enabling continual flow of information collected in the entire life cycle of a CEPAT. The main idea is to enable non-technicians to *search* incrementally for the most suitable form of requested CEPATs and to continually improve their quality taking into account changes that might happen in the internal or external world. Obviously the basis for the development of the methodology were the requirements given in the previous section. The methodology has been influenced by our past work in the area of knowledge management and ontology evolution [20], [21]. Below we describe these phases in more detail.

4.1 Generation Phase

The life cycle of CEPATs starts with their development which is classified into three categories (see figure 1 block *Generation*). The simplest way to develop a CEPAT is a manual development which can be done only by business experts. However, we cannot expect that an arbitrary user spends time finding, grouping and ordering the events in order to create their desired CEPAT. In order to do that the user must be aware of the way of combining events, he/she must find the right type of events, foresee and solve the intermediate conflicts that might appear and order events in a right way. A more user-oriented approach can be obtained by reusing existing CEPATs, the so-called search-based approach. Here, the users should be able to specify their complex goals without considering how they can be realized. By selecting *similar* existing CEPATs at the start of the CEPAT development process, the users could realize their request much faster. For instances if the user intends to generate a new pattern for a fraud situation

he/she can search for events which belong to the type *fraud* from a certain event source. In that way the generation of a CEPAT could be realized as a straightforward search method, where the specificities of the CEPAT domain are used to define a search space and to restrict the search in this space. Besides CEPATs developed by experts or users explicitly, there are also implicit CEPATs in the domain, reflected in the behavior of the system. These can be discovered through the analysis of log files, etc. Various data mining methods can be used for the extraction of such knowledge. Independent of how CEPATs are identified, in order to deal with them they have to be represented in a suitable format which supports their management. Therefore, the generated complex event patterns must be represented in an internal, CEP platform independent way.

RDFS-based complex event pattern representation. A well defined event pattern model has a major impact on the flexibility and usability of CEP tools [18]. Contemporary event and pattern models lack the capability to express the event semantics and relationships to other entities. Although the importance of such a semantic event representation is demonstrated in practice [12], there does not exist any systematic work on this topic so far. Still most of the existing event models consider an event as an atomic unstructured or semistructured entity without explicitly defined semantics. Also the existing CEP languages are product specific languages designed solely for the matching process rather than for their own management and evolution. Our semantic model for event and pattern representation is based on RDFS⁵. RDFS in general has the advantage that it offers formal and explicit definitions of relations between concepts. We use our RDFS based pattern representation at different stages within the methodology: increasing the reusability of existing pattern artefacts, validation of defined patterns and identification of relations between two patterns in order to evolve them. We believe the explicit representation of events through RDFS will enable CEP tools to provide a qualitatively new set of services such as verification, justification and gap analysis. These systems will be able to weave together a large network of human knowledge and will complement this capability with machine processability. Various automated services will then aid users in achieving their goals by accessing and providing information in a machine understandable form. It is important to note that semantic-based representations not only define information but also add expressiveness and reasoning capabilities. In general, reasoning tasks can be used to verify a model (i.e. a set of CEPATs). Consistency checking, detection of redundancies and refinement of properties are just some of these reasoning capabilities.

The upper-level ontology contains a set of concepts *Event*, *EventOperator*, *EventSource*, *EventType* and a set of event properties. Each property may have a domain (denoted by $\sqsubseteq \exists$) concept as well as a range (denoted by \sqsupseteq) concept (see figure 2). These concepts can be specialized in order to define new types, sources and operators.

⁵ RDF Schema (RDFS) is an extensible knowledge representation language providing basic elements for the description of ontologies.

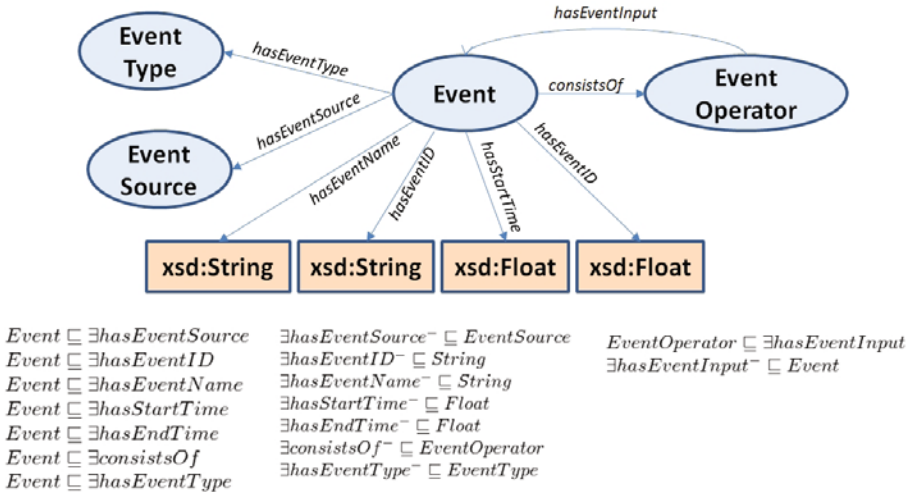


Fig. 2. Upper-level event and CEPAT ontology

Definition 1: An event is defined as a 7-tuple $E := (I, N, ST, ET, ES, EO, T)$ where:

- I is a numeric event id
- N is a alphanumeric event name
- ST is numeric event start time of an event
- ET is numeric event end time of an event
- EO is a reference to the event operator concept
- ES is a reference to the event source concept
- T is a reference to the the event type concept

A concrete event occurrence, simple or complex, is an instantiation of E . It has an unique id and an event name. The start time indicates when the event occurred. The end time of an event is relevant only for complex events and indicates the triggering of a complex event which might be different from the start time for an event with a duration. In comparison to a complex event that contains a complex event pattern via a reference to the concept EO a simple event does not contain any reference to the concept EO . In this sense the concept EO indicates whether an event is simple or complex with a proper complex event pattern.

In order to classify events of similar characteristics we use event types and event sources. The event type characterizes a class of event objects.

Definition 2: The event type is defined as a tuple $T := (H_t, A_t)$ where:

- H_t is an acyclic event type hierarchy
- A_t is a set of event type specific attribute $\langle at_1, \dots, at_n \rangle$

An event source is an entity that sends events e.g., a software module, a sensor, a web service etc. The definition of the event source is similar to the definition of the event type.

Definition 3: The event source is defined as a tuple $ES := (H_{es}, A_{es})$ where:

- H_{es} is an acyclic event source hierarchy
- A_{es} is a set of event type specific attribute $\langle as_1, \dots, as_n \rangle$

A complex event is composed of an event operator via the *consistsOf* property. An event operator is further classified into operator classes according to the nature of the operator. Currently the model contains operators defined in [3]. An event operator again has events as input. This allows us to build more complex event patterns recursively.

Definition 4: The event operator is defined as a tuple $EO := (H_{eo}, A_{eo})$ where:

- H_{eo} is an acyclic event operator hierarchy
- A_{eo} is a set of event operator specific attribute $\langle ao_1, \dots, ao_n \rangle$

One of the key idea of the approach is having a well defined representation of the complex event patterns in order to recognize different relations between them. The information about the relation can be used for getting more precise information on top of the pattern statistics knowledge.

A pattern is formed as a tuple (E, EO) where E is a set of events and EO is a set of event operator.

Let:

- Γ be the set of all CEPAT triples within the Knowledge Base (KB) representing all defined CEPATs.
- ϵ_i represent all event triples and σ_i all operator triples created for a CEPAT instance.
- τ_i be the set triples of a CEPAT instance containing all event and operator triples, i.e. $\tau_i = \epsilon_i \cup \sigma_i$.

Then a CEPAT instance can be seen as a finite subset of Γ , i.e. $\tau_i \subseteq \Gamma$. On top of that definition we define the relations between two CEPATs:

- subsumption: A complex event pattern $CEPAT_1$ subsumes another complex event pattern $CEPAT_2$, i.e. $\tau_1 \subset \tau_2$
- overlapping: A complex event pattern $CEPAT_1$ have a common overlap with another complex event pattern $CEPAT_1$, i.e. $\exists x, y \in \epsilon : (x \subset \tau_1 \wedge x \subset \tau_2) \wedge (y \subset \tau_1 \wedge y \not\subset \tau_2)$
- disjointness: Two complex event patterns do not have an overlap, i.e. $\tau_1 \cap \tau_2 = \emptyset$
- equalness: Two complex event patterns represent the same situation, i.e. $\tau_1 \equiv \tau_2$

Regarding the *Generation* phase, the main strength of our approach lies in the presented semantic-based nature of our CEPAT representation which enables formal representation of the relations between existing CEPATs and consequently better understanding of that information space. It increases the reusability of existing patterns enabling a faster development of new patterns.

4.2 Refinement Phase

As we already mentioned the process of creating CEPATs is a kind of weakly-defined search operation in an information space rather than a search with very precisely defined queries. Lessons learned from the Information retrieval community show that the right mechanism for the former case is the incremental refinement of queries [20], based on various feedbacks that can be obtained in that process. It would mean that the main task in the CEPAT creation process is not the very precise formulation of a pattern in the first place but an iterative improvement of the CEPAT. This corresponds to the second requirement from section 3. As presented in figure 1 this phase has the CEPAT as input created in the previous phase. It allows the user to fine tune the CEPAT if he/she is not sure about the quality of the created CEPAT. Let us assume our goal is to build new patterns N1 and N2. We further assume that there exist two CEPATs, P1 and P2, in the repository⁶.

```
P1- Stock.VW.value=117 AND Stock.Daimler.value=34.78
    AND Stock.Porsche.value=47.90
P2- Blog.Twitter.Tweet.Person="Barack Obama" AND
    Blog.Twitter.Tweet.Person="Nicolas Sarkozy"
```

We can use the previously defined relations in order to demonstrate the refinement for the following new patterns and show how relevant hints can be given about related patterns.

```
N1- Stock.VW.value=117 AND Stock.Porsche.value=47.90
N2- Blog.Twitter.Person="Angela Merkel"
    AND Blog.Twitter.Person="Barack Obama"
```

The following relations exist between the four patterns:

- $subsumption(P1, N2)=true$
- $disjoint(P1, N2)=true$
- $disjoint(P2, N1)=true$
- $overlapping(P2, N2)=true$

The result of this relation detection is that in the first case P1 is presented to the user in order to be reused. In the second case P2 can be shown in order to inform the user that there exists another pattern that can be relevant for him as well. By using the RDF relation *RDF:instanceOf* we also can identify patterns that are different at the instance level but equivalent at the class level. For instance if we

⁶ The patterns in this example are presented in a pseudocode form.

change the pattern artefact *Stock.VW.value=117* into *Stock.VW.value=118* we are still able to detect the subsumption relation based on the class information although they differ at the instance level.

4.3 Usage Phase

Once created, CEPATs must be deployed in a CEP engine, after being transformed into the corresponding syntax. However in order to ensure the quality of created patterns, it is necessary to monitor what happens to a CEPAT or a set of CEPATs in a CEP engine at runtime. Nowadays there is no monitoring of the defined CEPATs e.g., if they had been executed in a CEP engine. It is not obvious to see how often certain patterns have been triggered and which parts of the pattern have been executed how often. It is not available either which patterns are going to be executed next. However we believe information of how often a pattern was triggered or how high the current degree of fulfillment is might be essential for the pattern evolution. The goal of this phase is to track as much as possible of this information and process it in the context of the defined CEPATs. For this reason we think a component may be developed that might capture all the pattern related information. These statistics can be used within the *Evolution* phase in order to evolve and optimize a pattern.

4.4 Evolution Phase

A pattern that has not become rapidly obsolete must change and adapt to the changes in its environments, user's needs, etc. Therefore, if a pattern aims at remaining useful it is essential that it is able to accommodate the changes that will inevitably occur. Developing patterns and their applications is expensive but evolving them is even more expensive. The experiences show that the traditional software system maintenance costs exceed the development costs by a factor of between two and four. There is no reason to assume this should be any different for CEP systems when they are used during a longer period of time. Facilitating those changes is complicated if large quantities of events and CEPATs are involved. The Evolution phase is responsible for coping with changes that may impact the quality of a CEPAT. The causes of the changes usually lay in the highly changeable internal and external factors. In a more open and dynamic business environment the domain knowledge evolves continually. The basic sources that can cause changes in a business system are:

- **The environment:** The environment in which systems operate can change.
- **The user:** Users' requirements often change after the system has been built, warranting system adaption. For example hiring new employees might lead to new competencies and greater diversity in the enterprise which the system must reflect.
- **The process:** The business applications are coupled around the business processes that should be reengineered continually in order to achieve better performances.

The goal of this phase is to use the analytics provided by the Usage phase and suggest to the user some improvements to evaluated CEPATs. While a good design may prevent many CEPAT errors, some problems will not pop out before CEPATs are in use. Therefore, the relationship with the usage of a CEP-based system is paramount when trying to develop useful CEPATs and cannot be neglected. One of the key ideas of the approach is having a well defined representation of the complex event patterns in order to recognize different relations between them. The information about the relations can be used for getting more precise information on top of the pattern statistics knowledge.

The most important analysis is the comparison with the usage data of related patterns. We give here an example: Let us assume that P1 and N1 (from the previously section) are in the event repository. Let us further assume that P1 has been triggered 10 times in a period and N1 1000 times. Obviously there might be a problem in the usage of the pattern P1. What can be concluded by comparing usage data from these two patterns is that part *Stock.VW.value=117 AND Stock.Porsche.value=47.90* in the pattern N1 had been fulfilled many times but the combination with the simple subpattern *Stock.Daimler.value=34.78* is rather critical. The system can discover such a discrepancy and suggest the user to replace this event. Moreover, the system can suggest which similar event is the most appropriate one. Obviously, judging whether a pattern is still correct based on its usage requires a kind of metrics. Our next step will be to develop a comprehensive notion of the quality of a CEPAT, which will alleviate the Evolution phase.

5 Implementation of the Methodology

The current version of our CEPAT life cycle management environment supports the Generation and Refinement phase implemented as a web application. The implementation supports the transformation of patterns and events into our RDFS representation, as described in section 4.1.

The CEPAT management environment consists of the CEPAT Input Section, the CEPAT Design Section, the CEPAT Refinement Section and the CEPAT Statistics Section (see figure 3). The pattern input section provides the event nodes and the operator nodes in order to build a new CEPAT.

Within the pattern design section new patterns can be created. Patterns can be generated by selecting the required event or operator nodes from the pattern Input Section and connect them. Event nodes should be connected to an operator node where each event node is connected to at most one operator node. An operator node can be connected to several event nodes but can only be connected to either an action node or another operator node. The pattern generated in fig. 3 shows the pattern N1 from section 4.2. The results of the refinement are shown in the CEPAT Refinement Section. It shows the pattern P1 from section 4.2. As soon as the user starts building a new pattern our system calculates the next similar patterns related to his current situation based on the relations. The results are presented in a graphical way. The user can see the details of the presented patterns by clicking on them and reuse parts or all of the selected pattern.

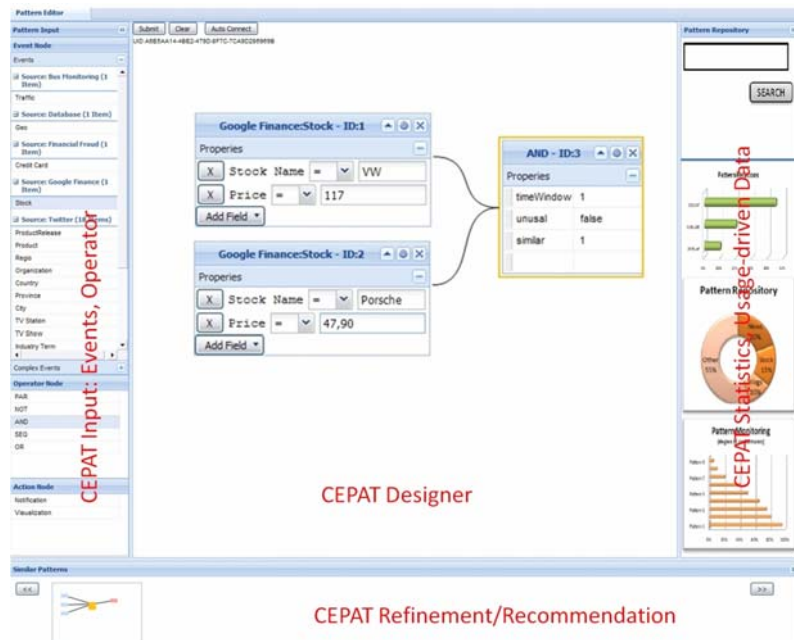


Fig. 3. Complex event pattern management environment including the pattern N1 from section 4.2

6 Conclusion and Future Work

The most important capability of CEP (Complex Event Processing) systems is to detect an occurred situation properly. Today's management tasks in CEP systems related to the evolution of complex event patterns (CEPATs) are performed manually without any systematic methodology and tool support. So far there is not any academic approach dealing with the management and evolution of CEPATs. In this paper we presented a methodology for CEPAT life cycle management supporting the generation, maintenance and evolution of CEPATs. In a nutshell the approach is a semantic-based representation of CEPATs which enables us to make the relationships between CEPATs explicit in order to reuse existing pattern artefacts and automatically suggest improvements for a CEPAT, in the case that it is necessary, based on usage statistics. We present the current implementation status of our methodology that supports the generation and the refinement phase.

Our long-term goal is to underline the importance of a methodology for CEPAT management through a proper reference implementation. Therefore we will extend the phases of our methodology presented in this paper in the following way:

- **Generation phase:** To cover also the implicit knowledge within the domain we will use existing data mining technologies for generating CEPATs automatically. They can be adapted by the expert later on. Furthermore we will develop advanced CEPAT search strategies in order to find existing pattern knowledge in the pattern repository.
- **Refinement phase:** Beside the reusability of patterns it is also useful to give experts hints about missing patterns for closing the pattern knowledge gap and to be up-to-date.
- **Usage phase:** In order to support the evolution we will develop a statistical manager component that monitors the execution of patterns and build some proper analytics for the evolution phase.
- **Evolution phase:** For the evolution of patterns we will introduce quality metrics for patterns in order to identify CEPAT candidates for update.

Acknowledgments. The research presented in this paper was partially funded by the European Commission in the project VIDJ (http://www.vidj-project.eu/), EP-08-01-014. We would like to thank to Ljiljana Stojanovic for her comments and contribution to this work. We also would like to thank Dominik Riemer, Ruofeng Lin and Weiping Qu for their contribution to the implementation.

References

1. Luckham, D.C.: *The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems*. Addison-Wesley Longman Publishing Co., Inc., Boston (2001)
2. Chandy, K.M., Charpentier, M., Capponi, A.: Towards a theory of events. In: DEBS 2007: Proceedings of the 2007 inaugural international conference on Distributed event-based systems, pp. 180–187. ACM, New York (2007)
3. Chakravarthy, S., Mishra, D.: Snoop: An expressive event specification language for active databases. *Data Knowl. Eng.* 14(1), 1–26 (1994)
4. Sobieski, J., Krovvidy, S., McClintock, C., Thorpe, M.: Karma: Managing business rules from specification to implementation. *AAAI/IAAI* 2, 1536–1547 (1996)
5. Wan-Kadir, W.M.N., Loucopoulos, P.: Relating evolving business rules to software design. *J. Syst. Archit.* 50(7), 367–382 (2004)
6. Lin, L., Embury, S., Warboys, B.: Business rule evolution and measures of business rule evolution. In: IWPSE 2003: Proceedings of the 6th International Workshop on Principles of Software Evolution, Washington, DC, USA, p. 121. IEEE Computer Society, Los Alamitos (2003)
7. Lin, L., Embury, S.M., Warboys, B.C.: Facilitating the implementation and evolution of business rules. In: ICSM 2005: Proceedings of the 21st IEEE International Conference on Software Maintenance, Washington, DC, USA, pp. 609–612. IEEE Computer Society, Los Alamitos (2005)
8. 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)
9. Pietzuch, P.R., Bacon, J.: Hermes: A distributed event-based middleware architecture. In: ICDCSW 2002: Proceedings of the 22nd International Conference on Distributed Computing Systems, Washington, DC, USA, pp. 611–618. IEEE Computer Society, Los Alamitos (2002)

10. Aguilera, M.K., Strom, R.E., Sturman, D.C., Astley, M., Chandra, T.D.: Matching events in a content-based subscription system. In: PODC 1999: Proceedings of the Eighteenth Annual ACM Symposium on Principles of Distributed Computing, pp. 53–61. ACM, New York (1999)
11. Adi, A., Etzion, O.: Amit - the situation manager. *The VLDB Journal* 13(2), 177–203 (2004)
12. Adi, A., Botzer, D., Etzion, O.: Semantic event model and its implication on situation detection. In: ECIS (2000)
13. Abadi, D., Carney, D., Çetintemel, U., Cherniack, M., Convey, C., Erwin, C., Galvez, E., Hatoun, M., Maskey, A., Rasin, A., Singer, A., Stonebraker, M., Tatbul, N., Xing, Y., Yan, R., Zdonik, S.: Aurora: a data stream management system. In: SIGMOD 2003: Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data, p. 666. ACM, New York (2003)
14. Brenna, L., Demers, A., Gehrke, J., Hong, M., Oshser, J., Panda, B., Riedewald, M., Thatte, M., White, W.: Cayuga: a high-performance event processing engine. In: SIGMOD 2007: Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data, pp. 1100–1102. ACM, New York (2007)
15. Arasu, A., Babcock, B., Babu, S., Cieslewicz, J., Datar, M., Ito, K., Motwani, R., Srivastava, U., Widom, J.: Stream: The stanford data stream management system. Technical Report 2004-20, Stanford InfoLab (2004)
16. Esper: Esper version 3.2.0, espertech inc. (2009), <http://esper.codehaus.org/> (last visited: January 2010)
17. IBM-ILOG: Agile business rule development methodology (2010), https://www.ibm.com/developerworks/mydeveloperworks/blogs/isis/?lang=en_us (last visited: January 2010)
18. Rozsnyai, S., Schiefer, J., Schatten, A.: Concepts and models for typing events for event-based systems. In: DEBS 2007: Proceedings of the 2007 Inaugural International Conference on Distributed Event-Based Systems, pp. 62–70. ACM, New York (2007)
19. Luckham, D.C.: What's the difference between esp and cep? (August 2006), <http://complexevents.com/?p=103> (last visited: January 2010)
20. Stojanovic, N.: Ontology-based information retrieval. Ph.D. Thesis, University of Karlsruhe, Germany (2005)
21. Stojanovic, L.: Methods and tools for ontology evolution. Ph.D. Thesis, University of Karlsruhe, Germany (2004)

Supporting Semantic Search on Heterogeneous Semi-structured Documents

Yassine Mrabet^{1,2}, Nacéra Bennacer², Nathalie Pernelle¹,
and Mouhamadou Thiam^{1,2}

¹ LRI, Université Paris-Sud 11, INRIA Saclay, F-91893 Orsay cedex, France
{Yassine.Mrabet,Nathalie.Pernelle,Mouhamadou.Thiam}@lri.fr

² SUPELEC Systems Sciences (E3S), F-91192 Gif-sur-Yvette cedex, France
Nacera.Bennacer@supelec.fr

Abstract. This paper presents *SHIRI-Querying*¹, an approach for semantic search on semi-structured documents. We propose a solution to tackle incompleteness and imprecision of semantic annotations of semi-structured documents at querying time. We particularly introduce three elementary reformulations that rely on the notion of aggregation and on the document structure. We present the Dynamic Reformulation and Execution of Queries algorithm (DREQ) which combines these elementary transformations to construct reformulated queries w.r.t. a defined order relation. Experiments on two real datasets show that these reformulations greatly increase the recall and that returned answers are effectively ranked according to their precision.

1 Introduction

The research advances on automating ontology population and document annotation are promising. But even for named entity-based approaches [1, 2] or pattern-based approaches [5] it remains difficult to locate precisely instances since some of them may be blended in heterogeneous semi-structured documents. The granularity of the annotation could be precise, at the term level, or imprecise, at the node level, in a semi-structured document [5]. In the worst case, the annotated unit is the whole document. Semantic imprecision may also appear when associated annotations are not accurate enough (e.g. using *Event* metadata instead of *Conference* metadata). From another hand, annotations are often incomplete since automatic annotators do not find all instances and relations. To alleviate these problems, some semantic search systems try to gather answers satisfying the user query by going beyond the simple use of available metadata. Some approaches [4, 6] deal with semantic imprecision by approximating the concepts and the relations expressed in user queries using an ontology (e.g. exploiting subsomption, contextual closeness, path of semantic relations). Other works combine ontology-based search and classical keyword search [3, 7] in order to deal with incomplete annotations. The use of keywords increases the

¹ SHIRI : Digiteo labs project (LRI, SUPELEC).

recall by retrieving instances that are not reachable using semantic annotations, but some semantic constraints of the query are relaxed.

In this paper, we propose an ontology-based search approach called *SHIRI-Querying*. Our contributions are: (i) a reformulation method to query incomplete and imprecise semantic annotations of semi-structured documents (ii) an order relation that ranks the constructed queries according to their relevance and (iii) a dynamic algorithm which builds and executes reformulated queries w.r.t. the defined order. The *SHIRI-Querying* system uses the standard W3C languages RDF/OWL for representing resources and SPARQL for their querying. It has two main components. The *adapter* is designed to conform the provided annotations to the *SHIRI* annotation model. It uses a set of logical rules and generates automatically the annotations base to be queried. The *Query Engine* processes ontology-based queries and reformulates them using the *SHIRI* annotation model. In this generic model [5] the granularity of the annotation is the document node (e.g. XML and HTML tags). Each node is annotated as containing one or several instances of different concepts of a given domain ontology. This allows bypassing the imprecise localisation of instances at the term level. The annotation model also allows representing structural links between document nodes, which enables dealing with the incompleteness of semantic relations in the provided annotations. We define three elementary query reformulations: the *SetOfConcept* and *PartOfSpeech* reformulations which allow retrieving instances that are aggregated in the same node and the *neighborhood-based* reformulation which allows retrieving instances located in close nodes that may be related by the required semantic relations. Reformulations of the user query are then obtained by combining these elementary transformations. The Dynamic Reformulation and Execution of Queries algorithm (*DREQ*) constructs these combinations and executes them w.r.t an order relation. This order relation gives priority to answers where nodes contain homogeneous instances and answers where nodes are linked by the required semantic relations. In contrast to most approaches which work on answers and/or whole annotated datasets, the answers are ranked as the reformulated queries are constructed. Experiments on two real datasets show that these reformulations greatly increase the recall and that the answers are effectively ranked according to their precision.

2 Annotation Model

Let $\mathcal{O}(\mathcal{C}_O, \mathcal{R}_O, \preceq, \mathcal{D}_O)$ be the domain ontology where \mathcal{C}_O is the set of concepts, \mathcal{R}_O is the set of relations between concepts ($\mathcal{R}_O^f, \mathcal{R}_O^{if}$ are resp. functional and inverse functional relations), \preceq denotes the subsumption relation between concepts or relations and \mathcal{D}_O defines the domain and the range for each relation. The annotation model, denoted $\mathcal{A}(\mathcal{C}_A, \mathcal{R}_A, \preceq, \mathcal{D}_A)$, is generated automatically from the domain ontology. $\mathcal{C}_A = \mathcal{C}_O \cup \mathcal{C}_S$, $\mathcal{R}_A = \mathcal{R}_O \cup \mathcal{R}_S$. \mathcal{C}_S and \mathcal{R}_S are the concepts and the relations defined for the annotation task. In this model, concept instances are identified by URIs of document nodes and the literals associated by the *hasValue* attribute are the textual contents of annotated nodes.

We define the following aggregate metadata in C_S and R_S :

- The *PartOfSpeech* concept is used to annotate document nodes containing several instances of different concepts.
- The *SetOfConcepts* metadata is used to annotate document nodes containing several instances of the same concept. A concept *SetOfc_i* is defined as a subclass of *SetOfConcepts* for each concept $c_i \in \mathcal{O}$. Moreover, we define relations denoted $rSet$ and $rSet^{-1}$ in \mathcal{R}_S derived from (inverse) functional relations r in $\mathcal{R}_\mathcal{O}$ in order to represent relations between an instance and a set of instances.
- The *neighborOf* relation expresses a path in a XML/HTML document tree.

Instances of these metadata are generated by the *adapter* using a set of logical rules. If a document node contains only one instance of a domain concept c , it is annotated by c . The datatype properties of this instance become properties of the node. Else, it is annotated either by *SetOfc_i* metadata or *PartOfSpeech* metadata. The property *isIndexedBy* is instantiated for *PartOfSpeech* nodes. The provided annotations of domain relations $r \in \mathcal{R}_\mathcal{O}$ are instantiated between nodes whose types are in $\mathcal{C}_\mathcal{O}$ (domain concepts). In the case where r links a node of type $c_j \in \mathcal{C}_\mathcal{O}$ with a node of type *SetOfc_i*, r is substituted by $rSet$ or $rSet^{-1}$.

3 Query Reformulations

Preliminary Definitions: Consider the pairwise disjoint infinite sets I , B , L and V (IRIs, Blank nodes, Literals and Variables). A triple pattern is a triple $(s, p, o) \in (I \cup V) \times (I \cup V) \times (I \cup V \cup L)$. A **basic graph pattern** P is a set of triple patterns. $?v$ in a triple indicates that v is a variable. An RDF query is a basic graph pattern or a constructed graph pattern (using constructors such as union or intersection). To facilitate the reading of this paper, we consider only queries described by basic graph patterns. The filters that we consider use equality and inclusion operators between variables and literal values.

We define a **model-based query** q as a quadruplet (P, S, F, D) where :

- P is a basic graph pattern which complies with a model (i.e. \mathcal{O} or \mathcal{A}). $V(P)$ denotes the set of variables of P and $C(P)$ denotes the set of concepts of P .
- F is a constraint defined by a logical combination of boolean-valued expressions.
- S is the set of variables that appear in the *SELECT* clause of the query.
- D is an \mathcal{A} -compliant RDF dataset to be matched with P and F .

Example: The \mathcal{O} -based query q_0 is defined by (P_0, F_0, S_0, D) where :

$P_0 = \{ (?art, \text{rdf:type}, \text{Article}), (?aut, \text{rdf:type}, \text{Person}), (?aut, \text{hasName}, ?aName), (?conf, \text{rdf:type}, \text{Conference}), (?art, \text{publishedIn}, ?conf), (?art, \text{authoredBy}, ?aut), (?conf, \text{hasName}, ?cName) \}$
 $F_0 : \{ ?cName = "WW2008" \}$ and $S_0 : \{ ?art, ?aut, ?aName \}$

Neighborhood-based Reformulation: The aim of the neighborhood-based reformulation is to exploit the structural neighborhood of document nodes in order to find nodes that may be related by the semantic relations expressed in the user query. The neighborhood-based reformulation, denoted f_{nr} , substitutes the ontological relation of a given triple by a *neighborOf* relation.

Example: $f_{nr}(q_0, (?art, authoredBy, ?aut)) = q'_1(P'_1, F_0, S_0, D)$ where $P'_1 = \{(?art, rdf:type, Article), \dots(?art, neighborOf, ?aut), \dots\}$. Applying f_{nr} may generate semantically-independent subgraph patterns.

Definition 1. p is a *semantically-independent subgraph pattern* of P if :
 $-\forall v_1, v_2 \in V(p), (?v_1, r, ?v_2) \in P \rightarrow (?v_1, r, ?v_2) \in p$
 $-\forall (?v_1, r, ?v_2) \in P$ s.t. v_1 (resp. v_2) $\in V(p)$, v_2 (resp. v_1) $\notin V(p) \rightarrow r = neighborOf$
Splitting a query into semantically-independent subgraph patterns allows applying distinct aggregative reformulations on distinct sub-parts of a query.

PartOfSpeech Reformulation: The *PartOfSpeech* reformulation denoted f_{pr} assumes that the required semantic relations can be found between instances aggregated in the same node. It is applied on semantically-independent subgraph patterns. The target subgraph must be \mathcal{O} -based, i.e. it does not contain metadata from C_S or R_S (this constraint is part of the reformulations construction plan). f_{pr} substitutes all filter constraints of the subgraph pattern by filter constraints on the textual contents of *PartOfSpeech* nodes. Equality constraints are relaxed into inclusion constraints.

Example: for $p \in P'_1$ s.t. $p = \{(?art, rdf:type, Article), (?conf, rdf:type, Conference), (?conf, hasName, ?cName), (?art, publishedIn, ?conf)\}$,
 $f_{pr}(q'_1, p) = q'_2(P'_2, F'_2, S'_2, D)$ s.t.
 $P'_2 = \{(?pos, rdf:type, PartOfSpeech), (?pos, isIndexedBy, Conference), (?pos, isIndexedBy, Article), (?pos, hasValue, ?lPos), (?pos, neighborOf, ?aut), (?aut, rdf:type, Person), (?aut, hasName, ?aName)\}$
 $F'_2 : \{(?lPos \text{ contains } "WW2008")\}$, $S'_2 : \{?aut, ?aName, ?pos\}$

SetOfConcept Reformulation: The *SetOfConcept* reformulation, denoted f_{sr} , substitutes the ontological type c of a given variable v by the *setOfc* type if for all triples of P : (1) if v is the subject, the relation r is not inverse functional and (2) if v is the object, the relation is not functional. The relation r is then substituted by $rSet^{-1}$ (case 1) or $rSet$ (case 2).

Example: $f_{sr}(q_0, ?aut) = q'_3(P'_3, F_0, S_0, D)$ s.t.
 $P'_3 = \{(?art, rdf:type, Article), (?aut, rdf:type, SetOfPersons), (?conf, rdf:type, Conference), (?art, publishedIn, ?conf), (?art, authoredBySet, ?aut), (?aut, hasValue, ?aName), (?conf, hasName, ?cName)\}$

Reformulations Construction Plan: The reformulation of a query $q_0(P_0, F_0, S_0, D)$ is a query $q_i(P_i, F_i, S_i, D)$ obtained by the composition of elementary *PartOfSpeech*, *SetOfConcept* and *neighborhood-based* reformulations. We consider that a set of document nodes is more relevant if its nodes do not contain aggregated instances and if they are related by the expected semantic relations.

Definition 2. Let $N(q)$, $Pos(q)$ and $Sets(q)$ be resp. the number of *neighborOf*, *PartOfSpeech* and *SetOfc* metadata in a query q . The (well) order \preceq is defined s.t. $q_i \preceq q_j \leftrightarrow ((N(q_i) > N(q_j)) \vee ((N(q_i) = N(q_j)) \wedge ((Pos(q_i) > Pos(q_j)) \vee ((Pos(q_i) = Pos(q_j)) \wedge (Sets(q_i) \geq Sets(q_j))))))$

Dynamic Reformulation and Execution of Queries Algorithm (DREQ)

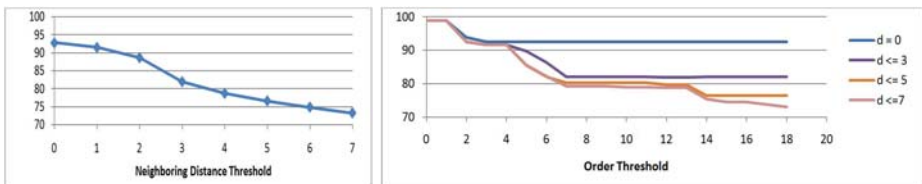
DREQ allows constructing and executing the reformulated queries with respect to \preceq . When *DREQ* is stopped at a given order, the answers are those retrieved by the best constructed queries. *DREQ* computes all reformulations in EXPTIME w.r.t the number of variables of the user query, but, as the algorithm is dynamic, we obtain a new set of equally-ordered reformulations in PTIME.

4 Experimental Results

SHIRI-Querying has been implemented and experimented to study how the precision and the recall measures vary according to the order relation. The *neighborOf* relation is defined as an undirected path of length d in the HTML /XML tree. We also study how d influences the results. The reformulations proposed in our approach can introduce wrong answers which may appear when the query is reformulated using f_{sr} and f_{pr} which relax filters in *PartOfSpeech* or *SetOfConcepts* nodes or using f_{pr} and f_{nr} which relax semantic relations. The two experimented datasets belong to the scientific conferences and publications domain.

The first dataset is composed of annotated publication references provided by the DBLP XML repository, the INRIA HTML server and the HAL XML repository. It consists of more than 10.000 RDF triples describing 1000 publications. We submitted a set of queries looking for conferences, their dates, locations, papers and authors. A precision of 100% and a recall of 100% were reached with an order threshold of 9 and $d \leq 3$. A smaller order threshold leads to a smaller recall and a higher distance d leads to almost 0% precision. In this case ($d > 3$), in two data sources, each paper is associated to all conferences. The 100% values for the recall and the precision measures are due to the regular structure of the data sources. However, each data source has a different and specific structuring and the *DREQ* reformulations were able to integrate answers from all sources.

The second corpus consists of RDF annotations of 32 call-for-papers web sites and is consequently very heterogeneous. These annotations (consisting of 30.000 RDF triples) were generated automatically using *SHIRI – Extract* [5]. We then submitted a set of 15 queries. Without reformulation, all queries have no answers (0% recall), while we obtained a 56% recall by using the *DREQ* algorithm for $d \leq 7$. At the same distance threshold ($d \leq 7$) the precision is still 72%. The results

(a) Precision according to d

(b) Precision according to order

Fig. 1. Answers' Precision

show that domain relations can often be retrieved between instances located in close document nodes. Figure 1(b) presents the average precision value for the same set of user queries, for several values of d , by varying the order threshold from 1 to 18. The precision variations show that the order relation is relevant to rank the answers.

5 Conclusion and Future Work

In this paper, we presented the *SHIRI-Querying* approach to support semantic search on heterogeneous semi-structured documents. Ontology-based user queries are reformulated to gather document nodes from documents that were annotated in an imprecise and incomplete manner by semantic annotation tools. These reformulations allow retrieving instances that are related by the requested semantic relations even if these relations are not available in the knowledge base. We defined an order relation between reformulated queries to give priority to queries that preserve most the semantics of the user query. All reformulations are constructed dynamically w.r.t this order relation in the *DREQ* algorithm. Experimental results show that the recall greatly increases and that the precision decreases reasonably as the ordered reformulated queries are performed. In the near future we plan to combine keyword-based search with our reformulation approach to increase the recall without losing the semantics of the query. We also plan to use semantic-based heuristics exploiting functional properties of relations in order to avoid some wrong answer cases.

References

1. Borislav, P., Atanas, K., Angel, K., Dimitar, M., Damyan, O., Miroslav, G.: KIM - Semantic Annotation Platform. *J. of Nat. Lang. Engineering* 10(3-4), 375–392 (2004)
2. Etzioni, O., Cafarella, M., Downey, D., Kok, S., Popescu, A., Shaked, T., Soderland, S., Weld, D., Yates, A.: Unsupervised named-entity extraction from the web: An experimental study. *Artificial Intelligence* 165(1), 91–134 (2005)
3. Bhagdev, R., Chapman, S., Ciravegna, F., Lanfranchi, V., Petrelli, D.: Hybrid Search: Effectively Combining Keywords and Semantic Searches. In: Bechhofer, S., Hauswirth, M., Hoffmann, J., Koubarakis, M. (eds.) *ESWC 2008*. LNCS, vol. 5021, pp. 554–568. Springer, Heidelberg (2008)
4. Corby, O., Dieng-Kuntz, R., Gandon, F., Faron-Zucker, C.: Searching the semantic web: Approximate query processing based on ontologies. *IEEE Intelligent Systems Journal, Computer Society* 21(1), 20–27 (2006)
5. Thiam, M., Bennacer, N., Pernelle, N., Lo, M.: Incremental Ontology-Based Extraction and Alignment in Semi-Structured Documents. In: Bhowmick, S.S., Küng, J., Wagner, R. (eds.) *DEXA 2009*. LNCS, vol. 5690, pp. 611–618. Springer, Heidelberg (2009)
6. Hurtado, C.-A., Poulouvasilis, A., Wood, P.-T.: A Relaxed Approach to RDF Querying. 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. 314–328. Springer, Heidelberg (2006)
7. Castells, P., Fernández, M., Vallet, D.: An adaptation of the vector-space model for ontology-based information retrieval. *IEEE T. on Know. and Data Eng.* 19(2) (2007)

Query Ranking in Information Integration

Rodolfo Stecher, Stefania Costache, Claudia Niederée, and Wolfgang Nejdl

L3S Research Center, Appelstr. 9a, 30167 Hannover, Germany
{stecher,costache,niederee,nejdl}@L3S.de

Abstract. Given the growing number of structured and evolving on-line repositories, the need for lightweight information integration has increased in the past years. We have developed an integration approach which relies on partial mappings for query rewriting and combines them with a controlled way of relaxation. In this paper we propose a novel approach for ranking results of such rewritten and relaxed queries over different sources, by punishing lack of confidence in mappings used for rewriting, as well as punishing higher degrees of controlled relaxation introduced, and present the performed evaluation.

Keywords: Query Ranking, Information Systems, Query Relaxation.

1 Introduction

A growing number of structured large information collections is made accessible over the Internet, e.g. Freebase, Linked Data, also in the area of Personal Information Management, new repositories as Flickr, YouTube are increasingly used. Efforts in adding structure and giving semantics to the available information, result into further information collections (e.g. UMBEL, DBPedia). The easy exploitation of such sources, however, requires approaches for flexible, lightweight integration of distributed and evolving structured information sources. Thus, lightweight approaches are required for query answering over evolving information systems with reasonable result quality, even when only partial mapping information is available (incomplete mappings with various confidences). We follow a lightweight approach similar to a pay-as-you-go (results are as good as possible given the available evidences) integration, which uses partial mappings for rewriting and relaxing structured (triple-based) queries and learning of new mappings from the results, as described in more detail in [1].

Even with such a solution at hand, the multitude of data within information systems makes finding the needed results still difficult, since relaxed queries do not always provide an exact set of results, but rather an extended, more permissive set of results. When information comes from various sources with different degrees of confidence, an ordering among the results reflecting this confidence is needed. In this paper we propose a ranking algorithm which punishes lack of confidence in the used partial mappings and query rewriting strategies.

The main contributions of our approach can be summarized as follows: 1) An innovative query ranking approach for our lightweight information integration

approach [1]; 2) An evaluation of the proposed ranking approach using large real world data sets showing its applicability in realistic settings.

Next, Section 2 presents our query ranking approach, its evaluation is presented in Section 3, Section 4 presents the related work, and Section 5 summarizes our conclusions and future work.

2 Ranking for Lightweight Information Integration

After applying our relaxation strategies [1], we use the ranking of *queries* to rank results from different information sources. Considering that complete and confident mappings can only give correct results (the ideally rewritten query), our approach for a *query* ranking function is based on introducing punishments to each of the aspects which might introduce errors. Intuitively, the higher the difference of the resulting query from the ideally rewritten query, the more “punishment” we add to the results obtained from executing it. Due to space constraints we refer to the definitions presented in [1].

2.1 Punishment Derived from Mappings

The higher the confidence in the correctness of a mapping is, the lower the probability of introducing an error when this mapping is used to rewrite the user query. For reflecting this, and also considering that mappings are independent, we define a factor called **Query Mapping Confidence** (Qmc), reflecting the probability of introducing errors by the product of the confidences of all mappings actually used in the rewriting of Q^u , multiplied once (as also done in [2]):

$$Qmc = \prod v|(e, e', v) \in M'_i \quad (1)$$

The rationale for this is that each successfully applied mapping will avoid that this element is relaxed with a wildcard, but it will also possibly add some incorrect results, depending on the confidence of this mapping.

2.2 Punishment Derived from Relaxation

Our query relaxation might introduce errors by allowing bound expressions in the user query Q^u to become unbound in the reformulated query Q^{S_i} . The computation of the punishment considers the number of introduced variables which remain bound and the number of variables introduced which do not even have a value constraint. The computation relies on the following additional notations:

- qL : the number of triple elements in Q^u : $qL = \text{Number of triples} * 3$
- \mathcal{LU} : the element names in Q^u not part of $A^u \cup \tau \cup VAR_{Q^u}$ (i.e. Literals and URI's not belonging to the schema)
- nb and nu : the number of bound and unbound variables introduced in the relaxation process (defined below)

Next we define the introduced number of bound and unbound variables:

- *nb*: For $\langle s, p, o \rangle$, p is replaced with $var_p: \langle s, var_p, o \rangle \wedge o \in \mathcal{LU}$, then $nb = nb + 1$ once for var_p . This is, nb will be incremented by one the first time var_p is introduced, even though any other occurrence of p will also be replaced with the same var_p . The idea is that we are introducing a wildcard, but the values it can take are restricted by s and the given value of o which is fixed.
- *nu*: we have two options: 1) For $\langle s, \tau, o \rangle$, o is replaced with $var_c: \langle s, \tau, var_c \rangle \wedge o \in A_C$, $nu = nu + 1$ for every usage of var_c . We increase nu for every occurrence, since more relaxation is added with every replacement. In this case we are allowing s to be of any type, as well as any other thing originally specified to have the same type as s , so we are relaxing the “essence” specification of things; and 2) For $\langle s, p, var_o \rangle$, p is replaced with $var_p: \langle s, var_p, var_o \rangle \wedge var_o \in VAR_{Q^u}$, $nu = nu + 1$ for every occurrence of var_p . We increase nu since each occurrence increases the degree of relaxation because we had already only the relation p as a restriction between the values that s and var_o could get, and now we are even relaxing this last restriction. In this case the relaxed query expresses the fact that there must be some connection between s and var_o , but without saying which connection.

With these measures, we can compute a punishment for bound variables as:

$$P_{nb} = 1 - \frac{nb}{qL} \quad (2)$$

We need this factor, since even though the variables are bound, their number influences the relaxation relative to the query joins (therefore the normalization to the query length). Also, the correlation between nb and the ranking is indirect, since the less bound variables are introduced, the higher we can rank that query. The introduction of unbound variables has a higher influence on the accurateness of the query results than the bounded ones, because they introduce more relaxation to the query, and therefore more penalty is needed. We define the punishment for unbound variables as:

$$P_{nu} = \alpha^{1 - \sqrt{\frac{nu}{qL}}} - 1, \quad (3)$$

where α , the relaxation penalty, has to be defined. It grows differently than the one computed for bound variables: errors introduced by unbound variables make the query less accurate than introduction of bound variables, and therefore this factor should have a more dramatic decrease when the number of unbound variables is high. We experimented with different functions for the exponent factor of P_{nu} , but they all behaved similarly.

2.3 Ranking Function

Based on the factors presented above, we define a ranking function, giving higher ranking to results of queries with likely less errors. The factors presented in Equations 1, 2 and 3 are probabilistically independent and therefore the ranking function can be computed as a product of the possible punishments (errors) introduced:

$$R(Q^u) = Qmc * P_{nb} * P_{nu} \quad (4)$$

$R(Q^u)$ estimates the expected correctness of the modified query results, by considering the confidence of the mappings, and the effects of applying a “wildcard” strategy on the original query. The combination of these values leads to a measure of how much the modified query “deviates” from the ideally rewritten query.

3 Evaluation

By computing the precision for different top-k results, we prove the ranking method to be efficient in ordering the results. Since for the ideal query, all results are equally correct, we could not compare our results against a ranked ground truth, instead, we used the complete ground truth. As strategies SUB^2 and SUB^4 presented similar behavior as SUB^6 they will not be discussed in detail.

3.1 Evaluation Setting

Information Sources. The heterogeneous information sources considered with the number of contained triples are presented in Table 1. In detail they are:

Virtual Personal Desktop (VPD) obtained from crawling 16 desktops (PDFs, Word documents, Emails, Wiki pages, FOAF profiles) using the crawling approach and ontologies presented in [1], each with their own user ontology and query set (see [1] for details).

UMBEL (<http://www.umbel.org/>) provides an ontology and its instances, along with many definitions of equality (using “sameAs” relations) to instances in the other datasets used for this experiment - YAGO and DBPedia. Therefore, the instances provided by UMBEL were used to compute the ground truth. The *UMBEL Ontology* was taken as the *user ontology* (in a filtered version), and will be denoted from now on as user ontology O^u . In order to construct the *UMBELInstances* data source and its ontology, original references to concepts and properties, as well as to resources in the instances were modified programmatically, simulating in this way a new source, from now on UMBEL.

YAGO [3] - from the provided instances a simple ontology (*YAGO Ontology*) describing them was extracted.

DBPedia (<http://wiki.dbpedia.org/>) - two sources were created: the *DBPediaPersons* containing all available “Persondata” files which are represented using the *FOAF* ontology, and the *DBPediaInfoboxes* containing the “Infoboxes” and “Types” files which are represented using the *DBPedia Ontology*.

Table 1. Information Sources Overview

| Source | VPD | UMBEL | YAGO | DBPediaPersons | DBPediaInfoboxes |
|-------------|-------------|-------------|---------------|----------------|------------------|
| No. triples | 3, 329, 376 | 6, 740, 428 | 460, 418, 591 | 812, 339 | 9, 877, 564 |

Initial Mappings. Were computed between the *UMBEL Ontology* and the ontologies of the sources using [4], and later randomly modified to serve as initial *partial* mappings. Partial mappings were also computed for the VPDs, between the user ontology specified in [1] and the ontologies describing the different crawled sources.

Queries. Our query set is an extension of the queries used in [1] and consists of more than 70 queries containing mainly 1 to 3 joins.

Ground Truth. For each query there is a ground truth, which contains all correct results expected. In order to have comparable query results from the different sources, explicit equivalences between resources have been exploited.

3.2 Experiments

For each information source, we employ relaxation strategies (wildcard based) together with rewriting strategies (mapping based), and the techniques for simulating user feedback and learning of mappings as described in [1]. We compared the top-k *integrated* results of the same query over all available information sources with the ground truth and measured the precision at top-k (top-1 to top-5) using a precomputed value of $\alpha = 2$ (the experiments for determining the α value are not presented due to space constraints). The mean results obtained from running this evaluation over all presented datasets, iterations and strategies is presented in Figure 1(a), and by strategy in Figure 1(b). We computed precision by considering all queries, also the ones having less than k results. There is not much difference between the obtained precision results at top-1 to top-5, all of them being around 0.9, which we consider to be a good precision for our lightweight integration approach. Most of the strategies have high precision, being notable that strategies SUB^1 and SUB^5 give the best results in our settings. Strategy SUB^3 shows the worst precision results at top-1, which is an indicator that the usage of this strategy for our presented settings needs to be revised.

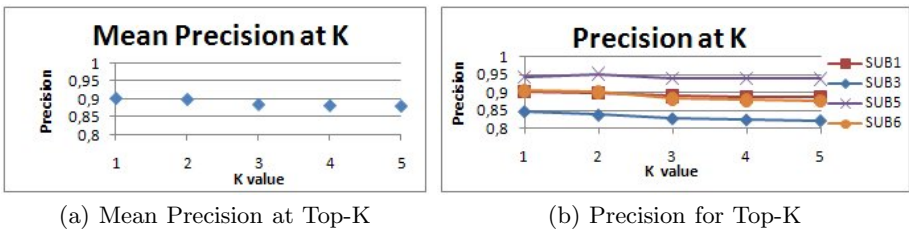


Fig. 1. Overall Precision

4 Related Work

Ranking is mostly known from Information Retrieval approaches which rank the results from structured queries, nevertheless, none to our knowledge combine rewriting queries using wildcards and partial mappings, with ranking. Ranking is computed on relaxed or malleable queries in [5], by considering the quantification

of the correlations existing between attributes (of duplicate entities detected in the data), and ranking higher the results based on relaxations using higher correlated attributes. This is similar to our idea to use the confidence of the mapping in computing the ranking function, but we don't require data access to detect correlations. In [6], a technique for ranking query results on the semantic web takes into consideration the inferencing processes that led to each result, where the relevance of the returned results is computed based upon the specificity of the links used when extracting information from the knowledge base. This approach is complementary to our approach, since the confidence values from the inferencing process could be an additional confidence to the computation of our ranking. [2] aims also at integrating distributed sources containing RDF data described by ontologies. An important difference to this approach is that only rewriting without relaxation of queries is produced, so, the ranking for the results of queries only considers the confidence of the used mappings. Due to space constraints many other related approaches had to be left out of this section.

5 Conclusions and Future Work

We presented an approach for ranking results of rewritten and relaxed queries executed over different repositories. We use the confidence of employed mappings in combination with heuristics which consider the amount and position of wildcards introduced. These factors are combined in a weighted fashion, to produce a ranking value for the results of executing the modified query on a specific information source. Our evaluations performed over real world datasets show the efficiency of our introduced ranking function.

The usage of the query ranking value for deciding on executing a query (or not) on a given information source is an interesting idea to be explored. It could serve for finding a trade-of between result precision and recall of unknown but relevant information. A future idea would be to take into account in the ranking computation a factor reflecting the confidence of the used data sources.

References

1. Stecher, R., Niederée, C., Nejd, W.: Wildcards for lightweight information integration in virtual desktops. In: CIKM (2008)
2. Straccia, U., Troncy, R.: Towards distributed information retrieval in the semantic web: Query reformulation using the oMAP framework. In: Sure, Y., Domingue, J. (eds.) ESWC 2006. LNCS, vol. 4011, pp. 378–392. Springer, Heidelberg (2006)
3. Suchanek, F.M., Kasneci, G., Weikum, G.: Yago: A Core of Semantic Knowledge. In: International World Wide Web Conference, New York, NY, USA (2007)
4. van Elst, L., Kiesel, M.: Generating and Integrating Evidence for Ontology Mappings. In: Motta, E., Shadbolt, N.R., Stutt, A., Gibbins, N. (eds.) EKAW 2004. LNCS (LNAI), vol. 3257, pp. 15–29. Springer, Heidelberg (2004)
5. Zhou, X., Gaugaz, J., Balke, W.T., Nejd, W.: Query relaxation using malleable schemas. In: SIGMOD: International Conference on Management of Data (2007)
6. Stojanovic, N., Studer, R., Stojanovic, L.: An approach for the ranking of query results in the semantic web. In: Fensel, D., Sycara, K., Mylopoulos, J. (eds.) ISWC 2003. LNCS, vol. 2870, pp. 500–516. Springer, Heidelberg (2003)

Validity of the Documentation Availability Model: Experimental Definition of Quality Interpretation

Raimundas Matulevičius^{1,2}, Naji Habra¹, and Flora Kamsseu¹

¹ PReCISE, Computer Science Faculty, University of Namur
rue Grandgagnage 21, 5000 Namur, Belgium
{nha, fka}@info.fundp.ac.be

² Institute of Computer Science, University of Tartu,
J. Liivi 2, Tartu, Estonia
rma@ut.ee

Abstract. System and software documentation is a necessity when making selection and acquisition decisions, when developing new and/or improving existing system and software functionality. A proper documentation becomes even more crucial for open source systems (OSS), where, typically, stakeholders from different communities are involved. However there exist only limited or no methodology to assess documentation quality. In this paper we present a quality model and a comprehensive method to assess quality of the OSS documentation availability (*DA*). Our contribution is threefold. Firstly, based on the criteria defined by Kitchenham *et al.* we illustrate the theoretical validity of the *DA* model. Secondly, we execute the first step towards the empirical validity of the *DA* model. Finally, our work results in a comprehensive and empirically grounded interpretation model of the documentation quality.

Keywords: Documentation availability, theoretical and empirical validity, quality metrics and indicators, open source software documentation.

1 Introduction

Nowadays, information systems (ISs) are based on open source software (OSS) products either partially or fully. OSS applications might vary from the middleware and platform levels (e.g., operating systems, software development environment) used to support ISs' development to the direct management and dissemination of information (e.g., calendars, text editing and Web applications). Working with a system of interrelated products implies the necessity to have and maintain a high-quality documentation. This is true for any system and software products, but especially this becomes critical for the OSS products, where documentation is also used for communication [6] between distributed stakeholders.

There exist a number of quality models (e.g., ISO-9126 [13], McCall [19], Boehm *et al* [1]) addressing software *product* criteria like maintainability, usability, and reliability. Other models (e.g., CMM [20], SPICE [5]) consider software *process* criteria like maturity or capability with the underlying hypothesis that good process would result in a good product. Although it is recognised that a *high-quality documentation*

is necessary to different stakeholders (users, developers and acquirers) [15], little is done to consider the quality criteria of the documentation itself.

A proper assessment of the software documentation is necessary for several reasons. Firstly, it is a part of the overall product assessment at the decision making process during product acquisition. Secondly, assessing documentation can help a choice between several alternatives. Thirdly, assessment results might help developers to improve the documentation quality *per se*. Documentation quality assessment is also important in agile approaches, which apparently contradict the absolute necessity of extensive documentation and focus on delivering the workable software products as the main measure of software process progress. Even in such cases, software product and process quality depends on a minimal documentation while the inherent rule is to avoid unnecessary documentation. Therefore, a quality approach necessitates some means (i.e., some precise criteria) to evaluate whether the documentation is really adequate or too excessive with respect to the context of use.

To achieve a documentation assessment, one needs a comprehensive quality model comprising precise guidelines and criteria which are (i) *consensually admitted* by the different stakeholders as reflecting their perception of documentation quality, (ii) *easy to apply*, and which produce (iii) *quality indicators in a repeatable and reproducible* manners. To our knowledge, there is no an established model today. Thus, our long-term research goal is to establish such a model.

In a previous work [18] we have elaborated a method to assess documentation availability (*DA*). Our method with its underlying model supports investigation of the documentation context and judges about the completeness of the information. In this paper, we make a step towards the theoretical and empirical validation of the *DA* method. As the part of the empirical validation we report on the performance test, where we have applied our *DA* method to assess documentation quality of 28 OSS projects. The findings help us to define the quality interpretation model. In this paper we report a number of observations resulting from our assessment.

The structure of the paper is as follows: in Section 2 we present the documentation availability model and estimation method. In Section, 3 we discuss theoretical and empirical validity of this model. In Section 4, we continue our discussion of the empirical validity by describing the performance test, which investigates the feasibility of our proposal. Finally, Section 5 discusses the findings, presents the conclusions and future work.

2 Documentation Availability

In this section we overview the *DA* model. We present metrics, indicators, estimation method, and quality interpretation model.

2.1 Documentation Availability Model

The quality model for the OSS documentation availability is presented in [18]. It is a part of the bigger effort to develop the QualOSS¹ assessment method [3, 23]. In

¹ QualOSS stands for Quality of Open Source Software. Project is funded by European Commission under the FP6-2005-IST_5 Framework, Contract number N° 33547.

QualOSS, the OSS product is described through four dimensions: (i) *the software product*, defined by software code, documentation and test; (ii) *the community*, characterised by interconnected community members; (iii) *the development process*, expressed by rules that community members should follow when performing activities, and (iv) *the tools* used by the community to build, manage and maintain the OSS. Hence, the documentation quality is a part of the aggregated *product* quality.

For documentation quality a distinction could naturally be done between two levels (Fig. 1). Firstly one needs to identify characteristics related to the content of documentation and determining its *accuracy* with respect to what documentation is supposed to describe [21], what is the purpose of documentation, and what stakeholders' behaviour the documentation might lead. Secondly, one needs to investigate characteristics related to the form and determining *completeness and availability* of different structural parts [4]. Although theoretically both aspects are important, assessing the first one appears to be difficult as long as we admit that the documentation could be written manually and/or in a non-formal style. In this work we focus on the second aspect, thus, we develop a systematic method [18] to assess *documentation availability* (DA) using two indicators (Fig. 1): *documentation type availability* and *documentation information availability*.

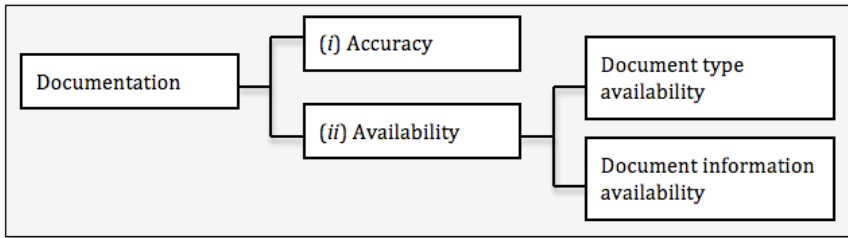


Fig. 1. The Documentation Assessment Model

2.2 Documentation Type Availability

In [18] we define thirteen document types and classified them to four categories: presentation documents, product installation and application documents, documents of product development process, and management and copyright documents. *Document type availability* (DTA) characterises availability of documents belonging to a certain document type:

$$DTA = \frac{DF}{DN} \quad (1)$$

here, DN is the number of considered document types (i.e., chosen from the set of 13 identified types), DF is a number of documents types for which documents are found.

This indicator tells that there should exist a set of *documents* allowing the *stakeholders* to achieve their goals. We indicate that a single stakeholder can play four different roles: *product acquirer* (interested in presentation documents), *product user* (concerned by the product installation and application document), *product developer* (interested in documents of product development) and *product contractors* (concerned

by management and copyright document). Depending on the stakeholder's goal, different documents would be evaluated. For example, to achieve the goal of product acquirer, one needs to consider presentation documents; to reach the goal of product user, one needs to evaluate availability of product installation and application documents. In this paper we will focus on the overall documentation assessment taking all four stakeholders' goals together.

It is very important to know the *location* of these documents. It might be situations when a document actually exists, but the stakeholder is not able to find where this document is stored. This means that the document is still not available for the stakeholder.

2.3 Documentation Information Availability

Document information availability (DIA) indicates whether the documents contain *organised* and *complete information*, presented at the *complete level of detail*:

$$DIA = \frac{\sum_{i=1}^{DN} (dor_i + dco_i)}{2DN} \quad (2)$$

here dor_i – is an aggregated metric representing the organisation of documents belonging to document type i ; dco_i – is an aggregated metric representing the completeness of documents belonging to document type i ; DN – is the number of considered document types.

Document organisation can be estimated as easiness to locate information and logical relationships among adjacent chapters, sections and subsections [18]. Hence we defined a number of simple “Yes/No” questions (e.g., *Is there a table of content in the document?*; *Is the document divided to chapters?*, etc) that help estimate document organisation. The value is calculated as a ratio between the number of positively answered questions and the number of considered questions (recall that some questions could be non applicable and should be disregarded):

$$dor = \frac{\sum_{i=1}^N r_i}{N} \quad (3)$$

here $\{r_1 \dots r_N\}$ are answers (“Yes” $\equiv 1$, “No” $\equiv 0$) to questions about document organisation, N – number of questions having answers “Yes” or “No”, and dor – organisation of a single document.

A document is *complete* with respect to its content (*content completeness*) if it contains all the information necessary for the concerned stakeholder to satisfy his/her goal(s). Following software development standards [7, 8, 9, 10, 11, 12] we have developed a number of *content templates* describing the content structure of different document types [18]. When analysing a single document, an evaluator checks if this document contains the *information units* (sections, subsections, and paragraphs) as suggested in these templates.

Document is said to be at the *complete level of detail* (*information completeness*) if it defines *all* information units at the *high level of detail*. Information units, identified

when analysing content completeness, here, are considered for their information completeness. The *level of detail* is considered on an ordinal scale of four values (“null” \equiv 0, “low” \equiv 1, “average” \equiv 2, and “high” \equiv 3).

Document completeness is an aggregated measure of document content completeness and document information completeness [18]. It is expressed as the sum of multiplications between content completeness and information completeness, divided by the tripled number of analysed information units:

$$dco = \frac{\sum_{i=1}^M c_i d_i}{3M} \quad (4)$$

here $\{c_1 \dots c_M\}$ are the values $\{0, 1\}$ assigned to the answers to questions about content completeness, $\{d_1 \dots d_M\}$ are estimations of the information completeness according to the ordinal scale $\{0 < 1 < 2 < 3\}$, M – number of information units (questions), and dco – completeness of documents.

2.4 Documentation Availability Estimation Method

The application of the *DA* model consists of six steps (Fig. 2). Firstly, evaluator *defines the purpose and the scope* of the evaluation. This first step produces the scoped document types according to which a *search for documents* is performed in the second step. When screening for documents (step 2), one records the location of the found documents in order to access them after. The second step results in the (references to) documents which availability is under assessment. In the third step the evaluator *assigns documents* to the document types. The fourth step is the *review* of the document-document type pairs. This step should ensure that concerned documents are found and these are correctly assigned to the document types. In the fifth step, *each pair* is analysed for both the document organisation and document completeness. In the sixth step the evaluator computes the *DTA* and *DIA* indicators. As discussed above, *DTA* is estimated (equation 1) according to the number of found documents (resulting from step 4) and the number of considered document types (decided in step 1); *DIA* is computed (equation 2) from the document organisation (equation 3) and document completeness (equation 4) estimated in step 5.

2.5 Indicator Interpretation

Both *DTA* and *DIA* indicator values are estimated on the percentage interval (0 to 100% of availability), higher value means the documentation availability is higher. Following the guidelines of the QualOSS project [3, 23] we defined the indicator interpretation model (Table 1) both at the interval and at the ordinal scales (by mapping the former one to the latter one) [18]. The reason to define interpretation model at two scales is the simplicity to classify things into few categories. For the targeted audience it is much simpler to understand a given category (e.g. Yellow), than to judge about some number, received after calculation is done.

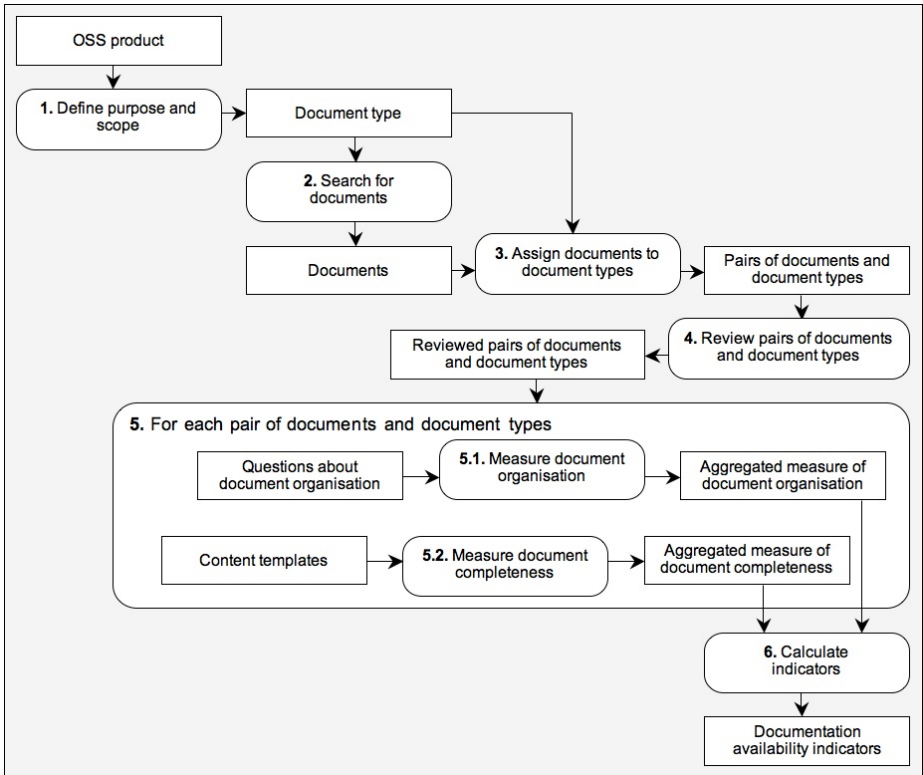


Fig. 2. Documentation Availability Estimation Method

Table 1. Interpretation of DA Indicators

| Interval scale (%) | Ordinal scale (colour) | Explanation |
|--------------------|------------------------|---------------|
| [0 ... 9,99] | <i>Black</i> | Not available |
| [10 ... 39,99] | <i>Red</i> | DA is limited |
| [40 ... 69,99] | <i>Yellow</i> | DA is average |
| [70 ... 100] | <i>Green</i> | DA is high |

However this interpretation model has several limitations. Firstly, its definition (e.g., thresholds and their suggested explanation) is highly subjective, provided without extensive testing of indicators on the OSS assessments. Secondly, the two indicators cannot be interpreted at the same scale, because their natures, measurement, and estimation inputs are different as defined in the above sections. Finally, although being applied to assess few OSS projects [18], the DA method itself was not yet applied to a sufficiently large extend. Thus, its validity needs to be investigated at a more coarse-grained level.

The main goal of this paper is twofold. Firstly, we need to validate the DA model [18]. Secondly, we need to establish an empirically grounded *interpretation model* for the DA indicators.

3 Validity of the Documentation Availability Model

In order for a quality model to be valid, all its metrics (including aggregated metrics and indicators) have to be valid. For a metric to be valid, the following two conditions must hold [16]: (i) this metric must not violate any necessary properties of its elements, and (ii) each model used in the measurement process must be valid. In other words to validate the *DA* method we need to show validity of

- the metrics (e.g., *chapter*, *section* used to gather data about document completeness; *information units* used to gather data on document organisation), aggregated metrics (e.g., *document organisation* (see equation 3) and *document completeness* (see equation 4)), and indicators (e.g., *DTA* and *DIA*);
- the measurement scales; and
- the estimation method, presented in Section 2.4.

Kitchenham *et al.* [16] define two major methods to check metrics validity (Fig. 3):

- *Theoretical validation*, which confirms that the measurement does not violate any required properties of measurement elements or of the definition models [16].
- *Empirical validation*, which corroborates that measured attributes are consistent with the values predicted by the models involving the attribute [16].

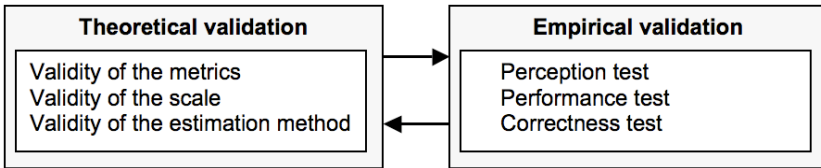


Fig. 3. Theoretical and Empirical Validation

3.1 Theoretical Validation of the Documentation Availability Model

Theoretical methods of validation allow us to say that a metric is valid to a certain defined criteria. Such a list of criteria is presented in [16]. We present them hereafter together with theoretical validity of our *DA* model.

Validity of the metrics. Four criteria are defined for the metric validity:

- *For the metric to be measurable it must allow different entities to be differentiated from each other* [16]. By definition [18] (also see Section 2.2) the *DA* metrics allow distinguishing different documents according to their document types;
- *A valid metric must obey the representation condition* [16]. For instance, the *DA* metrics preserve the intuitive notions about the document type’s property and the way in which document type properties distinguish between document types.
- *All scales contributing to a valid metric are equivalent* [16]. It is possible to define different valid scales for the *DA* metrics. For instance, in Section 2 we define information units that are measured in the scale of [“null” ≡ 0, “low” ≡ 1, “average” ≡ 2, and “high” ≡ 3]. However this scale could be extended with intermediate values

(e.g., [“null” \equiv 0, “low” \equiv 1, “somehow low” \equiv 1.5, “average” \equiv 2, “somehow high” \equiv 2.5, and “high” \equiv 3]). Hence both scales contribute to a valid measure.

- *Different entities can have the same metric value* [16]. Applying the DA model, we estimate document organisation using the same set of metrics. These metrics can receive the same value for different documents of different document types. Thus, we say that the *document organisation* metrics satisfy all this validity criteria.

Validity of the aggregated metrics and indicators. The aggregated metrics (e.g., *document organisation* and *document completeness*) and indicators (e.g., *DTA* and *DIA*) are valid theoretically because:

- They are based on a model (e.g., *content templates* when calculating *documentation completeness*) concerning the relationships among document properties.
- They are based on a dimensionally consistent model (as defined in [16]), estimating the aggregated metrics and indicators on the percentage scale.
- The discontinuities are not possible because metric values are received by normalising (always larger or equal) numbers of the considered entities (e.g., for the *DTA* the number of documents found is always either equal or lower to the number of considered document types).
- They include the correct measurement scales, characterised as interval.

Validity of the documentation availability estimation method (or measurement protocol as called in [16]) is usually validated by peer acceptance. As mentioned in Section 2, the DA model and the estimation method are the parts of larger effort to construct a standard QualOSS assessment method. The DA estimation method was reviewed by the project partners several time [23]. It has also been tested in few minor OSS assessments [18]. Our experience shows that the DA estimation method is unambiguous, self-confident, and prevents the problems such as double counting.

Our theoretical validation corresponds to the application of [16] reported in [17]. But we also admit that the full consensus could be reach only through the iteration of discussions with the community (in this case, partners of the QualOSS project).

3.2 Empirical Validation of the Documentation Availability Model

To corroborate a metric empirically, one needs to perform experiments to show whether people agree that a measurable property exists and whether a mapping to value captures the understanding of the property [16]. In general, empirical validation could be performed through perception, performance and correctness tests.

The *perception test* involves investigation of the artefact usability, ease of use, and user satisfaction. Currently, this approach is not applicable for our DA model, because the model is a novel proposal and there is no sufficient experience to report on its perception on a large extend. The *performance test* describes the application of the artefact to see its feasibility. In this work we apply the performance test and we assess documentation availability of 28 of OSS projects. We report our experience in Section 4. Finally, currently we are designing the *correctness test*. We plan to contact the OSS community members in order to receive their feedback on the results received during the performance test. We hope to show that the performance of the DA model is indeed correct and corresponds to the perception of the experts.

4 Performance Test

In this section we present the first step towards the empirical validation of the *DA* model: a performance test to investigate our model feasibility. In Section 4.1 we describe its performance test design. Section 4.2 discusses validity threats. Section 4.3 presents the major results. Finally, in Section 4.4, we will refine the *DA* interpretation model on basis of the performance test results.

4.1 Design

As discussed in Section 2, one of our research goals is also to define an empirically grounded interpretation model for the documentation quality. We formulate the following **research question**:

How do the current indicators help to understand and interpret quality of the open source documentation availability?

Firstly, this research question challenges the application of the *DA* method to assess the *DA* of the OSS products. This shows the *DA* method validation through the performance test, as discussed in Section 3.2. Secondly, we will investigate whether the indicator values received for the sample of OSS projects cover the whole range of the ordinal scale defined for these indicators. If it is not the case, based on the assessment results we will refine the initial interpretation model (Table 1).

The **research method** is pretty straightforward. First, we formulated the research question presented above. Then, we defined a sample of the OSS projects. Next, we applied our *DA* model and calculated the values of *DTA* and *DIA* for each selected project. Finally, we analysed and interpreted the received results.

The **assessment sample** includes 28 OSS projects (see Table 2). We selected them based on the OSS survey provided in [14]. The selected OSS projects cover different Information System application domains, such as content management systems (e.g., *Plone*, *JetSpeed*, and *Jakarta Struts*), email systems, calendar systems, and address-book systems (e.g., *Thunderbird*, *Evolution*, and *Sup*), text viewing/editing systems (e.g., *Open Office writer*, *xEmacs*, *Evince*, and *xPDF*), Web application support systems (e.g., *Zope*, *Httpd*, *jMeter*, and *Galeon*), operating and their management systems (e.g., *FreeBSD*, *NetBSD*, and *Nautilus*), programming languages and their environments (e.g., *Python*, *Eclipse* platform, *gcc backend*, and *Findbugs*).

4.2 Threats to Validity

Before presenting the performance test results, we discuss some validity threats [24]:

- Reliability of the *QA* model could be seen as the *internal validity threat*. However, as we illustrated in Section 3.1 our proposal is theoretically valid. In addition it was developed systematically based on existing system and software development standards [7, 8, 9, 10, 11, 12]. Based on the assumption that the used system and software development standards (that provide also guidelines for the documentation preparation) also apply to the documentation of the OSS products, the *QA* model is systematically developed as illustrated in [18].

- Reliability of the assessment data on the 28 OSS projects could be found as the *conclusion validity* threat. Collecting of these data (step 5, Fig. 2) is a manual task, thus, it contains a certain degree of subjectivity. To mitigate this threat the assessment was always executed by at least two assessors. The first assessor was taking the actual metrics on the OSS documents. The purpose of the second assessor was to monitor and to review the assessment results. For some projects the assessment step (step 5, Fig. 2) was iterated few time in order to ensure the assessment quality and to decrease the result subjectivity.
- The majority of the OSS assessments was done by the same assessor (who was not the author of the *DA* model). Thus, there might be the *internal validity threat of maturation* [24], meaning that over the time the assessor learned how to take measurement and this resulted in the better/worse scores for the later projects. To mitigate this risk the measurement was monitored by a second evaluator who is one of the authors of the *DA* model.
- The internal validity might be influenced by the selection of the OSS projects for the evaluation. Our sample was selected according to the results of the larger survey [4]. This means that we selected the projects that are already quite popular either when developing Information Systems or when working with them.
- A possible threat to external validity is the nature of our experiment. We did not wish to select any OSS for an actual practical application (e.g., to support real decision). Here, we are only interested in testing the feasibility of the *DA* model.

4.3 Results

Table 2 presents the assessment results. Following the interpretation model presented in Section 2, the documentation type availability is high (*green*) for 19 projects, and it is average (*yellow*) for 9 projects. The documentation information availability is average (*yellow*) for 11 projects and limited (*red*) for 17 projects. Those assessment results confirm that:

- The range of each interpretation category is too broad. For example *DTA* interpretation is *green* both in the case when documents were not found for 4 document types (score 76.92%), and also when documents for all document types are found (score 100%). Similar situation can also be observed for the *DIA* indicator: the difference between the best and the worst evaluated projects at the *red* category is relatively large (e.g., *DIA* of *gcc* is 39.76%, *DIA* of *Omitux* is 15.56%, difference is 24.2%)
- The full range of the assessment is not covered for neither of the indicators. For instance none documentation was interpreted as limited (*red*) or as unavailable (*black*) for the *DTA* indicator; none documentation was found of high (*green*) availability, nor unavailable (*black*) with respect to the *DIA* indicator.

To mitigate these limitations of the interpretation model we refined it. The major idea is to assign to each category of the ordinal scale a sufficient number of values from the internal scale. In addition we observed (specifically for the *DIA* indicator) that it is possible to define four groups of data that is relatively close to each other.

Table 2. Documentation Assessment Results

| OSS projects | Interval scale | | Ordinal scale | |
|------------------------------|----------------|--------|---------------|-------------|
| | DTA, % | DIA, % | DTA, colour | DIA, colour |
| <i>Python</i> | 92.31 | 52.45 | Green | Yellow |
| <i>PLONE</i> | 84.62 | 52.08 | Green | Yellow |
| <i>Thunderbird (Mozilla)</i> | 100 | 50.46 | Green | Yellow |
| <i>Zope</i> | 84.62 | 49.47 | Green | Yellow |
| <i>FreeBSD</i> | 84.62 | 47.31 | Green | Yellow |
| <i>PhPMyAdmin</i> | 76.92 | 46.56 | Green | Yellow |
| <i>NetBSD</i> | 84.62 | 43.25 | Green | Yellow |
| <i>Eclipse</i> | 84.62 | 42.98 | Green | Yellow |
| <i>Writer (OpenOffice)</i> | 76.92 | 42.20 | Green | Yellow |
| <i>Hadoop</i> | 100 | 41.72 | Green | Yellow |
| <i>Xemacs</i> | 100 | 41.28 | Green | Yellow |
| <i>Gcc</i> | 76.92 | 39.76 | Green | Red |
| <i>VLC</i> | 76.92 | 38.74 | Green | Red |
| <i>HTTPD1.3</i> | 69.23 | 34.95 | Yellow | Red |
| <i>Jetspeed</i> | 84.62 | 34.83 | Green | Red |
| <i>Jakarta Struts</i> | 76.92 | 34.27 | Green | Red |
| <i>Evolution</i> | 92.31 | 32.72 | Green | Red |
| <i>Jmeter</i> | 92.31 | 32.08 | Green | Red |
| <i>k3b</i> | 69.23 | 31.52 | Yellow | Red |
| <i>Evince</i> | 84.62 | 30.85 | Green | Red |
| <i>xPDF</i> | 69.23 | 28.96 | Yellow | Red |
| <i>Findbugs</i> | 69.23 | 27.62 | Yellow | Red |
| <i>Nautilus</i> | 76.92 | 26.98 | Green | Red |
| <i>CVSanaly</i> | 69.23 | 24.54 | Yellow | Red |
| <i>Yanolc</i> | 46.15 | 18.41 | Yellow | Red |
| <i>Galeon</i> | 61.54 | 17.01 | Yellow | Red |
| <i>Sup</i> | 61.54 | 16.15 | Yellow | Red |
| <i>Omitux</i> | 53.85 | 15.56 | Yellow | Red |

The refined interpretation model is shown in Table 3. Here the mapping between interval scale and ordinal scale is different for both indicators, and, thus, corresponding to our empirical findings. In Table 4 we present a matrix that define correspondence between two DA indicators based on the refined interpretation model.

Table 3. Refined Interpretation Model for the DA Indicators

| Indicators | Interval scale (%) | Ordinal scale (colour) | Explanation |
|---|--------------------|------------------------|------------------|
| Documentation type availability | [0,00 ... 57,99] | <i>Black</i> | Not available |
| | [58,00 ... 72,49] | <i>Red</i> | DA is limited |
| | [72,50 ... 88,49] | <i>Yellow</i> | DA is average |
| | [88,50 ... 100] | <i>Green</i> | DA is high |
| Documentation information availability | [0,00 ... 21,49] | <i>Black</i> | Is not available |
| | [21,50 ... 36,99] | <i>Red</i> | DA is limited |
| | [37,00 ... 44,99] | <i>Yellow</i> | DA is average |
| | [45,00 ... 100] | <i>Green</i> | DA is high |

Table 4. Matrix of the OSS Project Assessment

| Indicators | | Documentation information availability (DIA) | | | |
|---------------------------------------|--------|--|---|--|---|
| | | Black | Red | Yellow | Green |
| Documentation type availability (DTA) | Green | - | <i>Evolution</i> and <i>jMmeter</i> | <i>Hadoop</i> and <i>Xemacs</i> | <i>Python</i> and <i>Thunderbird</i> |
| | Yellow | - | <i>Jetspeed</i> , <i>Jakarta Struts</i> , <i>Evince</i> , <i>Nautilus</i> | <i>NetBSD</i> , <i>Eclipse</i> , <i>Writer</i> , <i>gcc</i> and <i>VLC</i> | <i>PLONE</i> , <i>Zope</i> , <i>FreeBSD</i> and <i>PhpMyAdmin</i> |
| | Red | <i>Galeon</i> and <i>Sup</i> | <i>HTTDPD1.3</i> , <i>k3b</i> , <i>xPDF</i> , <i>Findbugs</i> and <i>CVSanaly</i> | - | - |
| | Black | <i>Yanole</i> and <i>Ommitux</i> | - | - | - |

“-” means that sample project is not found.

5 Discussion and Conclusions

In this section we discuss our results and present conclusions. We also situate our proposal into the state of the art, and provide some future research directions.

5.1 Conclusions

In this paper we presented a quality model to assess documentation availability for the OSS products. Our major discussion includes the analysis of the validity both at theoretical and at empirical level. Firstly, following the criteria defined in [16] we confirm the theoretical validity of the *DA* model, including its metrics, metric scales, and quality estimation process.

Next, we investigate the empirical validity through the performance test that helps to assess feasibility of our proposal. Our work results in the refinement of the interpretation model for the documentation availability indicators, as shown in Table 3. The new interpretation model is based on the empirical results and is more flexible to judge about the quality of the documentation itself: for example, using the refined model one gets the assessment data that are clustered into nine categories (using previous model it was separated to only three). This expands the assessment scale and gives more flexibility to judge about the documentation quality, especially, when the assessment purpose is selection and acquisition of OSS software to practice.

Although in the performance test we did not have as goal to select the best-documented OSS projects, the results points out few important suggestions for the OSS documentation improvement. For instance, availability of documents of a certain type does not guarantee availability of the information within these documents (e. g., *Evolution* and *jMeter*). This means that, although there are many documents for these OSS projects, those documents do not contain a sufficient level of guidelines, help, instructions, and etc. for the stakeholders.

We can also notice an opposite trend: some documents contain a lot of useful information, where the overall project documentation availability might be of a lower

degree (e.g., *PLONE*, *Zope*, *FreeBSD* and *PhPMyAdmin*). This means that the existing project documentation is of high quality to satisfy goals only for a limited group of stakeholders (for instance, there exist high-quality product installation and application documents to satisfy goals of product acquirers, but there is no product development documentation to fulfil goals of product developers).

A limitation of the refined interpretation model is that there is missing sample of projects that would illustrate its full feasibility (some of the cells in Table 3 are still empty). This does not mean that there is no such sample of project documentation; it might be the case that we did not happen to select such.

5.2 Related Work

Software development standards [7, 8, 9, 10, 11, 12] suggest documentation templates for different development stages. These templates include structural guidance to prepare documentation, however the standards do not explain how to assess quality of the documentation. Following the software development standards, the *DA* method suggests the guidelines on how to assess the structural completeness of documentation. Later, based on the structural completeness results, the information completeness is investigated.

Literature reports on few proposals to measure documentation quality. Davis *et al.* [2] define a set of qualitative characteristics (e.g., non-ambiguity, completeness, correctness, consistency, verifiability, traceability, modifiability, and others) to measure quality of requirements specifications. Further in [4] a comprehensive metrics, like page count, readability, pages per day and software dependent metrics, are proposed. However both works do not go beyond the metric definition phase. Elsewhere in [22], a process to assess documentation content quality is defined. The documentation content is assessed through the following characteristics: ownership, readability, accuracy, thoroughness, format, accessibility, currency, effectiveness, accountability, and ease of update. The assessment is summarised into a quality/value matrix comprising four subjective values: poor, fair, good, and excellent.

The *DA* method is specifically dedicated to assess the documentation availability. Similarly to [2] it relies on the qualitative characteristics, and analyses organisation and structure of the documentation like in [4]. During measurement the evaluators need to deal with the certain degree of subjectivity. Differently from the mentioned works, the *DA* method provides a systematic and comprehensive process (Fig. 1) to assess documentation of different document types and to fulfil goals of various stakeholders. In addition to all these works the *DA* method also provides the indicators and explains how it is possible to interpret these indicators.

5.3 Future Work

In this paper we have described the performance test to investigate the feasibility of the *DA* model. This is the first step towards a model empirical validation. The next step is the correctness test, where we are planning to perform an interview of community members on the documentation quality of the selected OSS projects. This would allow us to compare two sets of the data for the documentation availability and will answer the question about the empirical validity of the *DA* model.

Although we have developed the availability assessment model for the documentation quality of the OSS product, our future plan is to adapt it for documentation assessment of any type of systems. This is possible because we have based our model on the standards of the software development, like [7, 8, 9, 10, 11, 12]. The major difference for the measurement process stands in the definition of the measurement goals and scope. For example for the general software documentation assessment, the goal of becoming a member of the community might not be applicable. However other goals – like obtaining the products (especially for the commercial-of-the-shelf product selection), using the (general) product, or improving the current product – are often within the scope of the general system assessment.

Finally our long-term research plan includes expansion of quality model (Fig. 1) with the *accuracy* measurement. This will require understanding of the various metrics for documentation readability, understandability, consistency and others [2].

Acknowledgments. We would like to thank Judith Chambou Simo for her contribution to this research. We also thank the partners of the QualOSS project for the stimulating discussions on the OSS quality.

References

1. Boehm, B., Brown, J.R., Kaspar, J.R., Lipow, M., MacLoed, G.J., Merritt, M.J.: Characteristics of Software Quality, TRW Series of Software Technology. North-Holland Pub., Amsterdam (1978)
2. Davis, A., Overmyer, S., Jordan, K., Caruso, J., Dandashi, F., Dinh, A., Kincaid, G., Ledebor, G., Reynolds, P., Srimani, P., Ta, A., Theofanos, M.: Identifying and Measuring Quality in a Software Requirements Specification. In: Proceeding of the 1st International Software Metrics Symposium, pp. 141–152 (1993)
3. Deprez, J.-C., Haaland, K., Kamseu, F.: QualOSS Methodology and QualOSS Assessment Method, Deliverable D4.1, <http://www.qualoss.org/deliverables> (last checked 28.02.2010)
4. Le Vie Jr., D.S.: Documentation Metrics: What Do You Really Want to Measure? http://www.stc.org/intercom/PDFs/2000/200012_06-09.pdf (last checked 28.02.2010)
5. El Emam, K., Drouin, J.-N., Melo, W.: SPICE. In: The Theory and Practice and Software Process Improvement and Capability Determination. IEEE Computer Society, Los Alamitos (1998)
6. Forward, J.A., Lethbridge, T.C.: Qualities of Relevant Software Documentation: an Industrial Study, http://www.site.uottawa.ca/~tcl/gradtheses/forward/papers/forward_icse2003_sub.pdf (last checked 28.02.2010)
7. IEEE: IEEE Recommended Practice for Software Design Descriptions, IEEE Std 1016-1998 (1998)
8. IEEE: IEEE Standard for Software Maintenance, IEEE Std 1219-1998 (1998)
9. IEEE: IEEE Recommended Practice for Software Requirements Specification, IEEE Std 830-1998 (1998)
10. IEEE: IEEE Standard for Software Project Management Plans, IEEE Std 1058-1998 (1998)
11. IEEE: IEEE Standard for Software Test Documentation, IEEE Std 829-1998 (1998)
12. IEEE: IEEE Standard for Software User Documentation, IEEE Std 1063-2001 (2001)

13. ISO/IEC: Information Technology – Software Product Evaluation– Quality Characteristics and Guide Lines for their Use. ISO/IEC IS 9126, Switzerland (1991)
14. Izquierdo, D., Herraiz, I.: Qualoss 3.1 measurement Targets, Deliverable D3.2, <http://www.qualoss.org/deliverables> (last checked 28.02.2010)
15. Jazzar, A., Scacchi, W.: Understanding the Requirements for Information System Documentation: an Empirical Investigation. In: Proceedings of the ACM Conference on Organizational Computing Systems (COOCS 1995), pp. 268–279. ACM, New York (1995)
16. Kitchenham, B., Pfeegeer, S.L., Fenton, N.: Towards a Framework for Software Measurement Validation. *IEEE Trans. on Soft. Eng.* 21(12) (1995)
17. Loconsole, A., Borstler, J.: Theoretical Validation and Case Study of Requirements Management Measures. Technical report UMINF-03.02, Department of Computing Science, Umea University (2003)
18. Matulevičius, R., Kamsu, F., Habra, N.: Measuring Open Source Documentation Availability. In: Proceedings of the 12th International Conference on Quality Engineering in Software Technology (CONQUEST 2009), dpunkt.verlag GmbH, pp. 83–102 (2009)
19. McCall, J.A., Richards, P.K., Walters, G.F.: Factors in Software Quality, RADC TR-77-369, Vols I, II, III, US Rome Air Development Center Reports NTIS AD/A-049 014, 015, 055 (1977)
20. Paulk, M.C., Curtis, B., Chrissis, M., Weber, C.: Capability Maturity Model for Software: Version 1.1. Technical Report SEI-93-TR-24, Software Engineering Institute, Carnegie Mellon University (1993)
21. Singh, N.: Maintaining Quality Control in Documentation. In: Proceedings of Society for Technical Communication (2002)
22. Schiesser, R.: How does your process documentation measure up?, http://articles.techrepublic.com.com/5100-10878_11-1053164.html (last checked 28.02.2010)
23. Soto, M., Ciolkowski, M., Deprez, J.-C., Ruiz, J., Herraiz, I., Campos, C.G., Matulevičius, R.: Metrics and Indicators of the Standard QualOSS Assessment Method, Deliverable D4.2, <http://www.qualoss.org/deliverables> (last checked 28.02.2010)
24. Wohlin, C., Runeson, P., Høst, M., Ohlsson, M.C., Regnell, B., Wesslen, A.: Experimentation in Software Engineering. Kluwer Academic Publishers, Boston (2002)

Emerging Challenges in Information Systems Research for Regulatory Compliance Management

Norris Syed Abdullah¹, Shazia Sadiq¹, and Marta Indulska²

¹ School of Information Technology & Electrical Engineering, The University of Queensland,
4072 Brisbane, Australia

{norris,shazia}@itee.uq.edu.au

² UQ Business School, The University of Queensland, 4072 Brisbane, Australia
m.indulska@business@uq.edu.au

Abstract. Managing regulatory compliance is increasingly challenging and costly for organizations world-wide. While such efforts are often supported by information technology (IT) and information systems (IS) tools, there is evidence that the current solutions are inadequate and do not fully address the needs of organizations. Often such discrepancy stems from a lack of alignment between the needs of the industry and the focus of academic research efforts. In this paper, we present the results of an empirical study that investigates challenges in managing regulatory compliance, derived from expert professionals in the Australian compliance industry. The results provide insights into problematic areas within the compliance management domain, as related to regulatees, regulations and IT compliance management solutions. By relating the identified challenges to existing activity in IS research, this exploratory paper highlights the inadequacy of current research and presents the first industry-relevant compliance management research agenda for IS researchers.

Keywords: Regulatory Compliance, Business Information Systems, Empirical Study.

1 Introduction

Compliance involves ensuring that business processes, operations and practice are in accordance with a prescribed and/or agreed set of norms. Even though predominantly viewed as a burden by organisations [1], failing to comply is no longer an option [2, 3]. Non-compliance may not only result in the possibility of losing customers and damaging reputation, but can also lead to legal action. A number of corporate scandals - Enron, WorldCom (USA), HIH (Australia), Societe Generale (France) and, most recently, Satyam (India), to name a few - have exhibited this situation.

In addition, there is a general consensus that there will be an upsurge of regulatory reform as a response to the events that led to the global financial crisis. Developing strategies to manage inevitable regulatory shifts that emerge from government and global reactions to the financial crisis is going to be high on corporate agendas in the coming years. This situation is bound to put pressure on organisations already struggling with the economic downturn.

With compliance expectations on the increase, evidence suggests that organisations experience difficulties in managing compliance expectations and are increasingly concerned with high costs associated with compliance [4]. Indeed, spending on compliance is steadily increasing [5, 6]. With each new introduced regulation new challenges arise [7-9]. The inevitability of coping with compliance pressures identifies a need for new IT and IS solutions to compliance management and denotes a need for evolution of current IT and IS approaches such that they are better able to support the fast-changing regulatory compliance management field. Any developed solutions, to be adequate, need to be informed by industry practice and expert advice. The development of solutions in this domain without input from industry experts and professionals will only serve to increase compliance management spending without delivering on the promise of suitable IT and IS tools to alleviate compliance management problems.

In this paper we take steps to address this need through an empirical study with compliance management professionals. The main goal of the reported study is to investigate compliance management issues and challenges faced by industry, as perceived by compliance management experts, to investigate where IS research can contribute and how IS tools should evolve to support industry. We conduct a gap analysis of the perceived challenges with existing research in IS and identify a set of issues and challenges that should drive the future agenda of IS researchers.

2 Approach and Methodology

To extract issues and challenges, we utilise semi-structured interviews. Interviews are an established and popular means of carrying out qualitative enquiry in the fields of social sciences. We use a semi-structured interview type to provide the opportunity for participants to think about the challenges in compliance management and reflect on and relate their experiences [10]. The interview team consisted of two experienced empirical researchers, one with the role of the main interviewer and the other with a support role of note taking and further probing. The interviewers' domain knowledge and expertise with the interview method is an essential element for success in these semi-structured interviews.

2.1 Data Sampling, Participants and Protocol

The nature of the study led us to adopt a deliberately selective sampling approach to participant involvement. We were motivated to ensure that participants had extensive experience in the domain and would therefore provide an insightful and accurate reflection of the state of compliance management in practice. To that end, we enlisted the help of the Australasian Compliance Institute (ACI) and obtained a selection of experienced contacts with insight into both the mature and immature stages of compliance management in organisations. Eleven participants were invited to participate and all eleven agreed with no incentives present for participation. Accordingly, eleven Australian compliance management experts were interviewed in the last quarter of 2007. Typical roles interviewed were those of senior compliance management

advisors and consultants in large organizations that provide both advisory and auditing services in the context of regulatory compliance. Among the eleven, nine possess more than 10 years of experience in the field and the other two have five and seven years of experience in the field respectively.

The semi-structured interview protocol¹ was designed and pilot tested to elicit free flowing information from the interviewees. The protocol consisted of high level questions that captured the experience of the interviewee, their opinions relating to regulations and related challenges, as well as their experiences and observations of challenges in compliance management practice. The questions were open-ended in nature so as not to bias the interviewees. The researchers relied on probing to identify specific challenges when an interviewee indicated a limitation they had experiences.

The protocol included three main sections. The first section aimed to establish the context of the interview session. This section consisted of demographic inquiry such as role, experience and organisation description. The second section asked interviewees to provide opinions about heavily regulated industry sectors, what compliance responsibilities existed in their organisation and their awareness and experience with current IT- and IS-related tools in use. The third section of the protocol aimed to obtain explanation about the compliance-related factors - such as issues, hurdles and solutions. Interviewees were asked to identify these factors in relation to customers, regulations and solutions. In addition, participants were also encouraged to incorporate examples during elaboration of key points.

2.2 Data Analysis

All interviews were transcribed, anonymised, and analysed using a multi-coder approach and NVivo as the supporting tool. The multi-coder approach was used to reduce coder bias in the analysis of the text and multiple rounds of analysis were carried out for each transcription. As the study was exploratory in nature, all factors emerged from the interview data. To facilitate the exploration of factors, an initial meta-level node structure was derived from the interview protocol – *viz.* responses related to regulatees, regulations and solutions. This structure was used and expanded upon each time a new issue, challenge, problem, regulation, etc, was identified.

The detailed coding, using the initial node structure as the basis, was conducted in the first round by a third researcher, in a few iterations of the full transcripts so as to capture the most detail. After each iteration, the coding node structure and associated interview data was examined and revised before the next iteration of interview data coding took place. The final coding and the resulting expanded coding structure were then independently reviewed and re-coded separately by two researchers who identified the initial node structure. Following this individual analysis, the two researchers then jointly discussed each node and its contents and refined the coding structure to reduce overlaps and refine aspects of some identified issues (*i.e.* split nodes). The researchers then jointly again analysed each transcript to ensure that all relevant detail was captured and correctly codified.

¹ Due to space limitations the interview protocol is omitted from this paper. The interview protocol is available from the authors on request.

3 Results

In the following we present an in-depth discussion of the main challenges identified in the interview sessions. Based on the analysis of the expert interviews, Figure 1 shows the number of participants referring to each regulation and number of references made respectively. Among all regulations referred to by the participants, Anti-Money Laundering (AML) was the most frequently referred to and also the only one discussed by all the participants. This finding is perhaps not surprising given the recent introduction of the regulation. Australian Standards was the second most frequently discussed, followed by Financial Services Regulation (FSR), Organisational Health and Safety (OHS), Trade Practices Act, Corporation Act and Sarbanes-Oxley (SOX).

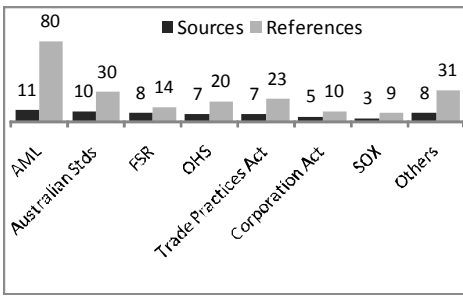


Fig. 1. Regulations Identified by Experts

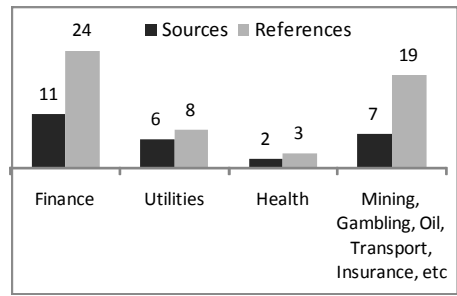


Fig. 2. Industry Sectors Identified by Experts

In terms of the industry sectors that are considered to be highly regulated, Figure 2 similarly depicts the number of references/participants regarding identification of the most regulated markets. The experts indicated that the financial sector is the most heavily regulated industry - addressed by all the participants with a total of 24 supported references. This finding correlates the earlier indication of a strong AML focus.

After understanding the landscape of various sectors and their compliance requirements, we turned to investigating the challenges that organisations face in their compliance management activities. The discussion of the identified challenges and expert opinion is provided in the next sub-sections, logically separated into challenges and problems related to customers (i.e. regulatees), regulations and solutions. We precede the discussion of the challenges with the table showing the challenges and the

Table 1. Customer Factors

| FACTORS | SOURCES | REFERENCES |
|--|---------|------------|
| Lack of Compliance Culture | 10 | 46 |
| High Cost | 9 | 43 |
| Lack of Efficient Risk Management | 9 | 37 |
| Difficulties in Creating Evidence of Compliance | 8 | 29 |
| Lack of Perception of Compliance as a Value-add | 8 | 25 |
| Lack of Understanding of its Relevance to Business | 6 | 12 |
| Lack of Communication among Staff | 4 | 4 |

number of participants who identified them, as well as the frequency of the identifications (number of references). Tables 1, 2 and 3 present factors relating to customers, regulations, and solutions respectively.

3.1 Factors Relating to Customers

Lack of Compliance Culture. All the participants indicated that culture plays a vital role in inculcating compliance. One of the participants indicated: "... compliance doesn't mean a thing if the culture isn't right within the organisation." Culture refers to the overall compliance culture of the organisation, which involves employees' perspective towards the organisation, what the organisation stands for, its customers, investors, regulators and fellow colleagues [11]. A good culture, though difficult to achieve [12] can promote a positive attitude towards legal compliance activity at all levels within an organisation.

Several issues relating to culture were identified from the expert interviews. The core issues related to top level management beliefs and mindset. The board often sees compliance as one of the least value-added activities for the organization. Consequently, the operational level often lacks guideline and advice, which further results in naive and inexperienced compliance approaches among employees. This situation is made worse because compliance officers often have little influence in the management board of organisations. A participant indicated that: "... they need to get top level buy in and then try – if there's no buy in from senior management, it's not going to run." This issue has also been highlighted by KPMG [13], where it is suggested that balancing risk and controls with business improvement begins with the identifications of priorities and opportunities from a high-level, or so called "top-down" perspective.

Similarly, the corporate mindset is affected directly by belief of employees. One participant stated: "... Even if it's a black letter law, it's still a mindset. ... Like you don't want to know that you're reporting valid financial information to the market..." This statement illustrates that some organisations lack willingness to accept compliance activities within business operations. Further, one of the participants pointed out that some organisations tend to allocate junior (those who "either had nothing to do in projects once projects finished or were being managed") and "non-star" resources to risk and compliance section. The experts also indicate that some organisations are reluctant to provide IT/IS tool support for compliance staff. Another culture related issue is the lack of pro-active culture that leads to low compliance achievement for the organisation. The problem is propagated to the mindset of the organisation, in which "compliance is done for the sake of compliance instead of compliance for the sake of good business."

High Cost. Cost of compliance is one of the vital issues that make organisations hesitant to get a compliance framework in place [14]. According to experts, one of the compliance cost-related issues is the size of the company, especially affecting small to medium sized companies (SMEs). Due to human resource and financial capital limitations, these companies are found to struggle to put a compliance framework in place - "... the smaller you are, the less capacity you've got to even feel confident about it

because you haven't got the skills either in capital, technology, or finance to actually put that control framework in place" (interview data). In other words, despite differences in organisational size, small companies those have an equivalent "complexity function" to that of large enterprises have to pay for an equivalent compliance frameworks without an equivalent budget.

Lack of Efficient Risk Management. Another issue recognised by participants is that of reluctance to allocate adequate resources to manage the risks, while being aware that those risks exist. As one of the participants stated: "... in a risk identification sense, you identified your risks and you said we've got to look at these, but then you didn't resource to actually address that..."

On the other hand, it is also possible that the issue is linked to the mindset of the employees. The employees might see that it is too risky to be a compliance officer: "... There's personal liability attaching to everything now. If you look at money laundering legislation for example, I've got clients who say "Who'd want to be the money laundering reporting officer? You could go to jail..." Complicating this issue is the fact that regulatory risk is not transferable (e.g through purchasing insurance). In risk management, there are several approaches that organisations use in dealing with risk: reduce, mitigate, accept or transfer the risks [15, 16]. Organizations have to rely on an effective approach to reduce regulatory risks given that the consequence of failing to report breaches is severe.

According to the findings, organisations tend to see compliance as being one of the risk management frameworks - this situation is found to be consistent with general organisation structure [13]. Developing effective risk management begins with a clear understanding of an organisation's appetite for risk [17]. However, this is difficult to achieve. It depends on the ability of the risk assessment framework to keep up with legislative and regulatory changes. It is a nature of legislations and regulations to change in order to capture changes and growth in business. Risk assessment, on the other hand, needs to be monitored and updated on an ongoing basis so that it can capture those changes. As a result, organisations are faced with high monitoring costs, which, in turn, prevent them from having effective risk management.

Difficulties in Creating Evidence of Compliance. One of the experts highlighted "...got to do the right thing and not only they have to do the right thing, they have to demonstrate that they're doing the right thing." This revealed that organizations need for effective techniques to demonstrate their business conformity to obligations. Other experts also agreed, e.g. "... People need systems to improve the internal efficiency to be able to demonstrate that compliance and document stuff." These views confirm the need for an appropriate control framework to facilitate internal controls, especially incident recording and reporting.

Lack of Perception of Compliance as a Value-add. Many organisations see little (or no) value-add of compliance controls being embedded in their business processes. Those organisations, having documented their business activities, argue that they see no returns for the time consuming and expensive documentation: "The business never gets anything back, so there's no value provided back to the business from the data that's collected and then heaven forbid, stuff gets reported to the board without the

business knowing about it.” Moreover, some organisations believe that risk and compliance frameworks add complexity to business: “risk and compliance actually makes it harder because they can’t visualize the business so they get attracted to adding systems into the processes that serve their purposes but don’t serve businesses purposes.” Furthermore, some organisations claim that the only business benefit derived from compliance frameworks is that they avoid getting fined by regulators.

Lack of Understanding of its Relevance to Business. Organisations face difficulties in relating obligations to their business; that is, which rule(s) is/are relevant to which business objective(s) and activities. To overcome such difficulties, it is required that regulations and legislations are interpreted in relation to, and mapped to, business processes by experts who deeply understand both the legal and the operational aspects of the organisation. Experts also recognise this issue, e.g. - “The rules and obligations are not the problem; it’s the organisation’s capacity to understand how it needs to run its business to achieve its objectives.” As a result of lack of understanding on how to correctly embed rules and obligations in their business, organisations are found [18].

Lack of Communication among Staff. Based on the interview data, the lack of efficient communication channels within the organisation is one aspect that prevents an organisation from having an effective compliance framework in place is. Once there is a change in obligation, organisations find it complicated to communicate the change: “... you can’t really have people in a business unit just by themselves ... So they have to be linked somehow and another situation is the fact that if they are not linked into that business unit somehow, they tend to be left ... they don’t participate in team meetings, people sort of isolate them and that’s a real danger” In addition, another communication issue is associated with compliance monitoring, where risks and compliance problems that are identified in business operations are not reported back to the board.

3.2 Factor Relating to Regulations

Frequent Changes in Regulations. The fact that regulations, legislations, government rules and laws regularly change means that organisations struggle to keep up with the new requirements [13, 19, 20]. Updates of existing compliance frameworks and internal controls to comply with the new obligations are a key problem, as highlighted by one of the participants: “... because of the rapid change around regulatory reform, we’re finding organisations becoming more inefficient in trying to meet these obligations.”

Table 2. Regulations Factors

| FACTORS | SOURCES | REFERENCES |
|---------------------------------|---------|------------|
| Frequent Changes in Regulations | 7 | 12 |
| Legislation Weaknesses | 6 | 11 |
| Inconsistencies | 5 | 8 |
| Overlap in Regulations | 3 | 7 |

Legislation Weaknesses. According to the experts, the challenges related with legislations were mostly found in principle-based legislation. Principle-based legislation puts the approach on meeting the legal obligations to be decided by the regulated party. Although this approach allows organisation to be creative in applying the legislations, it exposes the organisation to the risk of making inappropriate interpretations of legislation.

Inconsistencies. The problem of consistent application of common standards across several jurisdictions (states and territories) is recognized as one of the central problems of regulation [21]. As a result, some organisations choose to comply with those inconsistent regulations, which generate negative consequences, particularly compliance cost [22]. Not only are regulations inconsistent domestically, but also internationally. For example, "... most countries impose regulation for those entities that they regulate, not only in the home jurisdiction, but offshore, and if, let's take Deutchse Bank, if the German authorities impose restrictions or regulatory frameworks on Deutchse Bank, they've got to apply it here in Australia. That may or may not be consistent with domestic law here."

Overlap in Regulations. One of the main challenges with regulations is the problem of duplication. In the Australian context, for example, the duplication is mainly between the states and the Commonwealth [21], due to multiple legal parties. Organisations are affected because they are required to demonstrate evidence of compliance multiple times. As one expert exemplifies: "... I kept on seeing that I was going to organisations exactly the same obligations coming through in five different pieces of legislation and then the people out of the business pushing back saying 'But I've already answered this in relation to ...'"

3.3 Factor Relating to Solutions

Lack of Holistic Practices. Even if a high-quality compliance framework is in place, organisations can be stagnant in terms of improvement if the framework is not properly managed. It is believed that compliance must be cascaded through every layer in the organisation, starting with clear direction from the top, and then deployed appropriately at each level [23]. The relationship between the two is captured by a quote of one of the participants: "... governance is around the right behaviors, appropriate risk management, corporate social responsibility, ethics, managing, reporting to stakeholders,"

Table 3. Solutions Factors

| FACTORS | SOURCES | REFERENCES |
|-----------------------------------|----------------|-------------------|
| Lack of Holistic Practices | 5 | 8 |
| Lack of IT Support/Tools | 11 | 103 |
| Lack of Compliance Knowledge Base | 7 | 35 |

Lack of IT Support/Tools. Despite the rising investment from software vendors and entrepreneurs in governance, risk and compliance (GRC) software products, organisations are struggling to correctly identify the tools and its suitability to their

requirements. The features identified by Gartner [24] - reporting, dash-boarding, remediation management, business process modelling, risk management and support for multiple regulations across multiple business units - were mostly also identified by the participants. The participants also highlighted that lack of IT support/tools are related to usability and comprehensiveness of the tools, the supported learning and training program, tools features such as monitoring and reporting, self-assessment, management, newsfeed, alert, and updates. Participants highlighted a need for tools relating to reporting/monitoring, self assessment, newsfeeds/alert, learning and management. Moreover, participants also stressed that tools should include the ability to deliver not only regulatory compliance but also business benefits.

In particular, participants emphasized the important role played by monitoring tools in compliance management: "... you can't monitor customer's transactions manually, ... there's millions of transactions." It is also important that monitoring tools be able to oversee all business operations so that anomalies can be traced back to the source of the problem if a breach occurs. This requirement leads to the need of breach reporting and incident recording functionality. Furthermore, as one participant pointed out - "...reporting tools should be able to effectively filter out irrelevant data and provide meaningful reports to the audit committees". So that instead of acquiring an external audit firm, organisations can obtain internal audit management tools that can provide regular audits, and keep the risk library small.

Lack of Compliance Knowledge Base. Generally, the advisory program provides the necessary guidance to handle complex regulatory requirements. Experts highlighted that advisory related challenges centered on the development of compliance knowledge base, linking regulations to business processes, and the reliance to comprehensive guidelines and frameworks. Many consulting firms offer such a service including of course - the Big Four - i.e. KPMG, PricewaterhouseCoopers, Ernst & Young and Deloitte Touche Tohmatsu. According to the experts, some organisations that already have compliance frameworks in place continue to have difficulties in carrying out tasks properly as a result of lack of comprehensive knowledge. Advisory services contribute in this space by assisting on compliance strategies, and to overseeing and evaluating the overall performance of compliance outcomes. One of the main foci of the above is to define an appropriate training program. The experts confirmed the lack of effective training programs.

Issues related business processes were also identified - in particular linking or embedding controls/regulations to processes - and are focused on advisory services that analyse existing processes wrt. relevant regulations.

4 Information Systems Research on Compliance Management

In [8], the first comprehensive snapshot of the development and focus of compliance management related research in the Information Systems (IS) discipline is presented. The study includes papers from premium Information Systems journals (as promoted by the Association for Information Systems), and some additional popular journals in the discipline. Aiming to introduce a well-informed research agenda, we further extend the report to include reputed conferences in the discipline. Our target is to further identify the existing IS research that contributes to solving compliance management problems.

The results are presented within a framework that was developed to establish relevance and analyse the contributions². Within this framework, case study and exploratory papers are differentiated from papers that provide a solution to a compliance management related problem.

Table 4 shows the breakdown of papers relevant to compliance management and their source of publication. Out of 19637 articles, 232 articles matched the context. Although the number is relatively low, the roles of IS or IT as enablers of regulatory compliance have increased year by year (details below).

Table 4. Sources and Frequency of Publication

| SOURCES (Journals) | TOTAL | Relevant Articles | % | SOURCES (Conferences) | TOTAL | Relevant Articles | % |
|--------------------|-------|-------------------|-----|-----------------------|-------|-------------------|-----|
| CAIS | 659 | 16 | 2.4 | BPM | 189 | 7 | 3.7 |
| BPMJ | 336 | 5 | 1.5 | ACIS | 906 | 28 | 3.1 |
| JAIS | 158 | 2 | 1.3 | CAiSE | 346 | 9 | 2.6 |
| JI&M | 502 | 4 | 0.8 | ICIS | 959 | 14 | 1.5 |
| CACM | 2178 | 17 | 0.8 | PACIS | 1025 | 14 | 1.4 |
| JISR | 199 | 1 | 0.5 | AMCIS | 3822 | 46 | 1.2 |
| EJIS | 382 | 2 | 0.5 | HICSS | 4517 | 49 | 1.1 |
| MISO | 281 | 1 | 0.4 | ECIS | 1489 | 17 | 1.1 |
| | | | | ER | 400 | 2 | 0.5 |

The next step in the analysis carried out the classification with respect to the type of publication, *viz.* case study/exploratory and solution. As expected in an emerging research domain, the majority of the publications were found to be in the case study or exploratory paper category - 188 (81%) of the articles are case study/exploratory articles and 40 (17.2%) are solution articles. However, there are four (1.7%) articles that matched both types of articles. The results suggest that research on regulatory compliance solution has being initiated but remains still in the early exploratory stages.

Furthermore, we were interested to determine the emergence of compliance management research in Information Systems publication outlets. The breakdown of compliance management research per year of publication is shown in Figure 3.

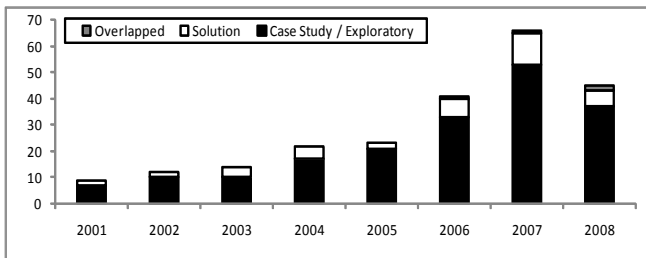


Fig. 3. Distribution of Article per Type per Year

² Each paper was prepared and included in a full text search for the purposes of identifying contributions relevant to the compliance management domain. Full text searches were conducted on the data set, using a keyword of “compliance” and “compliant”.

The figure shows an upward trend in compliance management research in Information Systems, with a spike of publications in 2007. We posit that this finding is in line with the increased focus on SOX Act of 2002 and also an early focus on HIPAA, given the lag of publishing in the Information Systems discipline. Prior to this event, little literature on compliance management exists, despite some other regulations having already been proposed.

Following this analysis, we carried out further classification for the articles that were classified as solution articles. This classification involved 44 solution articles and also 4 articles that contain both (case study and solution) discussion. These articles were reviewed to identify the focus of the solution in relation with challenges identified in section 3. The study reveals that 29 out of 44 articles offer a preventive (before-the-fact) solution, 14 offer a detective (after-the-fact) solution, while the remaining (one) offers both solutions.

5 Research Agenda for Compliance Management

In this section we present an analysis of the gaps between expert opinion and current status of IS literature with regard to published Information Systems compliance management solutions. A summary is provided in Table 5.

Due to space limitations it is not possible to discuss the contributions of the 44 papers identified as contributing to compliance solutions. The actual solutions presented in the papers vary substantially. Some of the papers, for example, [25] addressed detective types of solution and introduced the Hippocratic Database Compliance Auditing component, which facilitates audits in E-health records.

In [26] the researchers introduce the 'compliance by design' approach that proposes a compliance regimen with a preventative focus. Similarly [14] present a high level view of regulatory compliance through a policy-based framework for integrating regulatory compliance tasks with business processes.

We observe that the growing focus on technical aspects of compliance, is balanced by research on business or management aspects. For example, [27] detailed the development and application of an evaluative data model for ISO 9000 compliance. On the call to facilitate compliance with HIPAA, [28] introduce a framework that provides a useful way of identifying and analysing the training needs of organisations with diverse user communities and continuous change.

As shown in Table 5, the solution challenges, i.e. lack of holistic practices, lack of IT supports/tools, and lack of compliance knowledge base, have received most attention from the IS researchers with 13, 14 and 10 matching solutions respectively. This demonstrates that the focus of IS research community is slanted towards providing solutions either in form of best practices, automation, or guidelines. The analysis also exposed that regulatee challenges *viz.* high cost, lack of efficient risk management, and lack of perception of compliance as a value-add; despite receiving high attention from the experts, have not yet been addressed adequately. Difficulties in evidencing compliance, although addressed in research in 2002 and 2003, shows drought from 2004 to 2008. This is contrary to experts' opinion, which stresses difficulties in evidencing compliance and the need for appropriate incident recording and reporting mechanisms.

Table 5. Industry Challenges vs. Current Research Focus

| | | INDUSTRY CHALLENGES | | | | | | | |
|-----------------------------------|--|---------------------|------|------|------|------|------|------|------|
| | | SOLUTIONS (by year) | | | | | | | |
| | | 2001 | 2002 | 2003 | 2004 | 2005 | 2006 | 2007 | 2008 |
| Customers | Lack of Compliance Culture | | | | | | | 1 | 1 |
| | High Cost | | | | | | | | |
| | Lack of Efficient Risk Management | | | | | | | | |
| | Difficulties in Creating Evidence of Compliance | | 1 | 1 | | | | | |
| | Lack of Perception of Compliance as a Value-add | | | | | | | | |
| | Non-proactive | | | | | | | | |
| | Lack of Understanding of its Relevance to Business | | | | | | 1 | 2 | 1 |
| Lack of Communication among Staff | | | | | | | | | |
| Regulations | Frequent Changes in Regulations | | | 1 | 1 | | | | 1 |
| | Legislation Weaknesses | | | | | | | | |
| | Inconsistencies | | | | | | | | |
| | Overlap in Regulations | | | | | | | | |
| Solutions | Lack of Holistic Practices | | | | 4 | 1 | | 6 | 3 |
| | Lack of IT Support/Tools | 2 | 1 | 1 | 1 | 1 | 5 | 1 | 3 |
| | Lack of Compliance Knowledge Base | | | 2 | | | 3 | 4 | 1 |

In terms of solutions for regulations related challenges, very few solutions can be found and all (three) relate coping with frequent changes. Other challenges i.e. legislation weaknesses, inconsistencies, and overlap in regulations are not addressed, which is not surprising as this is more in the legal arena and controlled by government agencies, legislation authority or standardisation bodies.

The review of the related papers as well as the interview transcripts allowed us to extract key aspects where IS research can create results and value for organizations in meeting their compliance obligations. In the discussion below, we highlight the key challenges that need attention from the IS research community and indicate how the challenges relate to addressing industry challenges:

First and foremost, there is an urgent need for proper **benchmarking studies** to help address the challenge of *high cost*. Particularly for SMEs, there is high cost and great difficulty in **measuring the adequacy of controls** for principles based regulations where the onus is on the organization to design an appropriate compliance regimen. Benchmarking and best practice studies will allow improvement of controls effectiveness, a reduction of costs, and an improved potential to deal with resistance to change through demonstrating methods used by others. Such additional knowledge can further help alleviate the perception of *legislation weaknesses* in principles based regulations and consequently promote regulation acceptance.

In a related manner, there is also a need for investigation of **process reference models** relating to various regulations. A focus on the development of such reference models and the study of the impact of the use of such models in organizations (i.e. impact on compliance management spending, frequency of breaches, etc) is largely missing in Information Systems research. The development of proven reference

models, however, may significantly lessen the cost of compliance management in organizations.

The *culture of compliance* is ingrained in the daily rituals of each of the firm's employees, including senior management, who must learn to lead by example [12]. There is a clear lack of Information Systems research on **organisational behaviour**. In particular we see a need for investigation of how IT and IS tools can be used to incentivize employees to 'do the right thing' and adapt their practices. There is also a need for the development of relevant IT and IS tools that can help facilitate employee training for compliance management, promote *communication among staff* and increase organizational capacity to manage its *compliance knowledge base*.

How the compliance (and *risk*) factor interrelates with the operations of business units is understudied, with only a small number of researchers working on the **conceptualisation of compliance and risk** requirements per se let alone their interrelationships with business processes and business activities. A comprehensive and well-grounded conceptual model for compliance and risk is needed.

Further to the point above, tools and methods are needed to **annotate, enhance, analyse and simulate business models** with compliance and risk modeling elements. This will facilitate better coordination between an organization's compliance and business functions and help employees understand *compliance value* and *business relevance*.

Although reporting and monitoring tools of high sophistication are available, there is little development towards tools that provide **specialized solutions in monitoring and analysing** compliance related data (partly due the absence of any generic conceptual models for GRC), thus causing big problems for organisations required to create *evidence of compliance*. Accordingly, we see a need for affordable IT and IS tools that facilitate compliance management self-audits and compliance monitoring activities in general. Furthermore, there is also a clear need for tools that facilitate the identification of non-compliance processes with respect to a given regulation.

Although *frequency of change*, as well as *inconsistency* and *overlaps* in regulations is beyond the realm of IS research, studies to understand the **impact of regulation changes** (inconsistencies and overlaps) can promote better understanding of the cost of compliance and allow business to lobby for regulatory reform where needed. Multi disciplinary research is warranted in order to cover legal, business and IT aspects. From an Information Systems perspective, there is a need for solutions that can filter out updates that are not relevant to a given organization or industry sector, thus reducing the amount of information that the organization has to process in order to update or assess their compliance management initiatives.

In conclusion, this paper presents insights into the issues and challenges perceived by experts involved in managing compliance. In addition, we present a snapshot of IS research activity since 2001 and contrast it against the challenges identified by industry experts. The findings, and related discussion, are expected to be beneficial to the research community in particular as they communicate the opinion of industry experts that should be taken into consideration when undertaking research in the field, thereby resulting in research activity that has the potential to impact and contribute to practical problems faced by organizations.

One of the limitations of our work, beyond the geographic limitation to Australia, is the focus on experienced consultants. The consideration of views from various roles

in organisations will be a complement to the study in the future. Further we will undertake a review of contributions from computer science research (in particular research in the database community) as anecdotal evidence indicates that there have been substantial contributions with respect to e.g. solutions leading to automated monitoring and analysis of business /transactional data.

Further limitations of the work relate to the qualitative aspect of the study. Qualitative studies in particular can suffer from subjectivity in data analysis. In our study, through using multiple coding rounds, together with multiple coders, we have taken measures to ensure objectivity of the analysis.

References

1. Lu, R., Sadiq, S., Governatori, G.: Compliance Aware Business Process Design. In: ter Hofstede, A.H.M., Benatallah, B., Paik, H.-Y. (eds.) *BPM Workshops 2007*. LNCS, vol. 4928, pp. 120–131. Springer, Heidelberg (2008)
2. Anon, J.L., Filowitz, H., Kovatch, J.M.: Integrating Sarbanes-Oxley Controls into an Investment Firm Governance Framework. *The Journal of Investment Compliance* 8, 40–43 (2007)
3. Pershkov, B.I.: Sarbanes-Oxley: Investment Company Compliance. *The Journal of Investment Compliance* 3, 16–30 (2003)
4. Bace, J., Rozwell, C., Feiman, J., Kirwin, B.: *Understanding the Costs of Compliance*. Gartner Research. Gartner, Inc. (2006)
5. McGreevy, M.: *AMR Research Finds Spending on Governance, Risk Management, and Compliance Will Exceed \$32B in 2008*. AMR Research, Inc. (2008)
6. Reilly, K.: *AMR Research Finds Spending on Sarbanes-Oxley Compliance will Remain Steady at \$6.0B in 2007*. AMR Research (2007)
7. Robinson, K.T., Hawkins, R.W.: Investment Company and Investment Adviser Compliance Programs: New Requirements in a Changed Regulatory Environment. *The Journal of Investment Compliance* 4, 14–19 (2004)
8. Syed Abdullah, N., Indulska, M., Sadiq, S.: A Study of Compliance Management in Information Systems Research. In: *The 17th European Conference on Information Systems (ECIS 2009)*, Verona, Italy (2009)
9. Turner, R., Florio, C.D.: Investment Management Compliance: The Dawn of A New Era? *The Journal of Investment Compliance* 4 (2005)
10. Kramp, M.K.: Exploring Life and Experience through Narrative Inquiry. In: Marrais, K.d., Lapan, S.D. (eds.) *Foundations for Research: Methods in Education and the Social Sciences*, pp. 103–121. Erlbaum, Mahwah (2004)
11. Australian Competition & Consumer Commission: *Trade Practices Compliance Programs*. Commonwealth of Australia (2008)
12. Morton, J.C.: The Development of A Compliance Culture. *The Journal of Investment Compliance* 6, 59–66 (2005)
13. KPMG: *The Compliance Journey: Leveraging Information Technology to Reduce Costs and Improve Responsiveness*. KPMG International (2006)
14. Kharbili, M.E., Stein, S., Markovic, I., Pulvermüller, E.: Towards a Framework for Semantic Business Process Compliance Management. In: *GRCIS 2008*, Montpellier, France (2008)
15. SAI Global Research: *Risk and Compliance in Australia: The Issues and Trends as Seen by Practitioners* (2008)

16. Sadiq, S., Indulska, M.: Driving Compliance through BPM. The University of Queensland (2008)
17. Abrams, C., Känel, J.v., Müller, S., Pfitzmann, B., Ruschka-Taylor, S.: Optimized Enterprise Risk Management. *IBM Systems Journal* 46, 219–234 (2007)
18. Governatori, G., Milosevic, Z., Sadiq, S., Orłowska, M.: On Compliance of Business Processes with Business Contracts. ITEE Technical Report. The University of Queensland, Brisbane (2007)
19. Karagiannis, D., Mylopoulos, J., Schwab, M.: Business Process-Based Regulation Compliance: The Case of the Sarbanes-Oxley Act. In: 15th IEEE International Requirements Engineering Conference (RE 2007), pp. 315–321 (2007)
20. Liu, Y., Müller, S., Xu, K.: A Static Compliance-checking Framework for Business Process Models. *IBM Systems Journal* 46, 335–361 (2007)
21. Wilkins, R.: The Problems of Duplication and Inconsistency of Regulation in a Federal System. In: Grabosky, P., Braithwaite, J. (eds.) *Business Regulation and Australia's Future*. Australian Institute of Criminology, Canberra (1993)
22. Harmer, R.: Current Views on Compliance & Governance. Rob Harmer Consulting Services (2004)
23. Paul, S.: Demand for Governance, Risk and Compliance Products on The Rise. *The Hindu Business Line* (2008)
24. Caldwell, F., Eid, T.: Magic Quadrant for Enterprise Governance, Risk and Compliance Platforms. Gartner Research. Gartner, Inc. (2008)
25. Agrawal, R., Grandison, T., Johnson, C., Kiernan, J.: Enabling the 21st Century: Health Care Information Technology Revolution. *Communications of the ACM* 50, 35–42 (2007)
26. Sadiq, S., Governatori, G., Naimiri, K.: Modeling Control Objectives for Business Process Compliance. In: Alonso, G., Dadam, P., Rosemann, M. (eds.) *BPM 2007*. LNCS, vol. 4714, pp. 149–164. Springer, Heidelberg (2007)
27. Kim, H.M., Fox, M.S., Sengupta, A.: How to Build Enterprise Data Models to Achieve Compliance to Standards or Regulatory Requirements (and share data). *Journal of the AIS* 8, 105–128 (2007)
28. Davis, C.J., Hikmet, N.: Training as Regulation and Development: An Exploration of the Needs of Enterprise Systems Users. *Information & Management* 45, 341–348 (2008)

Experience-Based Approach for Adoption of Agile Practices in Software Development Projects

Iva Krasteva, Sylvia Ilieva, and Alexandar Dimov

Sofia University St.Kliment Ohridski, 65 Akad. J.Boucher str.,
Sofia, Bulgaria

iva.krasteva@rila.bg, sylvia@acad.bg, aldi@fmi.uni-sofia.bg

Abstract. The agile approach for software development has attracted a great deal of interest in both academic and industry communities in the last decade. Nevertheless the wide adoption of agile methods in ever growing number of software development projects, shifting the development process of an organization to an agile one is not straightforward. Certain considerations for the applicability of agile practices should be taken into account when this transition is performed. In this paper, an approach for situational engineering of agile methods is proposed. The approach is based on the experience gained in adopting agile practices in both internal and external projects of organizations. A knowledge-base supporting the selection of agile practices that are suitable for certain project is introduced. Automated generation of appropriate software development process is included as well. Particular realization of the approach supported by SPEM-based tools is also presented in the paper.

Keywords: agile practices, method engineering, project situation, context, SPEM.

1 Introduction

The agile approach for software development has attracted a great deal of interest in both academic and industry communities in the last decade. Based on common sense values and principles of collaboration, trust and the potential of the individuals, a number of agile practices have been proposed and a good number of agile methods have been developed. The agile software development has arisen as an alternative to the traditional development practice. A lot of success stories of its adoption in continuously growing number of domains and projects have been published. Nevertheless its wide adoption, shifting the development process of an organization to an agile one is not straightforward [1] [2] and certain considerations of the applicability of agile practices should be taken into account [3] [4]. In addition, the agile approach encourages adaptation and customization of the development method throughout the execution of the project, which makes the adoption process a continuous and interactive activity [5] [6] [7].

The objective of this research is to support organizations in introducing agile practices in their development methods and in further adoption and improvement

through the whole project. We propose a situational method engineering approach which is based on the experience gained in adopting agile practices in both internal and external projects. The suggested approach incorporates knowledge for agile practices applicability and suggests different mechanisms for knowledge acquisition. There are two major activities in the approach. The first one is *selection of agile practices* that are suitable for particular project. *Creation of a new agile development method* is the second one. Due to space limitations the paper presents the overall approach and describes in details the selection process, while the creation of the development method is discussed briefly.

The approach we propose supports iterative and incremental process for methodology creation and adaptability through a number of method engineering milestones during the execution of the development process. Our approach is built on an instance of a Software Process Engineering Meta-Model (SPEM) 2.0 meta-model [8] and its execution is supported by several tools implemented as extensions to Eclipse Process Framework (EPF) Composer [9]. Models that are used as input and output for the method engineering process are fully compatible with the standardized instance of SPEM 2.0 Meta-Model- the SPEM 2.0 Base Plug-in [8], and thus can be freely distributed among different SPEM-based tools.

The rest of the paper is organized as follow. Section 2 makes an overview of the related work and comparison between other approaches and this study is made. In Section 3 the method engineering process is presented as well as basic concepts and models used. Organization of the knowledge-base and the selection of applicable agile practices are described in details in Section 4. How the generation of new agile method is specified and realized is presented in Section 5. Section 6 proposes validation of the approach through comparison with an external source and a case study. Section 7 concludes the paper and makes suggestions for future work.

2 Related Work

Method engineering is an approach for conceptualization, construction and adaptation of methods and tools for information systems development [10]. Situational method engineering (SME) deals with the creation of project-specific method or tailoring existing ones to a particular project situation. Project situation is defined [11] as a combination of the (external) context of the project and the project type. We are adhering to this terminology throughout the whole paper.

A recent overview of the existing approaches for method engineering can be found in the works of Ralyte *et al.* [12] and Nehan *et al.* [13]. Depending on the method construction techniques a number of approaches can be distinguished. Assembly-based approach is based on the reuse of preexisting method components (in different approaches referred to as fragments [10] or chunks [12] with difference in the meaning). Components are selected to suit particular situation and are assembled using appropriate technique to form new development methods. The extension-based approach combines instantiation and extension techniques to form new methods by applying extension patterns. Another approach, the paradigm-based approach lies on the abstraction strategy by either abstracting from existing method or by instantiating a metamodel.

Two approaches which focus on method tailoring are developing recently. The practice-driven approach [14] enforces particular rules to configure company's development method. In the Method for Method Configuration (MMC) approach [15] particular process is customized by applying appropriate configuration patterns. As outlined in [14], different SME approaches differ in their ability to execute by providing sufficient details on how the approach actually works. In the presentation of our approach we describe in parallel its specification as well as its realization. Of course, due to space limitation the realization doesn't reveal all the details. Our approach is realized on SPEM 2.0 Meta-Model specification and the EPF Composer tool. By utilizing standardized and open source tools and specifications, we believe that the applicability and extendibility of the approach is greatly enhanced. In [16] [17] SPEM is used to represent agent oriented methodologies following the method engineering approach. They extend SPEM in appropriate way to support modeling of method fragments and agent design processes. Our approach, however, is different in the way the realization model is instantiated from SPEM 2.0 Meta-Model and the design of the development process.

The value of introducing a knowledge-base to SME approaches is recognized by others [18] [19]. Keeping track on how the method has been previously applied in different project situations gives the possibility to execute various analyses based on particular metrics. Klooster *et al.* [18] have introduced eighteen performance indicators to measure the success of a project which is determined by the dependencies among situation factors. Qumer *et al.* [19] argue on the importance of introducing a knowledge-base when creating agile methods and state that agile knowledge engineering and management approach should be integrated with an agile software development approach. Our knowledge-base is especially designed to support introduction of agile practices to a given project situation based on the experience of practices applicability from both outside and inside of an organization.

Comprehensive studies on the applicability of the agile approach by utilizing OPEN framework [22] have been published by Qumer *et al.* [19] [20] and Hendersson-Seller *et al.* [21]. In [19] a complete framework to assist managers in assessing the degree of agility they require and how to identify appropriate ways to introduce agility into their organization is presented. The Agile Software Solution Framework (ASSF) consists of agile conceptual aspect model and tools. The agile conceptual aspect model represents the aspects of knowledge, governance and method core; which are linked to business via a bridge that aligns business goals and agile software development goal. In addition to the framework, Agile Adoption and Improvement Model (AAIM) is suggested to support method adoption and further software process improvement efforts. Our approach shares some common points in the assessment of agile practices. However, while the ASSF focuses on assisting managers to decide on appropriate transition and the way to execute it, we are only focusing on method engineers whose objectives are to design and further tailor the method through its execution to best suit the people involved in the project and the particular project situation. It should support method engineers who are not experts in agile adoption to select appropriate practices and guide them in the creation of the new agile method.

3 Description of the Approach

The section makes an overview of the new experience-based situational method engineering approach suggested by us. First, the method engineering process is described. The concepts and models that are involved in the formal specification of the approach are presented in the second subsection.

3.1 The Method Engineering Process

The introduction of agile practices to the development methods of an organization should involve iterative and incremental process of adoption while continuously reviewing and customizing the practices to the particular project [6] [5] [19]. This process is based on two types of adaptation [7] - adaptation to the characteristics of the project and self-adaptation which is guided by the way the team responds to different development practices used in the project. For that reason our method engineering process is interleaved with the software development process and thus supporting the method engineer to adapt the process continuously either by manually tailoring it or by executing the method engineering process for any process iteration. The projects are described by a set of factors that characterizes particular project situation.

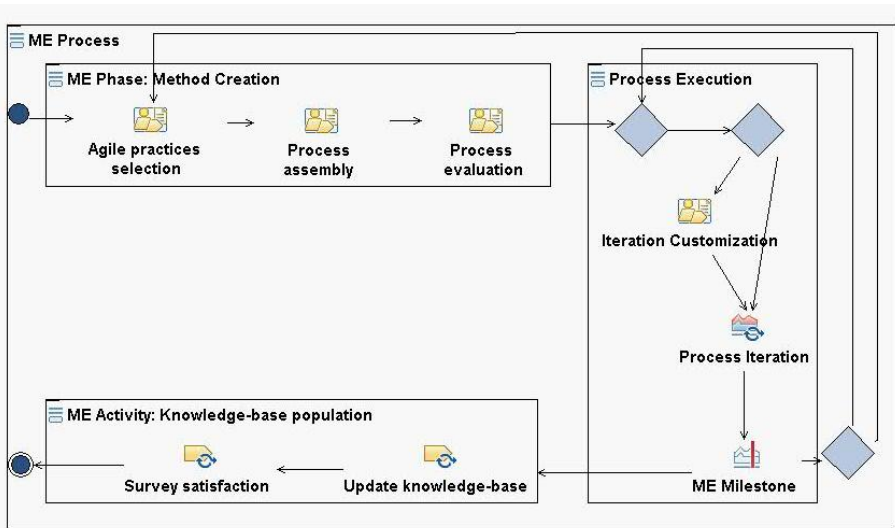


Fig. 1. The method engineering process

Three major parts constitutes the method engineering process proposed by our approach - method creation phase, process execution phase and knowledge-base population activity. The method creation phase begins with selection of agile practices that are applicable to the given project situation. The activity involves a number of steps for evaluation of practices applicability based on the information in a knowledge-base. The knowledge-base stores data for theoretical and empirical

adoption of agile practices when certain project and environmental factors are present. Appropriate practices are then assembled with the current development process in a number of possible processes which are further evaluated and one particular is chosen. The adoption is performed in a number of process iterations each of which ends with a method engineering milestone. If the iteration is not the last one, decides whether the next iteration should be changed or not. The method engineer can optionally customize some of the elements of the iteration or execute the method engineering process with a new set of practices. After the last iteration two additional method engineering tasks are performed- the information for the project is added to the knowledge-base and stakeholders' satisfaction is surveyed. The method engineering process is presented in Figure 1.

3.2 Concepts, Models and Tools

Our situational method engineering approach is based on SPEM 2.0, which is defined as a MOF 2.0-based Meta-Model as well as a UML 2 Superstructure-based Profile. SPEM is designed to provide the necessary concepts for modeling, documenting, presenting, managing, interchanging, and enacting development methods and processes [8]. In SPEM distinction between reusable method components (called Method Content) and instantiation of such components in particular processes is drawn. The method content provides the 'building blocks' such as tasks, work products, roles, tools, guidance, etc., while the process elements specify parts or the whole development process as a work breakdown structure (including activities as well as work products and roles) by implementation and further customization of the elements from the method content. In this sense a development method consists of different method content elements and process elements. Methods can be specified by reusing, extending and customizing method and process elements from other already defined methods by means of Method Configurations. The repository for methods is called Method Library. Figure 2 illustrates these concepts and their relationships.

Our method engineering approach is based on four metamodels. Two of them are standardized models- the SPEM 2.0 Meta-Model and the SPEM 2.0 Base Plug-in. We define the other two models- the Specification and the Realization metamodels. These models are derived from the SPEM metamodel by extension and/or instantiation.

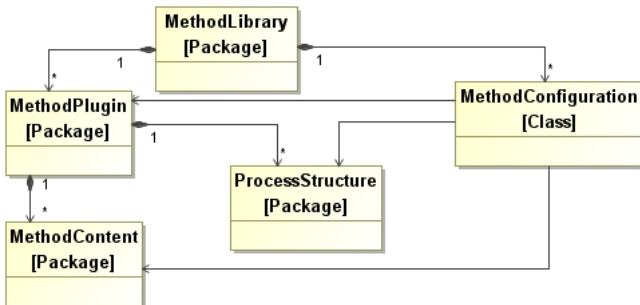


Fig. 2. Definition of SPEM 2.0 basic concepts

Most of method engineering concepts and reuse mechanisms can find their corresponding ones in SPEM elements. One such mapping assumed by us is presented in Table 1. The knowledge-base that we introduce in the approach is modeled using standard UML. However, it imports certain process and method content elements from our SPEM-based metamodels. The other way relation is not provided explicitly.

Table 1. Mapping of method engineering concepts to SPEM 2.0 Meta-Model elements

| Method Engineering concepts | SPEM 2.0 Meta-model elements |
|---|--|
| Method component | Method content; Process |
| Method repository; | Method library |
| Aggregation; Configuration; Instantiation | Variability types; extension mechanisms |

Specification metamodel presents the basic concepts of our approach and relationships between them. In order to execute our approach we have specified three tools that are realized as extensions to EPF Composer. EPF Composer utilizes the SPEM 2.0 Base Plug-in which is a standardized instantiation of SPEM 2.0 Meta-Model and provides commonly used instances for many SPEM concepts for the domain of Software Engineering [8]. For that reason we specify another model- the Realization metamodel, which imports the concepts form the Base Plug-in as well as certain design rules and validations to ensure that the model conforms to the Specification model. By introducing a second metamodel, the modeled elements that come as an input or output of our approach can be transferred to any SPEM-based tools for modeling software development processes. Figure 3 presents the relationships between different models used in our approach.

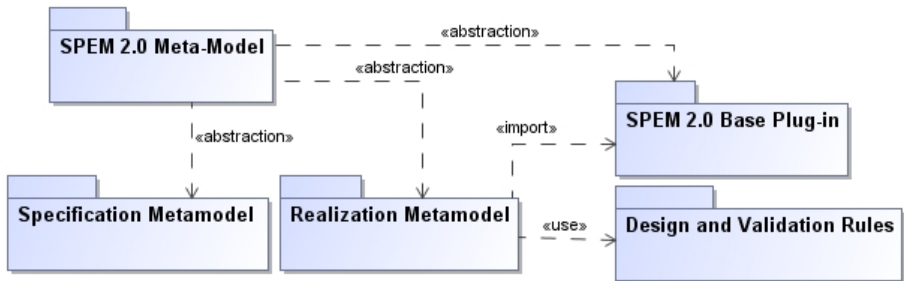


Fig. 3. Models introduced in the approach and their relationships

The three tools that are realized by us as extensions to EPF Composer are: the APA tool, the ConfCheck tool and the EBAGen tool. The Agile Practices Applicability (APA) tool supports all the operations with the knowledge-base of agile practices-feeding up data, editing data, configuring the situation parameters, analyzing the data. The tool also exposes a wizard-like interface to help the method engineer in the selection of agile practices appropriate for a particular project. The (Experience-Based

Approach method GENerator) EBAGen tool guides the generation of the new agile process from particular set of agile practices and a number of requirements specified by a method engineer. The Conformity Check (ConfCheck) tool is used mainly by the other tools to check the conformity of input method elements with the specification model.

4 Method Content Selection

As discussed in the introductory section the applicability of agile practices is heavily dependent on characteristics of the projects. Certain practice can be applied in one context while other cannot. For that reason in our approach we maintain a knowledge-base which stores information for applicability of particular agile practice when given project situation is present. The organization of the knowledge-base as well as different mechanisms that are used to analyze it and are used in the selection of appropriate agile practices for the project in hand are described in the subsections below.

4.1 Organization of the Knowledge-Base

Currently, there is no standard taxonomy of projects in the software industry that can be used to identify and categorize projects based on common factors. A lot of studies are published but there is no consistent and complete set of factors that are used to describe particular project situation. Jones [23] identifies 36 important context factors have been documented by Software Productivity Research (SPR) and used for benchmarking projects. However, this classification lacks some important factors that are relevant for agile development while introducing others that are not so important. Thus we use this project characterizing set of factors as a starting point and *map*, *adjust*, *add* and *remove* factors that are identified in related studies considering agile practices applicability [18] [20] [3] [24] [25]. As an outcome of we have identified 45 factors to be included as characteristics of different projects situations [26].

Once we have identified the relevant factors, the knowledge-base is designed to support the analysis of the applicability of agile practices according:

- Theoretical inapplicability
- Experience reports of other companies
- Experience of the company

The *theoretical inapplicability* analysis is based on the characteristics of a practice and makes assumptions about how appropriate is to apply certain practice when particular factor is present. We have defined 7-point scale $\{-5, -3, -1, 0, 1, 3, 5\}$ to measure the levels of applicability- from highly inapplicable (-5) to highly applicable (5). Zero means that applicability of certain practice is not affected by the presence of a factor. In previous works of ours [27] [28] we have studied theoretical applicability of agile practices when particular social, technological and business factors are present as well as their adoption in the development and usage of reusable components. In order to start populating the knowledge-base we are examining and synthesizing relevant theoretical research by us and others.

However deep and systematic those theoretical analysis are, the assumptions drawn by them can be validated through further *empirical studies*. The knowledge-base stores information of projects in which agile practices have been applied. Relevant data is taken from published experienced reports on agile adoption through systematic review of literature. The projects are described by means of the set of situational factors we have identified, and practice applicability is recorded in a 5-point scale {N/A, 0, 1, 3, 5} showing the extent to which the practice is used ('N/A' stands for 'Unknown'). The evidence of the success or failure of practice usage is also recorded in the knowledge-base. The information in the knowledge-base can be further analyzed and conclusions based on the experiences of other companies of the applicability of agile practice when certain factor is present can be made. We are currently researching appropriate statistical methods to analyze the data. Although based on empirical reports, there are a couple of issues that should be considered when this type of analysis is made. Such issues consider the high level of uncertainty and subjectivity of the information, as well as the quality of the reports. The presence of certain situational factors might not be explicitly revealed in the report however they can influence applicability of given practice in the real situation. Mapping project descriptions to situational factors, deciding the level of applicability and the evidence of success or failure involves speculative reasoning and its effect on the results should be taken into account. The last consideration is about the quality of the empirical reports. In a systematic review [29] of empirical studies up to and including 2005 year, out of 270 studies only 36 of them satisfied certain quality criteria. In order to mitigate the impact of such issues the statistical significance and quality of the data in the sample is examined as part of the statistical analysis. Furthermore, as an additional step a statistical analysis on a set of situational factors to a set of practices is introduced. In this way influences among practices are also taken into consideration and sets of practices are suggested for application. For the purpose of comparing how close one project situation is to the other, we introduce the measure of situation proximity. Situation proximity of two situations is measured as a sum of:

- '1' for each factor that are the same in both situations
- '0' for each factor that is different
- when a factor is not known, the value of the probability this factor to match the one in the other situation (this is dependent on the cardinality of the scale for particular factor).

An additional feature of the knowledge-base is the information about incompatible agile practices, which is based on theoretical research of the characteristics of agile practices like the one made by us [27]. Such information is useful when the set of practices is evaluated.

The knowledge-base contains also information for applicability of agile practices in the *past projects of the organization*. This is the most reliable information however not available for first-time agile adoption. The information from internal projects contains complete project situations and thus the uncertainty level is not an issue. The records in this part of the knowledge-base can contain customized practices as well as general ones. Also for this type of information we can collect additional information on results of practice application. In [18] eighteen performance indicators are suggested to measure the success of particular practice applicability. We would like to

keep our information repository light and measure the success of a given practice by the satisfaction of project stakeholders.

4.2 The Selection Process

The knowledge acquisition and analysis is supported by the APA tool. The selection of agile practice to be used in the project is iterative procedure and is based on a number of parallel analysis steps. A *precondition* to the selection procedure is the method content that contains the agile practices we want to be analyzed. As a *first optional step* of the selection procedure the state of the knowledge-base is set – it could be either fed up with data, or the data can be updated, or just reviewed. This data, however, is not stored in some dedicated repository. Once the information about theoretical and empirical applicability of practices is specified by the means of the tool, it is serialized as text (according predefined by us format) and attached to a dedicated text attribute in each agile practice. In such a way the information about agile practices applicability is attached to them and can be read by appropriate parser tool. On a *second step* the method engineer specifies the project by the set of situational factors.

The third step includes several parallel or sequential analyses:

- Theoretical analysis per practice per situational factor
- Empirical analysis per practice per situational factor
- Empirical analysis per project situation for external projects (according to situation proximity)
- Empirical analysis per project situation for internal projects (according to situation proximity)

The method engineer selects several sets of practices which are checked for incompatibilities. The data for theoretical incompatibility of two practices is stored in the knowledge-base. Some of the practices are substituted with custom practices from the internal records. The method engineer can decide to run some of the analyses again and change the preliminary sets of practices. At the end, one set is selected to be used in creating the development method. The output of the selection procedure is a new method configuration in EPF Composer that contains selected agile practices. The description of the project situation is attached to an appropriate text attribute in the new development process, which is to be populated in the next step.

5 Development Process Creation

Once the appropriate set of agile practices is selected they can be adopted with the existing practice of the organization in a particular development process. How the new development process is generated is presented in the current section. We first present the specification metamodel.

5.1 The Specification Metamodel

The metamodel we define specifies the basic concepts and the relationships between them to support our approach. The basic concepts that are introduced in the

metamodel are presented in Figure 4. We introduce AgilePractices and CustomAgilePractices elements as SPEM Guidance. Any custom agile practice customizes one of the general agile practices. AgileValues are SPEM Metrics that present agile values such as collaboration and simplicity [30] [7] [19] associated with each agile practice. Agile practices are categorized further depending on their purpose in the development method- either they are part of the development or project management activities in the software process, or they support the collaboration [7]. We introduce three breakdown elements to support the method engineering process- the MEPhase and MEActivity as SPEM Activities and MEMilestone as SPEM Milestone.

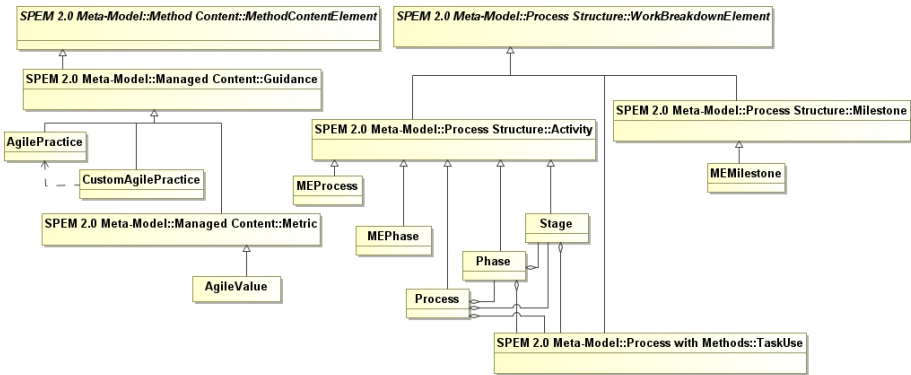


Fig. 4. Basic concepts in the experienced-based approach- specification metamodel

We use the process patterns as described in [31] to present and build our agile development process, which are modeled with Process, Phase, Stage and Task Description elements. The Task Description element is reused as-is from SPEM specification. A particular pattern of higher level can only contain patterns of the lower levels. An association of process pattern with AgilePractice guidance specifies an agile development process pattern. Each pattern of a higher level is associated with one input and one output Work Product Description. The Task Descriptions are associated with at least one Role Description and one Work Product Use. The Task Descriptions, Role Descriptions and Work Product Uses that are used in the process are instances of elements in the Method Content. Qualifications of Roles are specified with predefined enumeration of skills and competences, which is serialized in a text attribute of a Describable Element. In this way they are compatible with the model but are reused in our knowledge-base to model persons as part of the project situation. Work Product Definitions are categorized as Formal Work Product or Informal Work Products.

We also introduce the notion of Abstract Patterns and abstract Work Products. An abstract pattern is an activity of higher level- Process, Phase or Stage, in whose breakdown structure there is a leaf element which is also abstract. An abstract work product is a more general work product which needs to be instantiated in the process. Such general relationship among Work Products Definitions is expressed by A Work Product Definition Relationship in the SPEM Meta-Model.

5.2 Generation of Development Method

As discussed earlier, in order to provide compatibility of our approach with the SPEM 2.0 Base Plug-in implemented in EPF Composer, a number of design and validation rules are introduced. One such design rule is the use of specific custom categories instead of new stereotypes. The ConfCheck tool is used to check whether method and process elements that are inputs to the generation procedure conform to the specified design and validation rules. The EBAGen tool guides the method engineer in the generation of a new development process by introducing certain set of agile practices to the current development process. Two types of inputs are expected- the method and relevant process content for a set of agile practices and for the development method used in the organization. In addition, the method engineer should specify abstract patterns for the current development method as well as a number of disciplines that are domain specific and utilizes some of those patterns. The particular instance of the current development method should provide instantiation for these abstract patterns. Process generation algorithm uses a number of instantiations and assembly mechanisms to create new processes.

In the beginning, the method engineer specifies the number of method engineering phases and milestones. Then for each of them an abstract pattern to be instantiated is specified. The abstract patterns are available from the current method as well as from agile practices. Abstract patterns can be specified by the method engineer and can be included in the configuration to be used in the method engineering process. At the next step, the method engineer specifies abstract work products. If any tasks or work products should be preserved in the new method they are specified as mandatory. The method engineer also defines the roles to be used in the process by either selecting them from the predefined roles based on the available skills or by specifying new roles. Based on the skill set during process generation new roles are assigned to the tasks. Different variants of non-abstract methods are generated after selecting those roles, work products, tasks and lower level abstract patterns that can instantiate abstract patterns and their abstract work products. The method engineer evaluates the generated processes according:

- abstraction level- depends on the extent to which abstract patterns are instantiated. More abstract processes are liable to customization [7]
- business values of the included agile practices
- coverage of the disciplines of the domain
- lightness of the process from the involved types of work products

The method engineer can select one process or start the generation from the beginning with different initial parameters. When the method engineer chooses one particular process a new method configuration is created and project characteristics are copied to the delivery process. From any method engineering milestone the following ME phases can be regenerated starting from the initial configuration. After the regeneration the method configuration is updated accordingly.

6 Validation

We have designed and implemented our approach having in mind the methodology design principles and concepts identified by Cockburn [6].Cockburn mention four things that should be taken into account when designing a methodology:

- Variations in people
- Variations across projects
- Long debug cycles
- Changing technologies, techniques, and cultures

With the exclusion of the long debug cycles, the other three considerations are addressed in our approach by identification of project situation and people as part of it, and by supporting adaptation of the methodology throughout the project execution. Most of the methodology elements identified by Cockburn are either explicitly included in our approach or can be easily derived. Table 2 presents how the methodology elements are supported in our approach.

Table 2. Methodology elements and their support by the experience-based approach

| Methodology Elements [Cockburn] | Supported by: |
|---------------------------------|---|
| Process | Metamodel element Process |
| Milestone | Metamodel element Milestone |
| Activity | Metamodel element Stage |
| Technique | Metamodel element Task with associated Guidance |
| Skills | Metamodel element Skills and knowledge-base element |
| Role | Metamodel element Role |
| Team | Metamodel element Team profile |
| Tool | Metamodel element Tool |
| Standards | Work products and tools that are precondition for the creation of the development process |
| Product | Metamodel Element Work Product Description, Work Product Use |
| People | Knowledge-base- situation factors |
| Quality | Knowledge-base – satisfaction |

Currently we are in a process of verifying our approach in a case study of the applicability of agile practices in the domain of component-based development. The case study we are currently completing examines the adoption of agile development in the implementation of both components and systems based on components. We have previously studied the theoretical adoption [28] by systematic analysis of the applicability of the agile approach in the development processes of components and systems. As a second step, we conducted an empirical research [32] in which the assumptions made in the first study were verified by an industry survey. As we had expected, some of the assumptions was confirmed by the answers of the professional, while others were not. Most of the situational factors were included in the structure of the survey and the data is suitable to feed up the knowledge-base with a number of relevant experience reports. We have received 43 complete answers, from which 33

are related to development of components and component-based systems. The structure of the questionnaire as well as the data is available as technical report [33]. The preliminary analysis of the results has shown that most of the agile practices are applicable in projects for implementation of components and systems based on components. In addition, the practitioners have expressed strong preference towards using the agile practices more rigorously.

7 Conclusion and Future Work

The paper presented an experience-based approach for situational engineering of agile methods. The approach supports systematic adoption of agile practices to the development method of organizations by analyzing experience reports on applicability of agile practices when particular situation factors are present. It introduces iterative and incremental method engineering process, adaptable to project characteristics and customizable through the execution of the project. Its realization is supported by a number of tools which are specified as extensions to EPF Composer tool. Method and process elements, used as input and produces as an output of the method engineering process, are fully compatible with the standardized SPEM 2.0 Base Plug-in and can be further managed by any other SPEM-based tool.

We are currently implementing the tools by the specifications, described briefly in the paper. As well, we are developing appropriate criteria for systematic review of available empirical and theoretical studies on agile practices adoption, which is to be used for knowledge-base initialization. In future, we want to add more formal presentation and evaluation of methodology conceptual terms such as ceremony, and weight. Introduction of simulations for evaluation of generated development processes is another extension to our approach planned for the near future.

Acknowledgments. The work is funded by Bulgarian Ministry of Education and Science, National Science Fund. The authors want to thank the anonymous reviewers for their valuable feedback.

References

1. Hodgetts, P.: Refactoring the development process: experiences with the incremental adoption of agile practices. In: Agile Development Conference, Salt Lake City, pp. 106–113 (2004)
2. Krasteva, I., Ilieva, S.: Adopting an Agile Methodology - Why It Didn't Work. In: ICSE: International workshop on Scrutinizing agile practices or shoot-out at the agile corral, pp. 33–36 (2008)
3. Kruchten, P.: Keynote: Situated Agility. In: 9th International Conference on Agile Processes and eXtreme Programming in Software Engineering (2008)
4. Koch, A.: Agile Software Development Evaluating the Methods for Your Organization. Artech House Publishers (2004)
5. Beck, K.: Extreme Programming Explained: Embrace Change, 2nd edn. Addison-Wesley Professional, Reading (2004)

6. Cockburn, A.: *Agile Software Development: The Cooperative Game*, 2nd edn. Addison-Wesley Professional, Reading (2006)
7. Highsmith, J.: *Agile Software Development Ecosystems*. Addison-Wesley Professional, Reading (2002)
8. Software process engineering metamodel. Version 2.0. formal/2008-04-01, OMG (2008)
9. EPF Composer. In: *Eclipse Process Framework*, <http://www.eclipse.org/epf/>
10. Brinkkemper, S.: Method engineering: engineering of information systems development. *Information and Software Technology* 38(7), 275–280 (1996)
11. Butcher, T., Klesse, M., Kurpjuweit, S., Winter, R.: Situational Method Engineering on the Differentiation of “Context” and “Project Type”. In: Ralyte, J., Brinkkemper, S., Henderson-Sellers, B. (eds.) *IFIP International Federation for Information Processing, Situational Method Engineering: Fundamentals and Experiences*, vol. 244, pp. 33–48 (2007)
12. Ralyté, J., Deneckère, R., Rolland, C.: Towards a Generic Model for Situational Method Engineering. In: Eder, J., Missikoff, M. (eds.) *CAiSE 2003*. LNCS, vol. 2681, pp. 95–110. Springer, Heidelberg (2003)
13. Nehan, Y.-R., Deneckere, R.: Component-based Situational Methods A framework for understanding SME. In: *IFIP International Federation for Information Processing, Situational Method Engineering: Fundamentals and Experiences*, vol. 244, pp. 161–175 (2007)
14. Bajek, M., Vavpotic, D., Krisper, M.: Practice-driven approach for creating project-specific software development methods. *Information and Software Technology* (49), 345–365 (2007)
15. Karlsson, F., Agerfalk, P.: Towards structured flexibility in information systems development: Devising a method for method configuration. *Journal of Database Management* 20(3), 51–75 (2009)
16. Seidita, V., Cossentino, M., Gaglio, S.: Using and Extending the SPEM Specifications to Represent Agent Oriented Methodologies. In: Luck, M., Gomez-Sanz, J.J. (eds.) *AOSE 2008*. LNCS, vol. 5386, pp. 46–59. Springer, Heidelberg (2009)
17. Cossentino, M., Gaglio, S., Henderson-Sellers, B., Seidita, V.: A Metamodelling-based Approach for Method Fragment Comparison. In: *International Workshop on Exploring Modeling Methods in Systems Analysis and Design at CAiSE 2006*, pp. 419–432 (2006)
18. Klooster, M., Brinkkemper, S., Harmsen, F., Wijers, G.: Intranet facilitated knowledge management: a theory and tool for defining situational methods. In: *Conference on Advanced Information Systems Engineering, Barcelona, Spain*, pp. 303–317 (1997)
19. Qumer, A., Henderson-Sellers, B.: A framework to support the evaluation, adoption and improvement of agile methods in practice. *The Journal of Systems and Software* (81), 1899–1919 (2008)
20. Firesmith, D.G., Henderson-Sellers, B.: *The OPEN Process Framework*. Pearson Education, London (2002)
21. Qumer, A., Henderson-Sellers, B.: Agile software solution framework: An analysis of practitioners’ perspectives. In: *Information Systems: Modeling Development and Integration: Third International United Information Systems Conference, UNISCON 2009*, pp. 41–52 (2009)
22. Henderson-Sellers, B., Serour, M.K.: Creating a Dual-Agility Method: The Value of Method Engineering. *Journal of Database Management* 4(14), 1–16 (2005)
23. Jones, C.: *Software Assessments Benchmarks and Best Practices*. Addison-Wesley Professional, Reading (2000)

24. Turk, D., France, R., Rumpe, B.: Assumptions underlying agile software-development processes. *Journal of Database Management* 16(4), 62–87 (2005)
25. Cockburn, A., Highsmith, J.: Agile software development, the people factor. *Computer* 34(11), 131–133 (2001)
26. Krasteva, I., Ilieva, S.: Characterizing Agile Projects. Internal Report, Sofia University 'St. Kliment Ohridski', Sofia (2009)
27. Krasteva, I., Ilieva, S.: Rush into Agile- Analytical Framework for Agile Practices Applicability. In: *IET Conference Publications* (528 CP), Durham, pp. 229–237 (2007)
28. Krasteva, I., Branger, P., Land, R.: Challenges for agile development of COTS components and COTS-based systems A theoretical examination. In: *International Conference on Evaluation of Novel Approaches to Software Engineering*, Funchal, Madeira, pp. 99–106 (2008)
29. Dyba, T., Dingsyr, T.: Empirical studies of agile software development: A systematic review. *Information and Software Technology* 50(9-10), 833–859 (2008)
30. Manifesto for Agile Software Development, <http://agilemanifesto.org/>
31. Tasharofi, S., Ramsin, R.: Process Patterns for Agile Methodologies. In: *IFIP International Federation for Information Processing, Situational Method Engineering: Fundamentals and Experiences*, vol. 244, pp. 222–237 (2007)
32. Krasteva, I., Land, R., Sajeev, A.S.M.: Being Agile when Developing Software Components and Component-Based Systems- Experience from Industry. In: *EuroSPI Conference*, vol. *Industrial Proceedings*, pp. 8.7-8.17 (2009)
33. Causevic, A., Krasteva, I., Land, R., Sajeev, A.S.M., Sundmark, D.: An Industrial Survey on Software Process Practices, Preferences and Methods. Malardalen University, Sweden (2009)

Coordinating Global Virtual Teams: Building Theory from a Case Study of Software Development

Gaye Kiely¹, Tom Butler¹, and Patrick Finnegan²

¹ University College Cork, Ireland
gaye.kiely@ucc.ie, tbutler@afis.ucc.ie

² University of New South Wales, Sydney, Australia
p.finnegan@unsw.edu.au

Abstract. Global Virtual Teams (GVTs) enable organizations to operate across national, economic and social, and cultural boundaries. However, this form of teamwork presents issues for traditional project management coordination mechanisms. There is a significant body of research on these challenges. However, relatively little attention has been paid to the specific impact these issues may have on coordination mechanisms in GVTs. This paper seeks to address this gap by applying a theoretical model drawn from extant research to explore the coordination mechanisms used by a software development GVT in a Fortune 100 telecommunications manufacturer. The study employs a mixed methodology grounded theory approach to examine the effect that specific virtual team issues have on the effectiveness of team coordination mechanisms. It then develops a refined conceptual model to guide future research on GVTs involved in software development. The findings also inform practice on the problems encountered in ensuring the effective coordination of such teams.

Keywords: Global Virtual Teams, Coordination Mechanisms, Software Development, Project Management.

1 Introduction

The use of global virtual teams (GVTs) in software development (SD) has become standard practice for many organizations [1]. GVTs are a specific type of virtual team which are, typically, geographically, temporally and organizationally dispersed; they also tend to be culturally diverse in their constitution [2, 3, 4]. There has been extensive research into some aspects of virtual teams [37, 16]. However, there is relatively little empirical research on GVTs and associated team structures [16, 36] such as coordination mechanisms. Existing studies on GVTs and coordination mechanisms, for the most part, employ small, student teams and little focus has been given to long-term virtual teams [16] which limits the applicability to SD GVTs in practice. In addition, it is clear from extant research that the increased use of GVTs [5, 6, 7, 8, 9, 10, 11, 3] and the development of increasingly complex software artefacts have combined to bring new challenges to the SD environment [1, 5, 38]. In order to shed light on virtual team structures there is a clear need for new theory development. The

objective of this research study is to investigate the use of traditional coordination mechanisms in GVT SD projects and apply theory to explain the impact that specific GVT issues have on the effectiveness of such mechanisms. To facilitate this objective, a theoretical model and associated propositions will be developed from extant research findings. The resultant model will be used as a theoretical lens through which to study the phenomena and derive a set of hypotheses.

2 Towards a Theoretical Model for Studying the Coordination of Virtual Teams

2.1 Coordination Mechanisms

This study employ's Sabherwal's [13] classification of coordination mechanisms as a foundation for a conceptualist theoretical model. While Sabherwal's study does concern itself with coordination mechanisms, its primary focus is on the relationship between client and vendor in outsourced SD projects and how coordination mechanisms evolve in the course of the relationship. Nevertheless, we argue that Sabherwal's synthesis of existing research on coordination in SD provides a useful foundation for theory building on GVTs. Four constructs are posited by Sabherwal as key mechanisms to coordinate teamwork in SD teams [14, 15]: these are (i) coordination by standards, (ii) coordination by plans, (iii) coordination by formal mutual adjustment and (iv) coordination by informal mutual adjustment. Coordination by standards refers to those mechanisms which are used to direct team members to uniform practice such as methodologies, codes of practice etc. Coordination by plans refers to any documentation which may be employed to coordinate and direct team members (schedules, project plans etc.). Coordination by formal mutual adjustment (FMA) are those mechanisms which require team members to interact in a pre-defined manner such as project meetings. In contrast, coordination by informal mutual adjustment (IMA) involves team members interacting in an informal manner through ad-hoc meetings, impromptu communications, or co-location.

2.2 Factors That Affect Global Virtual Teamwork

Several factors or issues have been identified as influencing project outcomes in GVTs viz. distance [16], time zones [7], leadership [9], language [5], knowledge sharing [11], culture [3] and trust [8]. Based on our extensive analysis of extant research, we selected five of these issues to study their impact on coordination mechanisms in a GVT. There is significant support in existing literature to suggest that Distance, Time Zones, Language, Culture and Trust impact the coordination of a GVT as well as influencing project outcomes [5, 16, 40]. While there are a number of other issues (such as leadership and knowledge sharing) that influence project outcomes they were excluded from the study, in order to focus on issues which have been identified as having a negative impact on coordination of GVTs. Hence, we propose the following constructs:

- Distance: Defined as the physical separation of team members across geographically dispersed project sites [17].

- Time zones: Defined as the time difference(s) between the project sites [18].
- Language: Conceptualized as the difficulties arising when the GVT’s working language is not the native language team members across all project sites [8].
- Culture: GVT coordination will be affected by the fact that team members may possess diverse ethnic, national, and organizational backgrounds [19].
- Trust: Defined as the willingness to be vulnerable to the actions of another party based on the expectation that the other will perform a particular action important to the trusting party, irrespective of the ability to monitor or control that trusted party [20]

Though it is widely assumed that these constructs influence project outcomes through their effects on project team coordination mechanisms, there is a paucity of supporting evidence in the literature. In the next four sections, we draw on extant research on GVTs and project team coordination to theorize on the relationships between the proposed constructs and the effectiveness of different types of coordination mechanisms in GVTs (see Figure 1). Several theoretical propositions are then offered; these describe and explain the posited relationships.

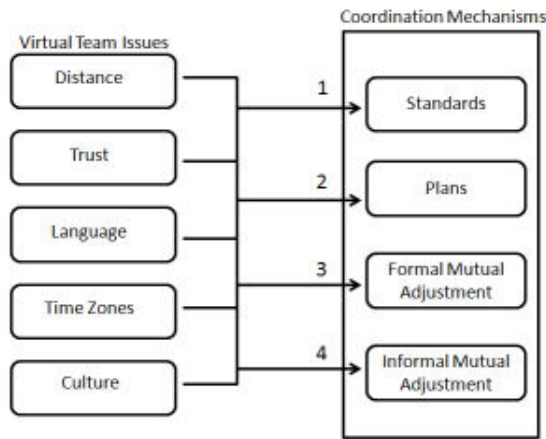


Fig. 1. Theoretical Propositions

The constructs of distance, time zones, language, trust and culture are generally viewed as having a negative impact on a virtual team’s ability to successfully interact and complete a SD project. The issues that arise from these factors create obstacles for the coordination of teamwork in particular [21]. There is general agreement that team coordination is a key activity in producing successful outcomes for project teams [22, 23].

As indicated, Standards, Plans, FMA and IMA are categories of coordination mechanism employed in SD teams [13]. Standards are typically established at the outset of a project, communicated to the SD team and, because they are typically determined at corporate level, they tend to be altered only in exceptional circumstances.

Plans are typically project specific and are formulated before the commencement of a project and communicated to the entire virtual team; because they are project-based, they may be subject to ongoing modification throughout the life cycle of a project. Unlike Standards and Plans, both FMA and IMA depend on a high degree of team member interaction and communication to be effective as a coordination mechanism. Scheduled weekly meetings, status calls and shared calendars are examples of FMA while IMA is exemplified by casual face-to-face meetings, ad-hoc phone calls or emails. Existing literature suggests that these four types of coordination mechanism are not immune from virtual team issues and experience a negative impact in terms of their effectiveness (Table 1).

Table 1. GVT issues on Coordination Mechanisms (from extant literature)

| | Distance | Trust | Time Zones | Language | Culture |
|----------------------------------|----------|-------|------------|----------|---------|
| Standards | 7, 10 | 24 | 16 | 25 | 21, 24 |
| Plans | 7, 16 | 2, 5 | 7, 41 | 6, 10 | 3, 24 |
| Formal Mutual Adjustment (FMA) | 26 | 2 | 5, 24, 41 | 5, 41 | 3, 41 |
| Informal Mutual Adjustment (IMA) | 16, 26 | 2, 5 | 21 | 5, 8 | 3, 27 |

However, there is a paucity of research which specifically looks at the impact of GVT issues on specific coordination mechanisms. While a number of studies point to the issue of coordination in GVTs [42], few studies focus specifically on coordination itself or its associated mechanisms. In addition, existing literature on GVTs is limited in its treatment. For example, while there are a number of GVT-based studies which focus on SD, the majority are concerned with short-term projects [2, 5, 10, 24, 28], operating over a relatively small number of geographical locations [2, 5, 7, 24]. Few studies look at GVT projects that span longer time periods, which are argued to be the norm for such initiatives [3]. In addition, a number of existing studies into GVTs employ simulated virtual team projects using university students [21, 24]. While such studies may contribute to theory building, we argue that their findings are not generalisable to industry contexts and thus may be irrelevant to practitioners. There is, therefore, a need to look specifically at industry GVTs (with relatively large teams, dispersed over multiple geographical locations) involved in long-term SD projects of 12 months or more and to identify the specific impact of GVT issues on coordination mechanisms. Given the foregoing observations, we posit the following propositions:

Proposition 1: The effectiveness of Standards is impacted by distance, trust, language, time zones and culture.

Proposition 2: The effectiveness of Plans is impacted by distance, trust, language, time zones and culture.

Proposition 3: The effectiveness of Formal Mutual Adjustment is impacted by distance, trust, language, time zones and culture

Proposition 4: The effectiveness of Informal Mutual Adjustment is impacted by distance, trust, language, time zones and culture

The independent and dependant constructs described above are presented in the conceptual model presented in Figure 1: the relationships between these constructs are presented as the above propositions.

3 Research Method

The objective of this research study is to investigate the use of traditional coordination mechanisms in virtual SD projects and to explore the impact that GVT issues such as distance, trust, language, culture and time zones have their effectiveness. As an initial step in theory building, a conceptual model (based on extant research) with associated propositions was presented in the previous section. Following the specification of the propositions, the next step in theory building is to determine empirical indicators and subsequently produce hypotheses for empirical testing [29]. We now describe the research design on which this study is based. A 'single' case study research approach was selected as the most appropriate vehicle for theory building and refinement [30, 31, 32]. Case studies are particularly well suited to IS research [33]. In addition, case research, which emphasizes the understanding of empirical data in natural settings, is a suitable method for studying IS issues and practices [31]. A single case study is viewed as being a potentially rich and valid source of data and is a particularly good fit for the purpose of exploring relationships between variables in context [32].

The case organization is a Fortune 100 telecommunications company with approximately 66,000 staff worldwide. The organization was purposively chosen for study on the basis of its predominant use of GVTs for software development. Out of several possible ongoing software development projects, one specific project was selected for study. The specific software development project was also purposefully selected, as it met several important criteria. That is it possessed a GVT that operated over, several geographically dispersed project locations with multiple time zones—also, the team was culturally diverse. The project focused on the SD of a network operations and maintenance interface for 3G technology. The project had a planned 18 month duration with team members located in six geographical locations—the USA, Ireland India, Israel, Malaysia and China. There were upwards of 48 team members working on the software project in different locations at any point in time. The development team was split into a number of distinct development functions; requirements engineering, systems engineering, software test, development, customer support, deployment, quality control, project management etc. Team sub-units in different geographical locations had distinct responsibilities viz. Israel (development), USA (development), Malaysia (development/test), China (development/test), India (development) and Ireland (project management/development). Data collection was conducted over a twelve-month period and concluded towards the end of the project development cycle during the software testing and implementation phases. Ten interviewees were purposively selected using the key informant approach in order to gather data from a wide variety of team members. Thus, respondents were selected from all project sites (excluding Malaysia) and those participating in the study performed a range of project roles within the team (project manager, development manager, software developer, system engineer, test team leader, software tester). An interview guide was used to structure interviews [34]. Follow-up interviews, emails

and phone calls were used to clarify and refine issues which emerged from primary interview transcripts. The study utilised concepts from a priori theory in formulating the interview guide and in specifying 'seed categories' to inform the content analysis [35]. Opening coding and axial coding were used to determine hypotheses about the relationships between the categories identified. This process continued in an iterative manner, and resulted in the modification of categories and relationships. Selective Coding was undertaken to identify the relationships between categories (constructs) using hypothesized conditions, context, strategies and consequences. A context rating scale [39] combined with frequency counts were used to attribute a high, moderate or low impact to identified hypotheses. Discriminate sampling [35] was used to select data to examine strong and weak connections between categories. The issues of validity, reliability, and objectivity [32] were addressed through: (a) Prolonged engagement and persistent observation; (b) triangulation techniques, which were extensively used to provide insights into events, relationships, etc. between data sets; (c) a data analysis approach based on rigorous coding and the use of memos, which together provided an audit trail; and, finally, (d) member checks were also employed as the findings were presented to team members for subsequent feedback [34].

4 Case Study Analysis

This section presents the findings of the case study. From the analysis of the data we were able to (i) identify empirical indicators for virtual team issues (Table 1); (ii) identify empirical indicators for coordination mechanisms (Table 2) and (iii) illustrate how the effectiveness of coordination mechanisms is impacted by virtual team issues (distance, time zones, language, culture and trust). All coordination mechanisms identified in the case organization were mapped to categories highlighted by existing research; hence, we find support for our conceptual model (see Table 2).

4.1 Coordination Mechanisms Employed within the Case Organisation

A range of coordination mechanisms were used by the virtual team. Exemplars of Standards employed as coordination mechanisms were: an explicitly defined software development lifecycle; an extensive repository of document templates; and code inspections. Exemplars of Plans were: a project plan that included a software configuration plan (SCM); a quality control plan for the project, and a project Gantt chart. Exemplars of FMA included weekly, pre-scheduled conference calls; local weekly site meetings and use of mail aliases and shared calendars. Exemplars of IMA included ad-hoc phone calls between team members; unscheduled emails between team members and ad-hoc conversations between co-located team members.

4.2 The Impact of Virtual Team Issues on the Effectiveness of Standards

The conceptual model (Figure 1) proposes that the effectiveness of Standards is impacted by GVT issues which include Distance, Time Zones, Language, Culture and Trust. However, the findings indicate that some virtual team issues have a greater impact than others on specific coordination mechanisms. The analysis of case study data revealed that Standards are impacted by language, physical distance, and team

member trust but time zones and cultural differences appear to have no perceived impact on Standards. Language had the greatest negative impact on the use of project documentation such as bug reports and templates. A project manager explained that in a number of instances documents had to be reviewed and they had to work out what the non-native language speaking member meant before rewriting the document for general consumption. Team members also reported the creation of additional documentation which acted as guidelines for members across different project sites. The test team leader created a set of documents for his team which detailed how to write an email, ask a question etc. He described this as “documents to show how to use documents”.

In relation to Distance, the moderately negative impact is observed in the instance of project sites diverging in terms of development processes and forming distinct methodologies. A development manager was particularly concerned about this divergence. He explained “when you have teams in different geographic regions you may have different kinds of standards and therefore some non-uniformity in terms of the quality”. In relation to project documentation and Standards, team members expressed high levels of distrust with members at other sites. When documentation was not updated in a timely fashion, it decreased the confidence the members had in the documentation and thus decreased its usefulness as a coordination mechanism. However, case study analysis found no support for Culture and Time Zones having an impact on the effectiveness of Standards. We can therefore continue the theory building process by specifying three hypotheses:

H1: Distance has a moderately negative impact on the effectiveness of Standards

H2: Trust has a highly negative impact on the effectiveness of Standards

H3: Language has a highly negative impact on the effectiveness of Standards

4.3 The Impact of Virtual Team Issues on the Effectiveness of Plans

The case study data revealed that Plans are greatly impacted by Distance, Time Zones, Culture and team member Trust. The data analysis revealed, however, that Language appeared to have no impact on Plans. The impact of Distance on Plans was evident in the need of team members at different project locations to maintain local project schedules as well as feeding into one global project schedule. In addition, project estimates were required from each project location. This meant that the project team spent a lot of time “going around in circles”. Culture was perceived as having a highly negative impact on the effectiveness of Plans as a coordination mechanism with some team members observing that those with different cultural backgrounds in other sites tended to avoid saying ‘no’ to work requests. This behaviour had serious ramifications for the project schedule and completion, due to inadvertent over-commitment through an inability to say ‘no’. A project manager provided an example of members in another location agreeing to a task even though they knew from the outset it was not feasible with the given schedule. The project manager perceived this as being a “cultural issue” as members in the other project sites would “generally say no to work which was not feasible straightaway”.

Table 2. Indicators for GVT Issues (Language, Trust, Distance, Time Zones and Culture)

| Construct | Indicator |
|------------|---|
| Language | Team members think that others: <ol style="list-style-type: none"> i. have a poor command of the working language ii. write the working language poorly iii. comprehend the project working language poorly |
| Trust | Team members do not understand others accents Team members assume that other team members: <ol style="list-style-type: none"> i. do not lie ii. do not withhold information iii. will do their work to the best of their abilities iv. will complete their work even if they are not physically present to monitor work |
| Distance | Team members cannot: <ol style="list-style-type: none"> i. meet face-to-face in the same room regularly or not at all ii. cannot monitor other team members work Team members do not have direct access to all team resources Team members operate in different countries |
| Time Zones | Team members : <ol style="list-style-type: none"> i. cannot work a full day together ii. access to other team members and project sites is limited iii. meetings will occur at irregular hours to facilitate different team locations iv. experience a 24/48 hour delay can occur for queries across project locations v. in some project locations must work irregular hours to facilitate other project locations vi. operate across multiple time zones All team members cannot attend one joint project meeting |
| Culture | Team members: <ol style="list-style-type: none"> i. perceive others as having different work and social etiquette to their own ii. have different vacation time, religious holidays, public holidays across different project locations iii. use different phrasing (slang) and employ language differently across project locations iv. perceive others as having preference for specific electronic communication |

The case data indicated that operating in different Time Zones had a highly negative impact on effectiveness of Plans as coordination mechanisms. The project manager explained the problems associated with receiving estimates, or addressing project plan issues that needed to be answered quickly, but being unable to act because key team members were not available as their site was off-line. A Technical Architect on the team echoed this view, explaining that on occasions when the team was under pressure to get something done quickly, they encountered problems because that work was dependent on tasks being completed by members in different geographical locations. If those sites had a holiday period, which was often the case, team members

were not available to take action. The negative impact of Trust on Plans was also in evidence. The development manager commented that it was difficult to trust any team members due to a number of project issues which another project site had been slow to bring to light. The data also revealed that in the initial stages of work relationships team members would require other dispersed members to “prove themselves”. It was also apparent that team members would heavily monitor others at related project sites. The less physical contact members had with others the more rigorous they would be in the conduct of this activity. We therefore continue theory building by specifying four hypotheses:

H4: Distance has a moderately negative impact on the effectiveness of Plans

H5: Trust has a highly negative impact on the effectiveness of Plans

H6: Time Zones have a highly negative impact on the effectiveness of Plans

H7: Culture has a highly negative impact on the effectiveness of Plans

4.4 The Impact of Virtual Team Issues on the Effectiveness of Formal Mutual Adjustment

The analysis of the case data confirmed that the effectiveness of Formal Mutual Adjustment (FMA) was impacted by Language, Time Zones, Trust, Culture and Distance. However, the level of impact differs. While Culture and Trust impacted the effectiveness of FMA mechanisms, Distance, Time Zones and Language had a greater perceived impact. Distance negatively impacted on the effectiveness of FMA by the physical separation of team members. An example of related communication problems was reported by a Development Manager who noticed during one project conference that a key participant had “gone quiet”. Both he and his colleagues could not tell if this developer had left the call or was silent through disagreement. It took longer to work out that this team member was dissatisfied with aspects of the project. Another issue was the high usage of conference calls. A software engineer stated that large-scale conference call meetings could not focus on specific technical issues and missed many related project problems “unless they were glaringly obvious”. To resolve this issue, she often ended up having to call team members one-to-one in order to resolve specific technical items. Time Zones were also revealed as having a highly negative impact on the effectiveness of FMA as a coordination mechanism. A development manager illustrated this through an example of team members at two sites taking project conference calls at 10:30 pm (local time) in order to facilitate members based in other project sites, due to the size of the communication window. A systems engineer echoed this by stating that “usually one team or the other has to pay for it with very early morning meetings or late night-time meetings”. In relation to these working hours, team members were concerned about overly long days and the inability to perform their work at optimum levels. Language was found to have a highly negative impact on the effectiveness of FMA. For example, a software engineer stated that “even though everyone speaks the same language in meetings, phone calls...words and context are often confused and it is harder to get the message across”.

Table 3. Indicators for Coordination Mechanisms (Standards, Plans, FMA and IMA)

| Construct | Indicator |
|----------------------------------|--|
| Standards | <ul style="list-style-type: none"> • Team members follow the organization defined development process • Team members use designated templates to create documentation at each phase of the development project • Team members pass project documentation from one sub-team to another (requirements engineering, software engineering, software test etc.) • Team members review documentation to ensure consistency across the team • Team members use a set of guidelines to author an email • Team members conduct different project phases in parallel with other project phases • Team members use designated synchronization points to ensure the defined development process is being adhered to |
| Plans | <ul style="list-style-type: none"> • The team authors and maintains a project plan for the project • The team authors and maintains a software configuration management (SCM) plan for the project • The team authors and maintains a quality control plan for the project • The team calculates and publishes project estimates for the project • The team authors and maintains a requirements document • The team authors and maintains a Gantt Chart for the project |
| Formal Mutual Adjustment (FMA) | <ul style="list-style-type: none"> • Team members participate in a weekly scheduled project conference call • Team members participate in scheduled, weekly status calls • Team members at local sites attend weekly, scheduled site meetings • Team members participate in weekly, scheduled meetings for their functional area (development, test, project management etc.) • Team members maintain and share calendars • Team members author, send and/or receive emails via mail aliases |
| Informal Mutual Adjustment (IMA) | <ul style="list-style-type: none"> • Team members make and receive ad-hoc phone calls from other team members • Team members author, send and/or receive instant messages to other team members • Team members author, send and/or receive ad-hoc emails from other team members • Team members make and receive ad-hoc mobile phone calls from other team members • Team members at local project sites hold face-to-face conversations with other local team members |

A project manager at one site commented on the type of communication problems that arose between team members from different geographical locations and cultures viz. where the project working language is not the first or native language, team members do not contribute to conference calls. “People who can write reasonably legible emails, they might not speak the same. Accent and pronunciation make it

difficult to understand what they are actually saying” (Senior Software Engineer). Culture was also observed to have an impact on the effectiveness of FMA. The test team leader, for example, stated that other sites reported that team members at his site “don’t say anything. They just listen. They do not provide any active feedback!” Whereas, this could be ascribed to language difficulties, another Senior Software Engineer explained that “cultural differences have implications on how people communicate”. Trust was also found to impact the effectiveness of FMA. Team members at the sites studied argued that it took a longer period of time to build trust with team members in other project sites as opposed to co-located team members. Once trust was established, however, it could be eroded or lost when colleagues at other sites did not respond to emails, instant messages or phone calls in a timely fashion. The absence of face-to-face contact made it more difficult to rebuild trust. These findings suggest five further hypotheses:

H8: Distance has a highly negative impact on the effectiveness of FMA

H9: Trust has a moderately negative impact on the effectiveness of FMA

H10: Language has a highly negative impact on the effectiveness of FMA

H11: Time zones have a highly negative impact on the effectiveness of FMA

H12: Culture has a moderately negative impact on the effectiveness of FMA

4.5 The Impact of Virtual Team Issues on the Effectiveness of Informal Mutual Adjustment

Data from the case supported the proposition that Informal Mutual Adjustment (IMA) is significantly impacted by issues of Distance, Culture, Time Zones, Language and Trust. Distance and Time Zones have the greatest impact on IMA with other factors having a relatively lesser impact. The effectiveness of IMA is significantly impeded by Distance as team members cannot have ad-hoc, face-to-face meetings. Close face-to-face and ad-hoc interaction tends to be replaced by less effective communication mechanisms such as emails, instant messages etc. Operating in different Time Zones minimized opportunities for problem-solving interactions between team members across project locations. The Senior Development Manager on the project explained that “You could find that an issue is raised by the test team who are located in two locations. Team members, located in three separate sites should reply. They reply asking for clarification so you are back into another 24 hours”. Language was also revealed as having a moderately negative impact on the effectiveness of IMA. For example, IMA among team members was negatively affected due to the lack of fluency by those whose native tongue was not the working language. Trust was also revealed as having an impact on the effectiveness of IMA. A software developer explained that trust was difficult to build because she could not engage with colleagues face-to-face and quickly resolve issues. She was entirely dependent on the team member answering their phone, replying to her emails, or using Instant Messenger. viz. “In a virtual team everyone is isolated -we are all outsiders to other sites so trust is there but it is harder to build”. Thus, as indicated previously, it was perceived that team members took longer to build trust with team members in other locations. Culture was found to have a negligible influence on IMA. However, one developer felt that team members in some locations acted deferentially in their interactions with

him, he stated: “There is no need to be apologetic. If you made a mistake, you made a mistake. There is no need to beg my leave to do something”. We therefore conclude our theory building process by presenting five more hypotheses:

- H13: Distance has a highly negative impact on the effectiveness of IMA
- H14: Trust has a moderately negative impact on the effectiveness of IMA
- H15: Language has a moderately negative impact on the effectiveness of IMA
- H16: Time zones have a highly negative impact on the effectiveness of IMA
- H17: Culture has a low negative impact on the effectiveness of IMA

In conclusion, the case study data allowed us to (i) refine our theoretical propositions, (ii) define empirical measures for our constructs (Table 1 and Table 2), and (iii) represent this paper’s theoretical model with accompanying hypotheses (see Figure 2).

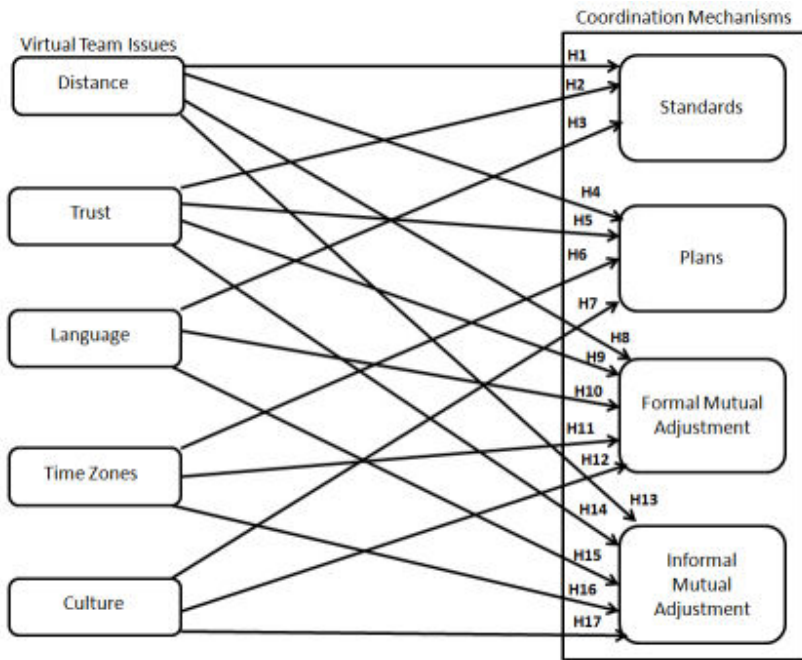


Fig. 2. Revised Theoretical Model (with associated hypotheses)

5 Summary and Conclusions

This paper has sought to contribute to a cumulative body of research on the Coordination Mechanisms employed in Global Virtual Teams (GVTs). The study focused on SD project in a multi-national telecommunications manufacturer that employed a GVT to develop a software sub-system for its 3G infrastructure product. Thus, the unit of analysis in this study provides unique insights into the complex nature of project management in a global context as it focuses on a long-term SD project. This has

implications for both theory and practice. The paper first presented a-priori theory based on prior research on project Coordination Mechanisms and extant research on the constructs of Distance, Time Zones, Language, Culture and Trust. This theory building activity resulting in (i) a bounded theory and (ii) the definition of constructs and their relationships in the form of propositions. We then employed the case study method to validate the propositions, refine the theory, and to derive empirical indicators for the constructs, and to specify hypotheses on the relationships between the constructs. This process resulted in a conceptual model of high empirical fidelity. In addition, we have shown that specific coordination mechanisms are more susceptible to the impact of GVT issues than traditional influences. The revised conceptual model presented in this paper may be employed in future research as lens through which to test the impact of virtual team issues on coordination mechanisms. While the “uptake” in the use of GVTs has increased, our understanding of the underlying team structures and processes is limited, as is our knowledge of the project management, coordination and control, of such initiatives. Thus, this study’s findings have a number of implications for practice. First, organizations looking to use GVTs for SD projects need to recognize the impact of the issues described herein on the effectiveness of coordination mechanisms. Second, organizations may need to review the coordination mechanisms employed in their GVTs. Considering the high level of task-interdependency associated with software development should a set of coordination mechanisms be tailored specifically for a global virtual team? If traditional coordination mechanisms are heavily impacted by issues such as those described, there are implications for controlling project outcomes.

References

1. Nunamaker Jr., J.F., Reinig, B.A., Briggs, R.O.: Principles for Effective Virtual Teamwork. *Communications of the ACM* 52(4), 113–117 (2009)
2. Jarvenpaa, S.L., Leidner, D.E.: Communication and Trust in Global Virtual Teams. *Organization Science* 10(6), 791–815 (1999)
3. Maznevski, M.L., Chudoba, K.M.: Bridging Space Over Time: Global Virtual Team Dynamics and Effectiveness. *Organization Science* 11(5), 473–492 (2000)
4. Ramachandran, S.: Effect of Cultural Norms on Media Choice in Global Virtual Teams. In: *Proceedings of the Eleventh Americas Conference on Information Systems*, Omaha, NE, USA, p. 1420 (2005)
5. Sarker, S., Sahay, S.: Implications of space and time for distributed work: an interpretive study of US-Norwegian systems development teams. *European Journal of Information Systems* 13(1), 3–20 (2004)
6. Majchrzak, A., Rice, R.E., Malhotra, A., King, N., Sulim, B.A.: Technology Adaptation: The Case of a Computer-Supported Inter-organizational Virtual Team. *MIS Quarterly* 24(4), 569–600 (2000)
7. Espinosa, J.A., Cummings, J.N., Wilson, J.M., Pearce, B.M.: Team Boundary Issues Across Multiple Global Firms. *Journal of Management Information Systems* 19(4), 157–190 (2003)
8. Dubé, L., Paré, G.: Global Virtual Teams. *Communications of the ACM* 44(12), 71–73 (2001)

9. Kristof, A.L., Brown, K.G., Sims, H.P., Smith, K.A.: The Virtual Team: A Case Study and Inductive Model. In: Beyerlein, M.M., Johnson, D.A., Beyerlein, S.T. (eds.) *Advances in Interdisciplinary Studies of Work Teams: Knowledge Work in Teams*, vol. 2, pp. 229–253. JAI Press, Greenwich (1995)
10. Cramton, C.: The Mutual Knowledge Problem and its Consequences for Dispersed Collaboration. *Organization Science* 12(3), 346–371 (2001)
11. Kanawattanachai, P., Yoo, Y.: The impact of Knowledge Coordination on Virtual Team Performance Over Time. *MIS Quarterly* 31(4), 783–808 (2007)
12. Ewusi-Mensah, K.: *Software Development Failures*. MIT Press, Cambridge (2003)
13. Sabherwal, R.: The Evolution of Coordination in Outsourced Software Development Projects: A Comparison of Client and Vendor Perspectives. *Information and Organisation* 13(4), 153–202 (2003)
14. Thompson, J.D.: *Organization in Action*. McGraw Hill, Chicago (1967)
15. Kraut, R., Streeter, L.A.: Coordination in Software Development. *Communications of the ACM* 38(3), 69–81 (1995)
16. Powell, A., Piccoli, G., Ives, B.: Virtual Teams: A Review of Current Literature and Directions for Future Research. *ACM SIGMIS Database* 35(1), 6–36 (2004)
17. Saunders, C., Van Slyke, C., Vogel, D.R.: My Time or Yours? Managing Time Visions in Global Virtual Teams. *Academy of Management Executive* 18(1), 19–31 (2004)
18. Herbsleb, J.D., Mockus, A., Finholt, T.A., Grinter, R.E.: Distance, Dependencies and Delay in a Global Collaboration. In: *Proceedings of the ACM Conference on Computer-Supported Cooperative Work*, Philadelphia, PA, December 2-7, pp. 319–328 (2000)
19. Kotlarsky, J., Oshri, I.: Social Ties, Knowledge Sharing and Successful Collaboration in Globally Distributed System Development Projects. *European Journal of Information Systems* 14(1), 37–48 (2005)
20. Mayer, R.C., Davis, J.H., Schoorman, F.D.: An Integrative Model of Organizational Trust. *Academy of Management Review* 20(3), 709–734 (1995)
21. Sutanto, J., Kankanhalli, A., Tan, B.C.Y.: Task Coordination in Global Virtual Teams. In: *Proceedings of the Twenty-Fifth International Conference on Information Systems (ICIS)*, Washington DC, USA, pp. 807–819 (2004)
22. Chen, H., Jiang, J.J., Klein, G., Chen, J.V.: Reducing Software Requirement Perception Gaps Through Coordination Mechanisms. *Journal of Systems and Software* 82(4), 650–655 (2009)
23. Parolia, N., Goodman, S., Yuzhu, L., Jiang, J.J.: Mediators Between Coordination and IS Project Performance. *Information & Management* 44(7), 635–645 (2007)
24. Kayworth, T.R., Leidner, D.E.: The Global Virtual Manager: A Prescription for Success. *European Management Journal* 18(2), 183–194 (2000)
25. Mortensen, M., Hinds, P.J.: Conflict and Shared Identity in Geographically Distributed Teams. *International Journal of Conflict Management* 12(3), 212–238 (2001)
26. Espinosa, J.A., Carmel, E.: Modeling Coordination Costs Due to Time Separation in Global Software Teams. In: *Proceedings of the International Workshop on Global Software Development*, part of the International Conference on Software Engineering Work in Portland, Oregon, USA (2003)
27. Zakaria, N., Amelinckx, A., Wilemond, D.: Working Together Apart? Building a Knowledge-Sharing Culture for Global Virtual Teams. *Creativity and Innovation Management* 13(1), 15–29 (2004)
28. Johansson, C., Dittrich, Y., Juustila, A.: Software Engineering Across Boundaries: Student Project in Distributed Collaboration. *IEEE Transactions on Professional Communication* 42(4), 286–296 (1999)

29. Wheeler, B.C.: A Dynamic Capabilities Theory for Assessing New-Enablement. *Information Systems Research* 13(2), 125–146 (2002)
30. Carroll, J.M., Swatman, P.A.: Structured-case: A Methodological Framework for Building Theory in Information Systems Research. *European Journal of Information Systems* 9(4), 235–242 (2000)
31. Eisenhardt, K.M.: Building Theories from Case Study Research. *Academy of Management Review* 14(4), 532–550 (1989)
32. Yin, R.K.: *Case Study Research, Design and Methods*, 2nd edn. Sage, Newbury Park (1994)
33. Benbasat, I., Goldstein, D.K., Mead, M.: The Case Research Strategy in Studies of Information Systems. *MIS Quarterly* 11(3), 369–386 (1987)
34. Patton, M.Q.: *Qualitative Evaluation and Research Methods*, 2nd edn. Sage Publications, Inc., Newbury Park (1990)
35. Strauss, A., Corbin, J.: *Basics of Qualitative Research: Grounded Theory Procedures and Techniques*. Sage Publications, Newbury Park (1990)
36. Massey, A.P., Montoya-Weiss, M.M., Hung, Y.: Because Time Matters: Temporal Coordination in Global Virtual Project Teams. *Journal of Management Information Systems* 19(4), 129–155 (2003)
37. Martins, L.L., Gilson, L.L., Maynard, M.T.: Virtual Teams: What Do We Know and Where Do We Go From Here? *Journal of Management* 30(6), 805–835 (2004)
38. Herbsleb, J.D.: Global Software Engineering: The Future of Socio-technical Coordination. In: Briand, L., Wolf, A. (eds.) *Future of Software Engineering 2007*. IEEE-CS Press, Los Alamitos (2007)
39. Jehn, K.A.: Qualitative Analysis of Conflict Types and Dimensions in Organizational Groups. *Administrative Science Quarterly* 42(3), 530–557 (1997)
40. Olson, G.M., Olson, J.S.: Distance Matters. *Human-Computer Interaction* 15(2/3), 139–178 (2000)
41. Ó’Conchúir, E., Ågerfalk, P., Olsson, H., Fitzgerald, B.: Global Software Development: Where are the Benefits? *Communications of the ACM* 52(8), 127–131 (2009)
42. Herbsleb, J.D.: Global Software Engineering: The Future of Socio-technical Coordination. In: *Future of Software Engineering, International Conference on Software Engineering*, Washington, DC, USA, May 23–25, pp. 188–198. IEEE Computer Society, Los Alamitos (2007)

Information Systems Evolution over the Last 15 Years

Magne Davidsen and John Krogstie

IDI, NTNU, Trondheim, Norway

Magne.davidsen@gmail.com, krogstie@idi.ntnu.no

Abstract. The information systems we see around us today are at first sight very different from those that were developed 15 years ago and more. On the other hand, it seems that we are still struggling with many of the same problems. To understand how we can evolve future ISs, we should have good understanding of the existing application portfolios. In this article we present selected data from survey investigations performed in 1993, 1998, 2003 and 2008 among Norwegian organizations on how they conduct information systems development and evolution. A major finding is that even if we witness large changes in the underlying implementation technology and approaches used, a number of aspects such as the overall percentage of time used for maintaining and evolving systems in production compared to time used for development is stable, and should be taken into account in the planning of information systems evolution for the future.

Keywords: Information systems evolution, maintenance.

1 Introduction

Modern information systems are the result of the interconnection of systems of many organizations, which are running in variable contexts, and require the capability to actively react to changing requirements and failures. Large changes in how we develop information systems have been witnessed over the last 15-20 years. The prevalent development methods, programming languages and general technological infrastructure have changed a lot. In the early nineties, one was going from mainframe solutions to a client-server, and then to an internet architecture for most applications. Usage of packages has been on the rise. Lately SOA, free and open source software, outsourcing and agile development are also expected to have impact. According to [9] one of the impacts on the state of IS-development is the increasing amount of time used for maintenance and support of systems, contrary to what is claimed to be the impact of e.g. the introduction of SOA.

In this paper, we present descriptive results from a survey-investigation performed in Norwegian organizations in this area during the end of 2008, comparing with similar investigations done in 2003, 1998 and 1993. We will first give definitions of important terms used within information systems evolution. We describe the research method, before the main descriptive results from our investigation are presented and compared with previous investigations. The last section summarizes our results.

2 Definition of Core Concepts

Maintenance is defined as the process of modifying a software system or component after delivery to production. It has traditionally been divided into three types: corrective, adaptive and perfective [8] inspired by, e.g. [21]. Corrective maintenance is performed to correct faults in hardware and software. Adaptive maintenance is performed to make the computer program usable in a changed environment. Perfective maintenance is performed to improve the performance, maintainability, or other attributes of the program. This has been divided into enhanceive maintenance [2] and non-functional perfective maintenance. Enhanceive maintenance implies changes and additions to the functionality offered to the users by the system. Non-functional perfective maintenance implies improvements to the quality of the system. In addition to the temporal distinction between development and maintenance, we have introduced the concepts application portfolio evolution and application portfolio upkeep (previously termed functional development and functional maintenance [10]).

1. Application portfolio evolution: Development or maintenance where changes in the application increase the functional coverage of the total application systems portfolio of the organization. This includes development of new systems that support new areas and enhanceive maintenance.
2. Application portfolio upkeep: Work made to keep up the functional coverage of the information system portfolio of the organization. This includes corrective adaptive and non-functional perfective maintenance and development of replacement system

We note that some writers provide more detailed overview of maintenance tasks [3, 9]. Jones [9] has in total 21 categories, also including user-support to be part of maintenance (a view shared with, Dekleva [4], but not with us).

3 Research Methodology

Our survey form was implemented in the SurveyMonkey web-tool and invitations were distributed by e-mail to 278 Norwegian organizations. The organizations were randomly selected from the list of member organizations of The Norwegian Computer Society – NCS. (NCS has currently around 1000 member organizations), using the same approach as in the previous investigations. The contents of the form were based on previous investigations within this area; especially those described in [7, 10, 12, 15, 16, 18, 22]. 79 responses were returned, giving a response rate of 28%. Out of these, 67 responses could be used for the analysis since the additional 12 responses were incomplete. This was a higher response rate than in previous investigations.

The forms were filled in by people with long experience with IT-related work (average 17.5 years), most filling the role as IT director in the company. We will compare some of the results with the results of similar investigations in particular.

1. The investigation carried out by Krogstie, Jahr and Sjøberg in 2003 [12].
2. The investigation carried out by Holgeid, Krogstie and Sjøberg [7] in 1998.
3. The investigation carried out by Krogstie [10, 11] in 1993.

These surveys contain the results from 54, 52 and 53 organizations, respectively. A number of similar investigations of this type were performed earlier in the USA [16, 18, 22]. Comparisons with these surveys have been done in [12]. A number of later investigations have been done, but they only look on the distribution of maintenance tasks [6, 13, 17, 19], looking on the situation in a few organization.

4 Selected Results

First, we present some of the overall demographics of the survey. Similar results from our previous surveys conducted in 2003, 1998, and 1993 are included in parenthesis where the numbers are comparable. The mean number of main systems in the organizations was 7.8 (2003-4.5; 1998-9.6; 1993-10.3). The mean user population of these systems was 4661 (2003-314; 1998-498; 1993-541). It is in particular the number of external users that has increased dramatically (also relative to number of employees in the organizations); the average number of internal users was 944. The average age of the systems was 4.9 years (2003-3.9; 1998-6.4; 1993-4.6).

In 1993, 58% of the systems were developed by the IS-organization, and only one percent was developed *in* the user organization. In 1998, however, 27% of the systems were developed by the IS-organization and 27% as custom systems *in* the user organization. In 2003 23 % of the systems are developed in the IS-organization, whereas in 2008 only 12% was developed in the IS organization. The percentage of systems developed by outside firms is higher (40% vs. 35% in 2003, vs. 22% in 1998 vs. 12 % in 1993). The percentage of systems developed based on packages with small or large adjustments is also comparatively high (41% vs. 39% in 2003 vs. 24% in 1998 vs. 28% in 1993). The new category we introduced in 1998, component-based development (renamed “use of external web services” in 2008) is still small (5%) although increasing (1.0 % in 2003, 0.4% in 1998) of the total systems.

From being dominant ten to fifteen years ago COBOL is almost not used anymore. The languages that are used in most organizations and for most systems are now Java (40%, 27% in 2003) and C++ (33%, 24% in 2003). Java was just starting to be in widespread use in 1998 and C++ was barely included in 1993. The percentage of organizations reporting to have COBOL applications has decreased from 73% in 1993 to 26% in 1998 to 5% in 2008.

94 new systems were currently being developed; 60 of these systems (64 %) were regarded as replacement systems. (2003-60%; 1998-57%; 1993-48%). 13% of the current portfolio was being replaced. (2003-13%; 1998-9%; 1993-11%). The average age of systems to be replaced was 7 years (2003 - 5.5; 1998 - 7.7; 1993 - 8.5).

Reasons for system replacements have changed slightly from earlier investigations. The most important reasons for replacement are need for integration and burden to maintain and operate, a bit surprising giving the relatively young age of the systems that are replaced. One area which is expected to influence the software development and maintenance landscape is Service Oriented Architecture (SOA) [14]. Transfer to SOA was very important as a reason to create replacement systems for only two organizations. Less than 20% of the organizations had started implementing SOA, and we could not find any significant impact on the use of SOA on maintenance figures.

Work on application systems was in the survey divided into the six categories presented in section 2. The same categories were also used in 1993, 1998 and 2003. We also asked for the time used for user-support and for systems operations which took up the additional time for the work in the IS departments.

In earlier investigation of this sort between 50% and 60% of the effort is done to enhance systems in operation (maintenance) when disregarding other work than development and maintenance. An exception from this was our study in 1998 that was influenced particularly by the amount of Y2K-oriented maintenance. The numbers for Dekleva [4] and those reported by Capers Jones [9] were also higher than this, but these also include user support as part of maintenance.

Table 1 summarizes the descriptive results on the distribution of work in the categories in our investigation, comparing to previous investigations.

Table 1. Distribution of the work done by IS-departments in percentage

| Category | 2008 | 2003 | 1998 | 1993 |
|---------------------------------|-------------|-------------|-------------|-------------|
| Corrective | 8.2 | 8.8 | 12.7 | 10.4 |
| Adaptive | 6.2 | 7.3 | 8.2 | 4 |
| Enhancive | 11.3 | 12.9 | 15.2 | 20.4 |
| Non-functional perfective | 9.1 | 7.6 | 5.4 | 5.2 |
| <i>Total maintenance</i> | 34.9 | 36.7 | 41.4 | 40 |
| Replacement | 9.7 | 9.9 | 7.7 | 11.2 |
| New development | 11.4 | 12.6 | 9.5 | 18.4 |
| <i>Total development</i> | 21.1 | 22.5 | 17.1 | 29.6 |
| Technical operation | 23.7 | 23.8 | 23 | NA |
| User support | 20.1 | 17.1 | 18.6 | NA |
| <i>Total other</i> | 44.0 | 40.8 | 41.6 | 30.4 |

When disregarding other work than development and maintenance of application systems, the percentages are as follows: maintenance activities: 65, 7%, development activities: 34.3%. This is at the same level as in 2003. 63% of development and maintenance work was application portfolio upkeep, and 37% was application portfolio evolution. This is almost the same as in 2003 and 1998, which in turn was significantly different from the situation in 1993 where application portfolio upkeep- and application portfolio evolution respectively amounted to 44% and 56% of the work.

Looking in detail on the distribution of maintenance activities, a number of later studies have been looking at this in particular. As stated in [5] corrective efforts is time-consuming. As reported in [1], it appears to be very large differences reported in different studies. Whereas Lientz/Swanson [16] reported 60% perfective, 18% adaptive and 17% corrective maintenance when asking about selected systems from a large number of organizations (one per organization), Sousa [33] reported (based on a number of systems in one organization) 49% adaptive, 36% corrective, and 14 % perfective maintenance. [17] reported 53% corrective, 36% perfective and 4% adaptive maintenance, based on data on three open source products. [13] reported 62% perfective, 32% corrective, and 6% adaptive maintenance based on data from one application in production. Most interesting for comparison with other surveys is

looking at corrective, adaptive, and perfective maintenance, which appears to be much more stable than the numbers reported from others above when looking upon this across our investigations looking on the application portfolios of a number of organizations.

5 Conclusions and Further Work

There are a number of differences in the underlying technology, which is as expected. This is very clearly witnessed in the distribution of programming languages used, where procedurally languages like COBOL have to a large extent been suppressed by object-oriented languages like Java, C++ and C#. New architectural trends such as SOA have yet to make a noticeable impact on the use of resources. Another marked difference is that less and less of IT is done internally in organizations (this applies to development, maintenance, operations and use), which will make appropriate software evolution harder to do. On the other hand, even if most organizations outsource part of the IT-activities, most still do some of the activities in house. Overall percentage of time used for evolving systems in production compared to time used for development is remarkably stable on average (these numbers differs a lot from year to year within individual organizations). The same can be said about the rate of replacement, although slightly increasing, more than 60% of 'new' systems to be developed are actually replacement systems, constituting around 13% of the current application portfolio. Since more complex infrastructures are supporting the information systems, more and more of the resources are used for other tasks such as operations and user-support, less and less time is available for providing new information systems support in organization, although it seems to have plateau on 20% of the overall time, a level reached already ten years ago in Norway (i.e. earlier than indicated in [9]). Even when using new approaches where evolution is better supported by the system itself and the design of the information system consider evolution as an inherent property of the system, it will take quite some time until such technology is used broadly. Waiting for this, we have to keep the system operational and continuously evolving, addressing new needs currently not known.

Several of our results have spurred new areas that could be interesting to follow up on in further investigations, and we are currently performing several detailed case studies. A long-term plan is to do a similar investigation in 2013.

References

1. Benestad, H.C., Anda, B.C.D., Arisholm, E.: Understanding software maintenance and evolution by analyzing individual changes: A literature review. *Journal of Software Maintenance and Evolution: Research and Practice* (2009)
2. Chapin, N.: Software Maintenance Types – A Fresh View. In: *Proceedings of the International Conference on Software Maintenance (ICSM 2000)*, pp. 247–252 (2000)
3. Chapin, N., Hale, J., Khan, K., Ramil, J., Tan, W.-T.: Types of Software Evolution and Software Maintenance. *Journal of Software Maintenance* (13), 3–30 (2001)
4. Dekleva, S.M.: Software Maintenance: 1990 Status. *Journal of Software Maintenance* 4, 233–247 (1992)

5. Ghazarian, A.: A Case Study of Defect Introduction Mechanisms. In: van Eck, P., Gordijn, J., Wieringa, R. (eds.) CAiSE 2009. LNCS, vol. 5565, pp. 156–170. Springer, Heidelberg (2009)
6. Gupta, A., Slyngstad, O.P., Conradi, R., Mohagheghi, P., Rønneberg, H., Landre, E.: An Empirical Study of Software Changes in Statoil ASA - Origin, Priority Level and Relation to Component Size. In: Proceedings of the international Conference on Software Engineering Advances, ICSEA, Washington, DC, October 29–November 03, p. 12. IEEE Computer Society, Los Alamitos (2006)
7. Holgeid, K.K., Krogstie, J., Sjøberg, D.I.K.: A study of development and maintenance in Norway: Assessing the efficiency of information systems support using functional maintenance. *Information and Software Technology* 42, 687–700 (2000)
8. IEEE Standard Glossary of Software Engineering Terminology (1991)
9. Jones, C.: *The Economics of Software Maintenance in the Twenty First Century* (2006), <http://www.compaid.com/caiinternet/ezone/capersjones-maintenance.pdf> (last accessed February 2010)
10. Krogstie, J., Sjølvberg, A.: Software maintenance in Norway: A survey investigation. In: Muller, H., Georges, M. (eds.) Proceedings of the International Conference on Software Maintenance (ICSM 1994), Victoria, Canada, pp. 304–313. IEEE CS Press, Los Alamitos (1994)
11. Krogstie, J.: On the distinction between functional development and functional maintenance. *Journal of Software Maintenance* 7, 383–403 (1995)
12. Krogstie, J., Jahr, A., Sjøberg, D.I.K.: A Longitudinal Study of Development and Maintenance in Norway: Report from the 2003 Investigation. *Information and Software Technology* 48, 993–1005 (2006)
13. Lee, M.-G., Jefferson, T.L.: An Empirical Study of Software Maintenance of a Web-Based Java Application. In: ICSM 2005 (2005)
14. Lewis, G.A., Smith, D.B.: Service-oriented Architecture and its Implications for Software Maintenance and Evolution Frontiers of Software Maintenance. In: FoSM 2008 (2008)
15. Lientz, B.P., Swanson, E.B., Tompkins, G.E.: Characteristics of application software maintenance. *Communications of the ACM* 21(6), 466–471 (1978)
16. Lientz, B.P., Swanson, E.B.: *Software Maintenance Management*. Addison-Wesley, Reading (1980)
17. Mohagheghi, P., Conradi, R.: An Empirical Study of Software Change: Origin, Acceptance Rate and Functionality vs. Quality Attributes. In: International Symposium on Empirical Software Engineering (ISESE 2004), pp. 7–16 (2004)
18. Nosek, J.T., Palvia, P.: Software maintenance management: Changes in the last decade. *Journal of Software Maintenance* 2, 157–174 (1990)
19. Schach, S.R., Jin, B., Yu, L., Heller, G.Z., Offutt, J.: Determining the Distribution of Maintenance Categories: Survey versus Measurement. *Empirical Software Engineering* 8(4), 351–365 (2003)
20. Sousa, H., Moreira, H.: A Survey of the Software Maintenance Process. In: Proceedings of ICSM 1998, Bethesda, Maryland, pp. 268–274. IEEE CS Press, Los Alamitos (1998)
21. Swanson, E.B.: The dimensions of maintenance. In: Proceedings of the Second International Conference on Software Engineering, San Francisco, USA, August 1976, pp. 492–497 (1976)
22. Swanson, E.B., Beath, C.M.: *Maintaining Information Systems in Organizations*. Wiley Series in Information Systems. John Wiley & Sons, Chichester (1989)

From Web Data to Entities and Back

Zoltán Miklós¹, Nicolas Bonvin¹, Paolo Bouquet², Michele Catasta¹,
Daniele Cordioli³, Peter Fankhauser⁴, Julien Gaugaz⁴, Ekaterini Ioannou⁴,
Hristo Koshutanski⁵, Antonio Maña⁵, Claudia Niederée⁴, Themis Palpanas¹,
and Heiko Stoermer¹

¹ Ecole Polytechnique Fédérale de Lausanne (EPFL)

`name.surname@epfl.ch`

² DISI, University of Trento

`{stoermer,bouquet,themis}@disi.unitn.it`

³ ExpertSystem s.p.a, Modena, Italy

`dcordioli@expertsystem.it`

⁴ L3S Research Center, Leibniz Universität Hannover

`surname@L3S.de`

⁵ Universidad de Málaga

`{hristo,amg}@lcc.uma.es`

Abstract. We present the Entity Name System (ENS), an enabling infrastructure, which can host descriptions of named entities and provide unique identifiers, on large-scale. In this way, it opens new perspectives to realize entity-oriented, rather than keyword-oriented, Web information systems. We describe the architecture and the functionality of the ENS, along with tools, which all contribute to realize the Web of entities.

Keywords: entity, Web, unique identifier.

1 Introduction

The information need of Web users is often related to their understanding and interpretation of real-world entities and their relationships. For example, someone might be interested in information about Paris, the picturesque capital of France. This mental interpretation of humans is very challenging to capture, thus information systems need a simplified representation.

The information on the Web is present mainly in the form of unstructured or semi-structured documents, thus it is very natural to model the Web data as a list of keywords, organized in documents, which might have links to each other. This simplified model was extremely successful, the Web search engines built around this model are used by millions of users every day.

This simple model however has a number of limitations, including the problem that the keywords are ambiguous. For example, if the keyword “Paris” appears on a page, it is not immediately clear, if this refers to the capital of France, to a city Paris in Texas, US or to the first name of Paris Hilton. This ambiguity is very problematic for human users, but it can cause more problems, if one intends to process the

page automatically. There are numerous efforts to address this ambiguity problem, notably the Semantic Web community has proposed a number of solutions.

In this paper we present an information system, accompanied by a number of extendable tools, which enables to realize the “Web of entities”. As opposed to the keyword-based model, in the Web of entities the keywords pointing to named entities, such as persons, geographic locations, etc., are annotated with a reference to an entity description. These entity descriptions contain important information about the entities and also have a unique identifier (OKKAM ID). The basic idea of the Entity Name System (ENS), is to provide a free, open service to the users of the (Semantic) Web, such that they can annotate the Web content with references to entities.

We describe the design and architecture of the ENS. The main functionality of the ENS is to process entity search requests and return a unique identifier (OKKAM ID) of the entity. In this way one can annotate the page or the keyword with this unique identifier, so later this unique ID can be used to avoid ambiguities. We designed the ENS in a way such that it can work on web-scale and provide well within second response time even in the presence of a large number of users. The services of the ENS are openly available via Web Services, thus enabling Web application developers to adopt an entity-oriented, rather than keyword-oriented model. We already integrated this support into a number of popular tools, such as gmail, blogging, MS Word, etc. We populated the repository with an initial set of entities (with data related to LOD¹), but the goal is to further continuously populate the entity descriptions, with the help of the tools, or through manual entity creation: similar to wikipedia, anyone can enter new entity descriptions to ENS. As entity descriptions might naturally evolve over time, ENS has support to accommodate these activities.

The rest of the paper is organized as follows. Section 2 discusses entity-oriented information systems, Section 3 explains the architecture and the functionality of the ENS. Section 4 presents a set of tools, which both use the services of the ENS and contribute to populate the entity repository. Section 5 evaluates how the ENS and the tools contribute towards realizing a Web of entities. Section 6 provides related work and finally Section 7 concludes the paper.

2 Entity-Oriented Information Systems

The ENS was developed in the course of the OKKAM² project. The overall goal of the project is to enable the Web of entities, a global digital space for publishing and managing information about real-world entities, where each entity is linked to a unique identifier, named OKKAM id. The ENS plays a central role achieving this goal: the ENS can store a large number of entity descriptions, it assigns the unique OKKAM identifiers to the descriptions (entity profiles). Furthermore, the ENS provides an entity search service, which returns the unique identifier of the requested entity.

¹ Linked Open Data, <http://linkeddata.org/>

² <http://www.okkam.org>

This infrastructure intends to open the possibility to develop entity-oriented applications, by reusing the unique identifier one can avoid a number of ambiguity-related problems. We envision a continuous feedback between entity use and content creation. The OKKAM empowered tools (Section 4) shall recognize named entities in plain text documents and obtain the corresponding unique identifiers. This process is often called OKKAMization. On the one hand, the ID can be reused, but on the other hand, the user can provide feedback, enter or update entity descriptions into the ENS. In this way, the entity population of the ENS will be continuously improved and enlarged, for the profit of the whole community.

3 The Entity Name System

In this section we give an overview of the functions of the ENS and its main components. In particular, Section 3.1 describes the main components of the system, Section 3.2 discusses the data population (currently available in the ENS) and its representation, Section 3.3 gives details about the storage infrastructure, Section 3.4 provides an overview of the entity search functionality, while the entity lifecycle management is discussed in Section 3.5. Finally, Section 3.6 discusses the security infrastructure of the ENS.

3.1 Overall System Architecture

The *ENS* consists of the following four main components:

1. **Entity Store.** This component is responsible for the low-level tasks of storing and managing entity data, as well as relevant metadata which are necessary for finding existing identifiers. It follows the usual create-read-update-delete pattern (CRUD) [16] of serialized entity data accessed via a primary key.
2. **Index.** Search operations for an identifier via an entity profile in the *ENS* are accelerated using index structures.
3. **ENS Core.** This component is implementing the services that are exposed by the *ENS*, namely, entity creation and update, matching, as well as processes relevant to the entity lifecycle management.
4. **Offline Processing.** This component handles tasks which shall be performed offline rather than request processing time, including statistics computations, data quality related jobs, etc.

The *ENS* Core itself consists of several parts that we describe in the following. The **Storage API** abstracts from the complexities of accessing the Entity Store and Index components mentioned above. In particular, the underlying strategies for addressing the individual, load-balanced clusters and retrieving all relevant data are hidden from upper layers. The **Matching** component is responsible for ensuring a high top-k precision of entity identifier search results. It has two main tasks: (i) parsing, analyzing and – if necessary rewriting or expanding – the search request coming from a client, and (ii) applying specific ranking mechanisms with the aim to establish the best match between the search request and

the candidate entities. The **Lifecycle Manager** takes care of the entity creation- and update-related tasks, and the important aspects of data quality and data lifecycle management. The **Access Manager** ensures that the requirements of access control and privacy are fulfilled. It performs, among other things, query and result set filtering to avoid undesired use of the system. All services that the *ENS* offers as its public API are exposed through the **Web Services** component, which constitutes the uppermost layer of the *ENS* component stack. The *ENS* also equipped with an end-user frontend for identifier search and entity creation, which is available at <http://api.okkam.org/search/>.

3.2 Data

Entity Representation and Request Language. In the *ENS*, an entity is identified by an entity identifier, *oid*, assigned by the system. This ID is often referred as OKKAM ID. Along with this identifier, we store a set of alternative ids, *Aid*-s, that other systems have assigned to the same entity, which can also be used to access the entity or produce *same-as* statements for the LOD community (see Sect. 4.2). The preferred id, *prid*, is one of the above identifiers that we use when displaying the entity³. The descriptive part of the entity profile contains a set of attribute name value pairs that describe the entity. Some of these descriptive attribute-value pairs might contain external references, such as for example links to Web documents describing the entities. Finally, we also store a set of metadata for each entity, that include usage statistics, and provenance and access control metadata. All the above information is individually indexed to allow fast access to it, and then stored as a single XML file on disk.

Our goal in *ENS* is to keep the entity representation simple and at the same time as general as possible, without imposing a fixed schema of entity types or attributes [24]. In order to reach a compromise between the generality of the description and the precision in the functionality of the system, we use an internal classification of all entities in six broad, top-level categories, namely, *person*, *organization*, *location*, *event*, *artifact*, and *other*. For each one of these entity types, we have identified a set of default attributes that are most commonly used to describe them [2]. These default attributes are suggested to users during entity creation, but they are of course optional. Nevertheless, we expect that most of the created entities will use (at least some of) the default attributes (for example, give a name for a person), which in turn will have positive influence on the performance of the system.

We adopted a simple syntax for the entity ID request language: a request may contain a (list of) keywords and a (list of) attribute value pairs. For example, “Paris” or “first name=Paris”. For a more precise description of the syntax we refer to <http://community.okkam.org/index.php/Documentation/APIs/entity-id-request-language.html>.

³ The only guaranteed, unique identifier is still the *oid*, which is always used internally in the *ENS*.

Entity Population. We populated the repository of the ENS in two different ways. We imported openly available datasets (using their structured representations available online), such as Wikipedia (ca.700 000 entities) and Geonames⁴ (ca. 6.5 million entities) to have an initial rich population⁵. We also added further entities manually, through the administrative interface of the ENS, so the current size of the entity population (as of February 2010) of the ENS is ca. 8 million.

3.3 Storage

The storage layer provides efficient means to store entity profiles and serves as an underlying infrastructure for the ENS. The storage layer also provides services for other components of the ENS, for example to realize entity search. The ENS adopts a two phase entity search approach. As a response to a (possibly reformulated) user query the storage layer first obtains a list of top-k candidates, which is then further processed in the search component (see Section 3.4). Thus, the primary goal of the storage layer to answer these queries with a very high recall, i.e. whenever a matching entity profile exists in the repository, it should be returned with a high probability.

The storage layer consists of two main building blocks: a key-value store and an inverted index [20]. The inverted index is used for query processing, to find the relevant documents (i.e. entity profiles), while the key-value store contains the entity profiles themselves. This organization of components is similar to the architecture of Web search engines [8]. The ENS shares many requirements with search engines, for example storing a large collection of files, having a high availability, fault tolerance, being able to process queries well below a second, even in the case of high query load, etc. To address these requirements we employed the same or similar techniques which are -to best of our knowledge- used to build Web search engines [8]. These techniques include the distribution of both the index and the data over several machines, replication of both the data and the inverted index and document partitioning.

There are however important differences to Search engines. The request language supported in the ENS allows to specify attributes to keywords (see Section 3.2). For example, the user might issue a query “city:Paris country:France”, not just a simple keyword oriented query “Paris France”. In order to support these queries, we apply a specific index structure. As opposed to the inverted indexes used by the Web search engines, our posting lists also contain the attribute information in addition to the document identifier, where the keyword occurs, for each element in the posting list. The other main difference is the ranking: we developed a ranking scheme specifically tailored for entity profiles.

The implementation of our storage relies on Hadoop⁶, while we have chosen Apache Solr⁷ to realize the large and distributed retrieval indexes.

⁴ <http://www.geonames.org/>

⁵ Overlaps between the datasets were eliminated by limiting the wikipedia import to entities of type person and organization.

⁶ <http://hadoop.apache.org/>

⁷ <http://lucene.apache.org/solr/>

3.4 Entity Search

A core functionality of the ENS is effective entity search: given a description of an entity, identify and return the corresponding entity already from the ENS's repository and then return the respective OKKAM identifier. Internally this translates into the matching of the requested entity with the entity descriptions available in the repository of the ENS.

At first sight, the entity search task has a strong similarity with entity linkage techniques [14], also known as data matching [49], deduplication [27], resolution [3], merge-purge [13], entity identification [21], and reference reconciliation [10]. Entity Linkage is the process that decides whether two descriptions refer to the same real world entity (see [12] for an overview). Actually, state-of-the-art methods from this area have also been reused and adapted in implementing entity search.

However, there are some major additional challenges, since—in contrast to the ENS—entity linkage typically relies on a well-defined schemata. In the ENS, the need for such an agreement on a joint schema or ontology has been deliberately omitted, since it is generally accepted that such an agreement is not viable in a large, heterogeneous, and evolving environments that involve various user communities. Instead, our entity search assumes a flexible data model, also envisioned for dataspace [11] (see also Section 3.2).

Entity search will accept entity requests from a wide variety of agents, including humans as well as applications. The employed very loose schema commitment and constraints within the repository, along with the variety of requesters, imposes a set of additional challenges on the described search task:

A. Over-Specification. The agent requesting an entity is not expected to have knowledge about the repository's data. Thus, the agent will use the information available to him, for example, extracted from text, for describing the searched entity. As a result, the information of the requested entity might have more or other knowledge about the entity than the ENS repository. Due to this, the interpretation of ENS requests deviate from the traditional query processing, which expects to only receive information that their systems contain. Entity search of ENS is forced to always consider that a given entity might contain information which the repository does not have.

B. Under-Specification. An opposite situation with over-specification is when the information of the requested entity is not sufficient to do a full disambiguation. This missing information can be collected in different ways, for example user interaction, analysis of repository entities (statistics), or analysis of previously requested entities.

C. Schema Heterogeneity. Omitting the commitment to an agreement on a common schema will leads to schema heterogeneity, which we have to deal with when processing the entity requests.

As explained in Section 3.3, the search process in the ENS is divided into two main phases. First a set of candidates is selected, then they are ranked with

the help of additional domain knowledge and heuristics. Below, we explain this second phase, which relies on a Generic Matching Module (explained in the next paragraphs). This module is extensible, one can implement additional modules for specific types of entity profiles.

The input to the **Generic Matching Module** is a request in our simple request language see Section 3.2. A request consists of segments, which may either be attribute value pairs or unqualified values. The requests intend to find an entity profile, that we interpret such that that each segment of a request *should* match the sought entity, much like usually done in Information Retrieval. It takes into account fuzzy matches between attribute names and values, and also partial mismatches, which may arise due to over-specification.

Given a request and a candidate entity, the Generic Matching Module computes a final *matching score* resulting from the aggregation of several features of two kinds: attribute level features and entity level features. The attribute level features are attribute *label similarity*, attribute *value similarity* and an attribute *boosting* factor. These three features are aggregated into *attribute similarity*. For each request segment, the entity attribute with the maximum attribute similarity is chosen, and the individual *segment similarities* are aggregated to obtain the first entity level feature: the *entity similarity*. The second kind of entity level feature is the *entity popularity*, which further differentiates matching entities especially for under-specified queries. Finally, in order to deal with geographical entities, two special entity level features are dedicated to them: the *location type* and *location population*. This gives four entity level scores which are aggregated to the final score for each candidate matching entity.

At all levels, feature scores are aggregated by a weighted log-based tempered geometric mean. Given the scores to aggregate $S = \{s_1, \dots, s_t\}$ and their respective weights $W = \{w_1, \dots, w_t\}$, the aggregated score is

$$\epsilon GM(S, W) = \exp \frac{\sum_{i=1}^t w_i \log(s_i + \epsilon)}{\sum_{j=1}^t w_j} - \epsilon$$

This is a weighted version of the ϵ -adjusted *geometric mean* proposed by Robertson et al. in [26]. According to [25] the geometric mean is less affected by outlying values than the arithmetic mean, and since it is based on multiplication, there is no requirement that the scores are on the same scale. The ϵ -adjustment avoids a null score forcing the aggregation to zero, and computing in the log space helps avoiding underflows.

The label and value similarities are standard string similarity metrics⁸ from the SimMetrics library⁹. The attribute boosting factor is a combination of two values: attribute selectivity and attribute popularity. We determined these factors through statistics on the data and query logs and through extensive experimentation.

⁸ Respectively Needleman-Wunch and a combination of Levensthein and Monge-Elkan

⁹ <http://www.dcs.shef.ac.uk/~sam/stringmetrics.html>

3.5 Entity Lifecycle Management

Lifecycle management of entities in the ENS includes aspects of entity representation, data quality, repository evolution, and online monitoring of the use of the repository, which we are going to illustrate in the following.

Data Quality. The aim of data quality at entity creation time is to ensure that the new entities satisfy a minimal set of quality requirements. This assessment also takes place when a new entity profile is created or when an existing entity is being modified.

The data quality assessment process at creation time that is currently in place within ENS, consists of the following three types of checks.

1. Attribute value quality, where we want to ensure that the values entered for the various attributes in the entity description are correct and valid (when possible). For example, we check that the provided attribute value is not empty or overly long, and that date and time attributes adhere to some known format.
2. Intra-entity description quality, where we check the quality of the entity description as a whole. For example we check whether attributes appear with the same name and different value (that would lead to a warning), etc.
3. Inter-entity description quality, where we check that changes in the repository will not degrade the overall quality performance of the system. In particular, we make sure that modifications do not introduce duplicate entity profiles, which would degrade the quality of search results. The duplicate detection relies on the matching techniques, discussed in Section [3.4](#).

Evolution of Identifiers. Even though the identifier of an entity should never change, in some special circumstances this may happen. For example, if we realize that two different entity profiles refer to the same real world entity, or when the same entity profile is already being used to refer to two distinct real world entities. In these cases, we would like to take corrective action, by performing a *merge* and a *split*, respectively.

The ENS supports the merge and split operations as follows. In the case of a merge operation, we select one of the existing identifiers (i.e., the one that according to the system statistics belongs to the most popular entity of the two) to be the identifier of the merged entity. The other identifier will still exist, so that users can refer to it, but only the selected identifier will be returned as a result. When splitting an entity representation into two new ones, we have no option but creating two new identifiers, since the old identifier has been used to refer to a non-existing (wrong) entity. In both cases, the system keeps the history of changes in order to be able to undo these operations. We also provide a service that makes the information about these changes available for users (and/or machines) to read, however in the current version the merge/split operations can be performed only manually.

Monitoring of Repository Usage. The way that users access the system and interact with it may provide useful insight on what actions to take in order to

improve the performance of the ENS. Therefore, we monitor the data streams relevant to the usage patterns of the system. We have extended and adapted algorithms that can operate in an online fashion, and are flexible enough to allow effective and efficient data analysis of the incoming data streams [28,19]. By monitoring and analyzing the way users interact with the ENS we can further improve the ENS services.

3.6 Security and Trust

To provide services for a large number of users, the ENS has a specific set of security requirements. These include access control requirements (fine grained policies, easy policy specification, automated access control enforcement), legal and privacy requirements (to be able to control in a confidential way, who can modify entity profiles) and usability requirements.

Our security architecture is based on advanced use of certificate technologies. Figure 1 shows a high-level view of the main elements of the security infrastructure and their interactions. Trust in ENS community is based on certificate authorities that qualify the ENS, third party service providers and users by means of identity and attribute certificates, compliant with X.509 [29] standard. The infrastructure adopts several widely-used security technologies such as https [10], Web Services security standards [11], and secure e-mail [12].

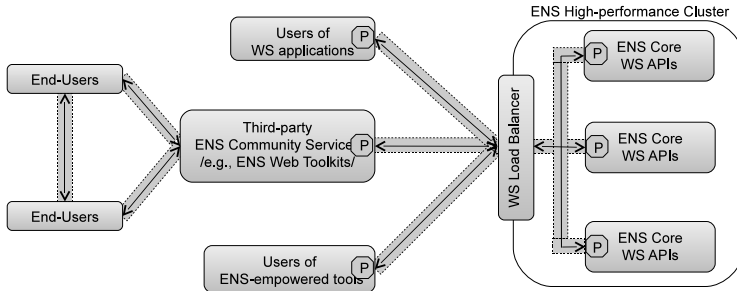


Fig. 1. ENS Security Architecture

All communications between a user and the ENS are realized through security proxies located at both at the ENS and the user sides. We have chosen a proxy-based solution to decouple all security management and technological aspects from application-level development. The proxy component allows transparent security management, so that also thin ENS-empowered tools (e.g., MS Word plug-in, see Section 4) can achieve a high level of secure communications. The proxy implements latest WS security standards [13] and advanced authorization process based on certificate-based automated trust negotiation model [17].

¹⁰ Secure HTTP communications based on SSL/TLS standards.

¹¹ WS-Security, WS-SecureConversation etc. at <http://www.oasis-open.org/specs>

¹² OpenPGP-compliant e-mail security <http://tools.ietf.org/html/rfc4880>

¹³ Using Metro high-performance Web Services stack at <https://metro.dev.java.net>

4 Tools

This section discusses tools, which benefit from the services of the ENS. OKKAM Empowered Tools are extensions of existing tools that both use the services of ENS, to enrich the processed plain text with entities and at the same time enable content creation: if an entity description does not exist in the ENS, the user might decide to create it. In this way, while they benefit from ENS, the ENS and the whole community benefits from the new content. This mutual feedback mechanism is intentional and shall foster the adoption of the ENS.

In the course of the OKKAM EU project, a whole suite of such tools is developed. These tools include an extension to ontology editors Protege and NeON [18], plugins for office products (MS word, outlook, Open Office), plugins for popular browsers (e.g. Firefox). In the following we demonstrate the use of OKKAM-plugins for popular Web tools.

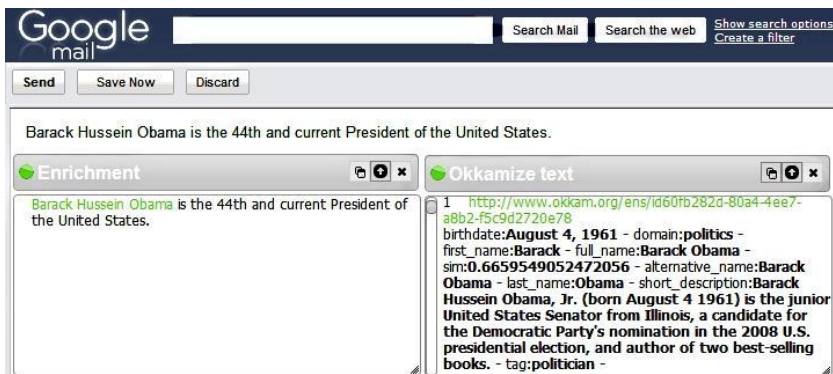


Fig. 2. Okkam4Gmail plugin

4.1 For Web Users

The use of *Okkam4Gmail*, the Okkam plugin for Google's webmail is depicted in Figure 2. The user can enrich and automatically annotate the text he enters. The extension identifies precisely and automatically entities by linguistic disambiguation techniques. They include morphological, grammar, syntactic and semantic analysis of the text. All fundamental information for the disambiguation process, i.e. the whole system knowledge, is represented as a concept-based semantic network. Expert System's semantic network, called Sensigrafo¹⁴, is a rich conceptual representation of the language containing more than 400,000 concepts and millions of links between these concepts. Once an entity is identified by linguistic and semantic analysis, the plugin returns its unique identifier from the ENS and injects it in an RDFa annotation inside the document. If the

¹⁴ Sensigrafo is a trademark of Expert System S.p.A., Italy.
<http://www.expertsystem.it/>

ENS does not contain a record about the entity, a new identifier can be generated. *Okkam4Blogger* is a web plugin for the free blog application Blogger¹⁵ from Google.

4.2 For the Linked Data Community

The goal of the Linked Open Data¹⁶ community is to connect several openly available datasets on the Web. The initiative is closely related to the Semantic Web community, indeed many of the datasets are published in the form of RDF/OWL. A link among two URLs shall be interpreted as a `owl:sameAs` statement. Identifying such links is a very work-intensive task and typically requires data-source specific heuristics¹⁷.

The ENS can help the Linked Data community in particular to store and represent the links, as they correspond to equivalent entities. The ENS implements a functionality exposed through the method `getAlternativeIDs()` (accessible via Web services), which allows someone working with Linked Data to retrieve all the alternative identifiers known to the ENS for the same entity¹⁸.

5 Evaluation

In this section we report about some of our evaluation experiments, which support that the ENS can cope with large entity profile collections, show scalable behavior and high search quality.

5.1 Scalability and Performance

For our scalability experiments we used a large dataset, which we syntactically generated based on the US Census statistics¹⁹. The dataset consists of 100 million entities and has a size of ca. 300GB. We queried the system using concurrent clients. For the performance and scalability evaluation we used a synthetic query set. Figure 3 depicts the average query throughput figures we measured with a population of 1, 10 and 100 million entities, with 1-100 clients. For this measurements we used the following configuration: 1 Index server with 8 cores (2.66GHz), 16GB RAM, 1TB disk, Solr 1.4, Tomcat 6, and 16 HBase nodes with 8 cores (2.66GHz), 8GB RAM, 2x1TB disks, HBase 0.20.2.

5.2 Search Quality

We evaluated the quality and performance of entity resolution on different kinds of queries over a storage containing 6.5 million entities, include people, organizations, and locations. We aimed at using queries covering a wide user and

¹⁵ <http://www.blogger.com/>

¹⁶ <http://linkeddata.org/>

¹⁷ <http://esw.w3.org/topic/TaskForces/CommunityProjects/LinkingOpenData/EquivalenceMining>

¹⁸ This function is inspired by what has been called “coreference service” by Glaser et al. in [15].

¹⁹ <http://www.census.gov/genealogy/names/>

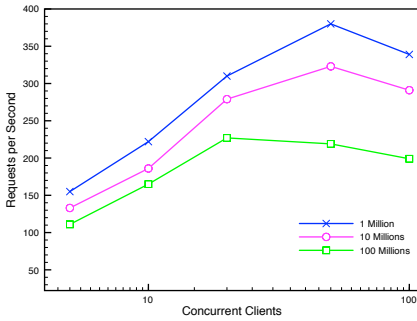


Fig. 3. Average query throughput

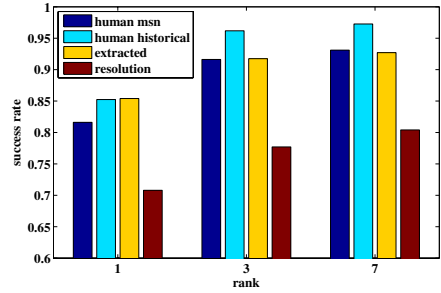


Fig. 4. Okkam entity search success rates for various query sets

applications range. For this reason, we collected query sets from four different sources. The first query set, named 'human msn', contains 610 queries from the MSN Search query Log (RFP 2006 dataset)²⁰. These queries are quite short in length that user provided as input in web search engines, e.g., “whitney houston”, and “Bloody Mary of England”. The next set, named 'human historical', are queries created from text that people included on web pages, for example to describe members of organizations, e.g., “Patrick de Gayardon French skydiver skysurfing pioneer”. Our third set, named 'extracted' contains 315 queries created from data extracted from web blogs and news articles. We used two extraction tools, the OpenCalais, and the Expert System's²¹ Cogito. Examples of the resulted queries are name=“Hillary R. Clinton”, and name=“Barack Obama”. The last set, named “resolution”, has 1000 queries created from structured DB-Pedia data. In contrast to the previous sets, these queries provide much information for matching while also providing the attribute values. For all queries, we manually or automatically retrieved the Okkam identifier. Queries are available upon request.

Figure 4 shows the success rate at different ranks, which measures the performance of Okkam for returning the requested entity in the rank position, i.e., as the first answer, third, and seventh. Except for resolution queries, we observe that the success rate stabilizes relatively quickly around rank 3 at values above 90%. Resolution queries show lower success rates, even though 80% at rank 7 is acceptable. The difference between resolution and extracted queries is that resolution queries tend to be over-specified, and a wider range of attribute label are used than in extracted queries. It is interesting to note that for those structured queries over-specification performs worse than under-specification, whereas for human keyword queries the inverse is true. This might indicate a problem with the attribute label similarity. Schema matching techniques will be investigated to correct this. The average time for processing queries is below 0.5 seconds, which means that Okkam can process up to 120 queries per minute.

²⁰ <http://research.microsoft.com/en-us/um/people/nickcr/WSCD09/>

²¹ <http://www.expertsystem.net>

6 Related Work

This paper presents the overall ENS and demonstrates through experiments, that the system can cope with large entity profile collections and high number of users. Thus, together with the OKKAM-empowered tools it can contribute to realize entity oriented information systems. The conceptual model for reusing OKKAM identifiers on the Web is discussed in [5]. The paper [7] presents an early prototype of the ENS, the system and the corresponding ecosystem we are presenting in this paper has significantly improved since these early versions. The paper [6] focuses on business-oriented use cases, scenarios and application prototypes that might benefit from the use of the ENS.

OpenCalais [22] is a popular plugin for Web browsers. Using the plugin, the identified named entities are highlighted on the Web page. The entities have an identifier, however this might be not globally unique. The entity descriptions (if they exist) are kept private and the Web users are not allowed to enter or edit their own entities. On the contrary, the ENS provides unique identifiers and also one can edit the entity profiles, under some access restrictions.

The zemanta [30] plugin, which is available for popular email and blog software, empowers their users to automatically associate the text they are writing with resources on the web, such as pictures, wikipedia links, etc. As far as we know, they do not use globally unique identifiers.

Bautin et al. [1] propose an entity search engine. They construct concordance documents for each entity consisting of all sentences, which contain references to the entity from their large text corpus. They do not provide tools or the possibility of editing entity profiles (concordance documents) and also they do not use globally unique identifiers. On the other hand they try to discover and depict the relations among entities, which the ENS does not support in this version.

OpenID [23] uses unique identifiers, but for a very different goal: they try to realize that users can identify themselves at different services with the same id and password.

7 Conclusion and Future Work

We implemented an enabling infrastructure, the Entity Name System, which opens the possibility to realize the web of entities. Our scalability and search quality experiments suggest that the ENS is able to serve a larger user community. The ENS is accompanied by a number of tools, which both exploit the services of the ENS, thus improving application experiences for users through disambiguation, and offer the possibility of content creation. In this way the whole user community can benefit from the individual contributions, and shall result in a continuous entity population growth.

The ENS is currently used in several pilot projects including the semantic information mashup Sig.ma [22] and at the Italian new agency ANSA [23]. The unique

²² <http://sig.ma>

²³ <http://ansa.it>

identifiers provided by ENS are also utilized in a private setting to manage publication records at Elsevier²⁴ and to manage a large software product portfolio at SAP²⁵. The ENS is also used to link physical entities to electronic ones, through QR codes, in the city of Manor, Texas, US.

Acknowledgements

This work is partially supported by the by the FP7 EU Large-scale Integrating Project **OKKAM – Enabling a Web of Entities** (contract no. ICT-215032). More details are available here: <http://www.okkam.org>. The authors would like to express their gratitude to all members of the OKKAM consortium for their contributions.

References

1. Bautin, M., Skiena, S.: Concordance-Based Entity-Oriented Search. *Web Intelligence and Agent Systems* 7(4), 303–320 (2007)
2. Bazzanella, B., Chaudhry, J.A., Palpanas, T., Stoermer, H.: Towards a general entity representation model. In: *SWAP* (2008)
3. Benjelloun, O., Garcia-Molina, H., Menestrina, D., Su, Q., Whang, S.E., Widom, J.: Swoosh: a generic approach to entity resolution. *The VLDB Journal* 18(1), 255–276 (2009)
4. Bilenko, M., Mooney, R.J., Cohen, W.W., Ravikumar, P., Fienberg, S.E.: Adaptive name matching in information integration. *IEEE Intelligent Systems* 18(5) (September 2003)
5. Bouquet, P., Palpanas, T., Stoermer, H., Vignolo, M.: A Conceptual Model for a Web-Scale Entity Name System. In: Gómez-Pérez, A., Yu, Y., Ding, Y. (eds.) *ASWC 2009*. LNCS, vol. 5926, pp. 46–60. Springer, Heidelberg (2009)
6. Bouquet, P., Stoermer, H., Barczynski, W., Bocconi, S.: Entity-centric Semantic Interoperability. In: *Cases on Semantic Interoperability for Information Systems Integration: Practices and Applications*, pp. 1–21. IGI Global (2009)
7. Bouquet, P., Stoermer, H., Bazzanella, B.: An Entity Name System (ENS) for the Semantic Web. In: Bechhofer, S., Hauswirth, M., Hoffmann, J., Koubarakis, M. (eds.) *ESWC 2008*. LNCS, vol. 5021, pp. 258–272. Springer, Heidelberg (2008)
8. Dean, J.: Challenges in building large-scale information retrieval systems (invited talk). In: *Proceedings of the Second ACM International Conference on Web Search and Data Mining, WSDM* (2009)
9. Doan, A., Lu, Y., Lee, Y., Han, J.: Object matching for information integration: A profiler-based approach. In: *Proceedings of IJCAI 2003 Workshop on Information Integration on the Web (IIWeb 2003)*, pp. 53–58 (2003)
10. Dong, X., Halevy, A., Madhavan, J.: Reference reconciliation in complex information spaces. In: *SIGMOD*, pp. 85–96 (2005)
11. Dong, X., Halevy, A.Y.: Indexing dataspace. In: *SIGMOD*, pp. 43–54 (2007)
12. Elmagarmid, A.K., Ipeirotis, P.G., Verykios, V.S.: Duplicate Record Detection: A Survey. *IEEE Transactions on Knowledge and Data Engineering* 19(1), 1–16 (2007)

²⁴ <http://www.elsevier.com>

²⁵ <http://www.sap.com>

13. Hernández, M.A., Stolfo, S.J.: Real-world Data is Dirty: Data Cleansing and The Merge/Purge Problem. *Data Min. Knowl. Discov.* 2(1), 9–37 (1998)
14. Ioannou, E., Niedere, C., Nejd, W.: Probabilistic entity linkage for heterogeneous information spaces. In: Bellahsene, Z., Léonard, M. (eds.) *CAiSE 2008*. LNCS, vol. 5074, pp. 556–570. Springer, Heidelberg (2008)
15. Jaffri, A., Glaser, H., Millard, I.: URI Identity Management for Semantic Web Data Integration and Linkage. In: Meersman, R., Tari, Z., Herrero, P. (eds.) *OTM-WS 2007, Part II*. LNCS, vol. 4806, pp. 1125–1134. Springer, Heidelberg (2007)
16. Kilov, H.: *Business Specifications: The Key to Successful Software Engineering*. Prentice Hall PTR, Upper Saddle River (1998)
17. Koshutanski, H., Massacci, F.: A negotiation scheme for access rights establishment in autonomic communication. *Journal of Network and System Management* 15(1) (March 2007)
18. Liu, X., Stoermer, H., Bouquet, P., Wang, S.: Supporting the Reuse of Global Unique Identifiers for Individuals in OWL/RDF Knowledge Bases (demo paper). In: *Proceedings of the 6th European Semantic Web Conference on The Semantic Web: Research and Applications (ESWC)*. LNCS, vol. 5554, pp. 868–872. Springer, Heidelberg (2009)
19. Manerikar, N., Palpanas, T.: Frequent Items in Streaming Data An Experimental Evaluation of the State-of-the-Art. *Data Knowl. Eng.* 68(4), 415–430 (2009)
20. Manning, C.D., Raghavan, P., Schütze, H.: *Introduction to Information Retrieval*. Cambridge University Press, Cambridge (2008)
21. Morris, A., Velegarakis, Y., Bouquet, P.: Entity Identification on the Semantic Web. In: *Proceedings of the 5th Workshop on Semantic Web Applications and Perspectives, SWAP 2008* (2008)
22. OpenCalais, <http://www.opencalais.com/>
23. OpenID, <http://openid.net/>
24. Palpanas, T., Chaudhry, J.A., Andritsos, P., Velegarakis, Y.: Entity Data Management in OKKAM. In: *Proceedings of the 2008 19th International Conference on Database and Expert Systems Application (DEXA)*, pp. 729–733. IEEE, Los Alamitos (2008)
25. Ravana, S.D., Moffat, A.: Score aggregation techniques in retrieval experimentation. In: *ADC*, pp. 59–67 (2009)
26. Robertson, S.: On gmap: and other transformations. In: *CIKM 2006: Proceedings of the 15th ACM International Conference on Information and Knowledge Management*, pp. 78–83. ACM, New York (2006)
27. Sarawagi, S., Bhamidipaty, A.: Interactive deduplication using active learning. In: *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pp. 269–278 (2002)
28. Tantonio, F.I., Manerikar, N., Palpanas, T.: Efficiently Discovering Recent Frequent Items in Data Streams. In: Ludäscher, B., Mamoulis, N. (eds.) *SSDBM 2008*. LNCS, vol. 5069, pp. 222–239. Springer, Heidelberg (2008)
29. X.509. The directory: Public-key and attribute certificate frameworks, ITU-T Recommendation X.509:2005 | ISO/IEC 9594-8:2005 (2005)
30. Zemanta, <http://www.zemanta.com/>

Transformation-Based Framework for the Evaluation and Improvement of Database Schemas

Jonathan Lemaitre and Jean-Luc Hainaut

Laboratory of Database Application Engineering - PReCISE research Center

Faculty of Computer Science, University of Namur

Rue Grandgagnage 21 - B-5000 Namur, Belgium

{jle,jlh}@info.fundp.ac.be

<http://www.fundp.ac.be/precise>

Abstract. Data schemas are primary artefacts for the development and maintenance of data intensive software systems. As for the application code, one way to improve the quality of the models is to ensure that they comply with best design practices. In this paper, we redefine the process of schema quality evaluation as the identification of specific schema constructs and their comparison with best practices. We provide an overview of a framework based on the use of semantics-preserving transformations as a way to identify, compare and suggest improvement for the most significant best design practices. The validation and the automation of the framework are discussed and some clarifying examples are provided.

Keywords: Database schema, schema evaluation, schema improvement, schema transformation.

1 Context and Motivation

Modeling activities are increasingly important in software engineering. According to the Model Driven Engineering (MDE) paradigm, database schemas are considered first-class artefacts. Due to the increasing cost of software design flaws, early evaluation techniques, that are quite common in software engineering, also prove necessary in the domain of database schema design. The use of software metrics have long been adapted to schema evaluation. They provide a synthetic measure of the quality of a schema by comparing some of its global characteristics to those of a set of reference schemas. The authors believe that the next step in schema evaluation should be the precise identification of schema defects according to a set of commonly agreed *best practices* that are to ensure specific requirements such as expressiveness, maintainability, evolutivity, performance, etc. From these defects and their scaling against best practices, it should be possible to measure the quality of a schema and to suggest improvement.

Considering these best practices, the scope of the work described in this paper is the evaluation and the improvement of database schemas for both existing

and under development systems. It relies on the use of semantics preserving transformations and on the identification of schema structures that have been defined to express specific application domain fact types. This analysis is used for evaluating the schema by comparing its structural content to a reference frame and the requirements of the schema context. The improvement activity will modify the schema structure in order to increase its compliance with the context while preserving its semantics.

In section 2, we present the conceptual background of our work, specifically the abstraction/paradigm space, the representation of heterogeneous schemas and the concept of schema transformation. In section 3, we define the concepts at the basis of the framework, namely equivalence classes of constructs, best practices, schema context and construct ranking. In section 4, we illustrate the framework through an example of alternative representations of a common data schema construct. In section 5, we suggest a possible application of the framework for the evaluation and the improvement of schemas. In section 6, we provide some hints about the limits identified so far. Section 7 shows how the framework can alleviate the problem of quality framework validation through field case studies. Sections 8 (Related works) and section 9 (Conclusions) are as usual.

2 Background

In this section we briefly (re)define some basic concepts and techniques that will be used to build our framework.

2.1 Abstraction Levels, Modeling Paradigms and Semantics

Database engineering processes generally rely on a hierarchy of abstraction levels. Currently called *Model-Driven Engineering*, multi-level approaches have been described since the seventies, when the terms conceptual, logical and physical were coined by several authors in the database realm. A formal description of the information/data structures of a database, be it in construction or in use, is called a *schema*.

A schema is positioned at a certain level of abstraction, depending on the technical detail it provides. In addition, a schema is expressed in a specification language, based on a definite paradigm (table II). Entity-relationship (ER) with its many variants, UML class diagrams, relational, object-relational, XML, IMS, standard files and even *schema-less*, are some of them. The database community calls them *models* (e.g., the *relational model*), a term we will use in this paper. There is an agreement on which abstraction level a given paradigm best fits. For instance, the Entity-relationship model is as its best at the conceptual level while the object-relational model should be used at the logical level. This defines a two-dimension space in which an arbitrary schema can be located and evaluated.

Table 1. A part of the abstraction/paradigm space

| Abstraction | Paradigms |
|-------------|---|
| Conceptual | Historical ER, rich ER, Merise (Individual model), EER, UML class diagrams, OO, Bachman DS, NIAM, ORM, binary ER, etc. |
| Logical | Relational, object-relational, OO, hierarchical, network, shallow, standard files, XML DTD, UML class diagrams, XML Schema, binary ER, etc. |
| Physical | Oracle, DB2, SQL Server, PostgreSQL, MySQL, COBOL files, IMS, IDS2, IDMS, Caché, ZODB, ADO.NET, etc. |

A schema will be considered *suitable* if the constructs¹ it is made up of comply with the usual way of doing according to the dimensions that define its position in this space.

Though a paradigm is considered suitable at some, but generally not all, abstraction levels (e.g., binary ER), we sometimes observe some cultural border crossover when a construct of one paradigm is used at an unusual position, in a foreign paradigm or in a non standard abstraction level. We mention three examples. A foreign key, which typically is a relational construct, could be found in an ER schema to avoid spaghetti-like schemas, for instance to reference DATE, LANGUAGE or NOTE general-purpose entities from hundreds of places in the schema. Some developers, quite familiar with XML processing tend to build database conceptual schemas that closely resemble a collection of tree structures. Finally, many current databases result from the straightforward migration from an obsolete technology, so that their schemas exhibit structural idiosyncrasies inherited from this technology.

The position of a schema construct in this space must be further refined by considering whether the intention of the designer has been translated adequately. In other terms, given a construct C , does C best expresses its intended semantics? We will call this property *semantic expressiveness*. A simple example will clarify the point. In modern ER models there is a wide agreement on representing sub-categories $C_{11}, C_{12}, \dots, C_{1n}$ of a reference category C_1 in the application domain by an *is-a* relation between supertype C_1 and subtypes $C_{11}, C_{12}, \dots, C_{1n}$. However, there are other ways to express these subcategories, such as by distributing the supertype components among the subtype (downward inheritance), by integrating the subtype components within the supertype (upward inheritance) and by representing is-a relation by one-to-one relationship types (is-a materialization). Therefore, there are often several ways to translate the idea of sub-category, but some can be considered more expressive than others.

To summarize, a schema is positioned at a certain level of abstraction, is expressed in a certain paradigm and is intended to translate the intention of the designer. A construct C of this schema can be evaluated through three questions: Does C naturally belong to this paradigm? Does C *feel comfortable* (so to speak) at this abstraction level? Does it best translate the intention of the designer? Our

¹ A construct is a distinct part of a schema considered as a whole for the purpose of the discussion. Typically it is a data structure (entity type, foreign key, is-a relation, the set of columns of a table, index) or a constraint. We consider in this paper constructs whose structure can be defined formally in a generic way, that is, through a *pattern*.

research consists in converting the answers to these questions into a fine-grained evaluation of a schema and into opportunities for improvement.

2.2 Schema Expression: The GER Model

Considering the multi-dimensional framework described above, we must be able to express non standard schemas that do not meet the ideal rule *only one paradigm at only one abstraction level*. In addition, a transformation can move a construct across abstraction levels and paradigm boundaries. Therefore, we will base the evaluation framework on a large spectrum data model encompassing all the abstraction levels and paradigms, namely the Generic Entity-relationship model, GER in short [1]. The GER is an extended Entity-Relationship model including, among others, the concepts of entity type, domain, attribute, relationship type, method, inheritance, primary and foreign keys, index, as well as various constraints. It also serves as a generic pivot model between the major database paradigms. In fig. 1 we illustrate the graphical notation of the GER objects used afterward. The graphical notation also supports informal notes (yellow boxes), which will be used to provide information that is not expressed structurally. For instance, a foreign key expressed informally through a note will not be declared in SQL-DDL but will be implemented in the application code. Such constructs are called *implicit constructs* [2]. For instance, in fig. 4 schema (e) includes two explicit foreign keys IdA and IdB to A, denoted by keyword `ref`, while in schema (f), these attributes form implicit foreign keys, defined through two informally noted inclusion constraints. Among both foreign key specifications, the first one clearly is of better quality.

An *operational model* M , that is, a model that is actually used in the design environment, can be described by a *GER submodel*, comprising a subset of the GER constructs together with a set of assembly rules that valid schemas must satisfy. A *M-compliant* schema is a GER schema that includes only constructs allowed in M and that satisfies all the assembly rules of M .

2.3 Schema Transformation

In this paper, we will address the multiplicity of representations of a given concept. The most appropriate tool to study this phenomenon is the transformational framework according to which a construct C in a schema can be replaced with another construct C' in a way that preserves some characteristics of C . In particular, we are interested in *semantics preserving*, or *reversible*, transformations that produce constructs C' that model exactly the same application domain situations as C does. A transformation is reversible iff there exists a function g with inverse g' such that, for each valid instance c of C , $g(c)$ is a valid instance of C' and $c = g'(g(c))$. Provided we have at our disposal an appropriate set of reversible transformation operators, a fairly large collection of constructs equivalent to C can be generated. The interested reader is referred to reference [1] for a more detailed description of the transformational approach.

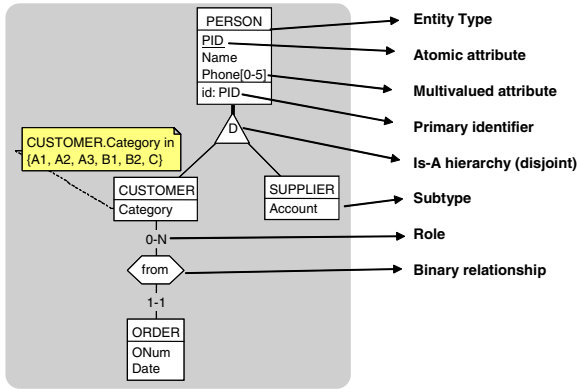


Fig. 1. Sample of GER schema at the conceptual level

3 Definitions

When a designer expresses an application domain fact type in a schema, s/he uses the data model construct that best fits its intention. The construct that most, if not all, skilled designers would choose in this situation is called a *best practice*². Best practices are defined considering the possible **alternative representations** and the **context** in which the schema is used.

3.1 Semantic Equivalence Classes and Best Practices

Semantic equivalence class of a construct. We consider K , the collection of all the constructs of the GER that are pertinent in some engineering processes and a set of transformations T . Let us also consider a construct C from K and all the equivalent constructs that can be derived through the reversible transformations of T . All these constructs, together with C , form an equivalence class called $ec(C)$. Since only reversible transformations have been applied, $\forall C' \in ec(C), ec(C') = ec(C)$. We now consider the function $sec : K \rightarrow (K \times 2^K)$ which associates to each construct in K its semantic equivalence class (sec). $sec(C)$ is an equivalence classes in which the specific element C has been tagged. We call C the *intention* of this equivalence class. $sec(C)$ provides all the constructs a designer can introduce in a schema to express the semantics (the application domain fact type) of C , hence the name *semantic equivalence class* or *sec*.

The concept of best practice. Let us consider a structure comprising a category A together with two of its sub-categories $A1$ and $A2$. It typically translates into an *is-a* relation construct. However, under certain circumstances, other equivalent constructs can be used instead as we have shown it in section 2.1. If we call C the *is-a* relation, $sec(C)$ includes all the constructs that express

² Intuitively, a best practice is a common practice among skilled designers.

category/sub-category structures, including C itself. C is the intention of its semantic equivalence class. For a seasoned designer, the *is-a* relation is the preferred translation but others can be used instead, though with a lower preference level. To better describe the notion of preference, we assign to each member C' of $sec(C)$ a preference score expressing the extent to which an expert designer will accept to use C' to express the semantic of C . The higher the score, the better C' will be to express this semantics. The preference score can be defined by a number or, more simply, by a partial order relation ($C''' < C'$ if C' is preferred to C''' to express the semantics of C). However, as we will discuss it later on, the preference scoring of sec is context dependent. We will call *best practices* of $sec(C)$ the constructs with the highest preference score. It must be noted that, depending on the context, C may not be the best practice in $sec(C)$.

Generation of SEC. The equivalence class of a construct C can be obtained by recursively applying the transformations of T until no new construct can be produced. However, this naive approach can lead to a very large (and, depending on T , possibly infinite) set of constructs of which only a small subset would be of interest. Appropriate meta-rules are necessary to keep the process into reasonable limits. Considering the *is-a* pattern, one can adopt a *regularity of treatment* meta-rule according to which each sub-category of a given category must be expressed in the same way. For example, a construct obtained by applying the upward inheritance transformation to one sub-category and the materialization transformation to another one would be rejected. Another example: when an entity type EA results from the transformation of an attribute A , the attribute(s) of the latter cannot be further transformed through the same transformation (figure 2).

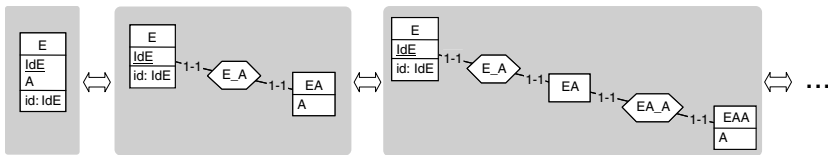


Fig. 2. Infinite transformation of an attribute

3.2 Context and Best Practices

The context of a schema S under evaluation is the external conditions that defines its intended use. S has been designed for the abstraction level A , according to the paradigm P and to meet design criterion D ³. We call (A, P, D) the *context* of S . Given a construct C that can appear in schema S , a scoring function is assigned to $sec(C)$ for a given context.

³ For simplicity, we consider that a schema is to meet one design criterion only.

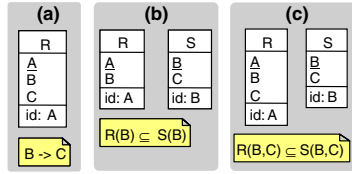


Fig. 3. (a) Relational schema with functional dependency. (b) Normalized schema. (c) Optimized schema.

Any model M is a point in the abstraction/paradigm space illustrated in table 1 (e.g., (*conceptual*, “*binaryER*”), (*logical*, “*SQL3*”)). The members of $sec(C)$ comply with the GER, but, being independent of any abstraction level and paradigm, they may not all comply with M . So, we introduce the concept of *projection* of a set of constructs on a model. Let us consider $K_M \subseteq K$ and model M , such that K_M is the set of constructs that are valid in M . K_M is the projection of K on M . The projection can be applied to semantic equivalence classes. We note $sec_M(C)$ the subset of constructs in sec that are M -compliant, i.e., $sec_M(C) = sec(C) \cap K_M$.

Now, we classify the members of $sec_M(C)$, ($\forall C \in K_M$) according to the evaluation criterion D . We assign (in a way that will be discussed later) each member of $sec_M(C)$ an order number, or, best, a numeric score in the range $[0 - 1]$. The first member(s) being the most appropriate according to D and the last one(s) the worst one(s). As we have said above, the intention can be, but need not be, the first member. For instance, in the sec of a foreign key at the conceptual level, the *one-to-many* relationship type will probably be assigned the highest score for $D = Expressivity$. Figure 3 describes a simple sec of unnormalized relation R for the SQL2 logical model. Considering access time optimization criterion, schema (c) can be assigned the highest score, since it is optimized for $R * S$ join while being easy to implement through a (non minimal) foreign key⁴. It is followed by schema (a), itself followed by schema (b). Of course, the *Normalization* criterion would have yielded quite different scores.

Considering construct C , the *bestpractice* for C in a given context (A, P, D) is the member (or members) of $sec(C)$ with the highest score.

4 Illustration

In this section, we illustrate the concepts of semantic equivalence class and schema context.

Figure 4 represents eight typical constructs that translate the category/subcategory structure where subcategories are pairwise disjoint. In these subschemas, **AttX** stands for a set of attributes of the entity type X ; **AttX[0-1]** means that all these attributes are optional; tag **id** in the 3rd compartment declared a

⁴ This pattern is known as the *Elementary Key Normal Form*.

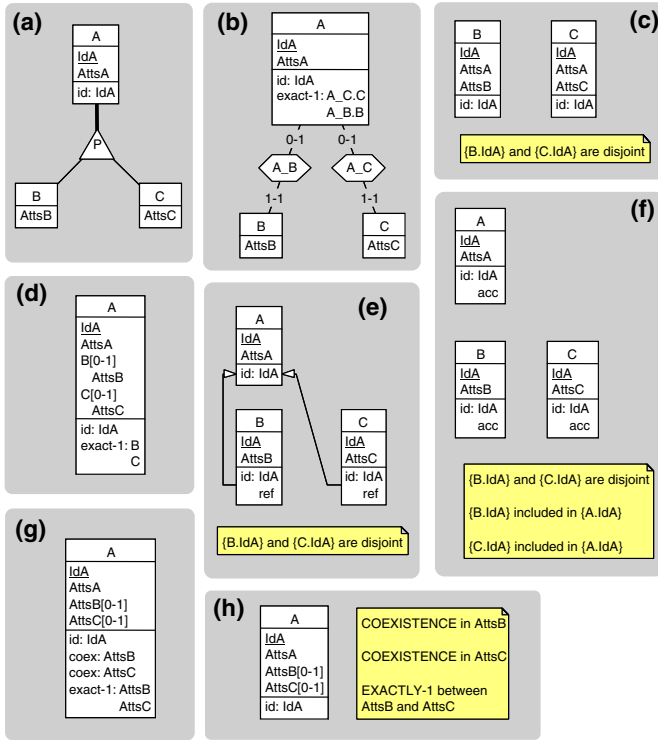


Fig. 4. SEC representing the category/subcategory with a partition constraint

primary key; `coex` means that the attributes must all be null or all not null; `exact-1` means that exactly one attribute must be not null; `ref` declared a foreign key. Since constructs (b) to (h) can be derived from construct (a) through semantics-preserving transformations, this set can be considered the *sec* of any of its members, in particular (a). The constructs (b), (c) and (d) are common alternative representations of *is – a* relations that are obtained respectively with the *materialization*, *downward inheritance* and *upward inheritance* transformations. (e) and (f) are derived from (b). In (e), the primary/foreign keys translate the relationship types of (b). In (f), the foreign keys are not declared but the referential constraints are expressed informally. (g) and (h) are obtained by the transformation of (d). In ER-like models, (b) will be considered the most expressive construct.

Let us now project this set of constructs on some point in the (A,P) space, namely: conceptual extended entity-relationship, logical IMS, logical relational (SQL2) and logical object-relational (SQL3). In table 2, we indicate the results of the projections on these four models. For convenience, we consider that models EER, SQL2 and SQL3 each include a language or a mechanism (OCL-like, check, triggers) to express `coex` and `exact-1` constraints.

Table 2. Suitability of the structures of fig. 4 in 4 projections

| | (a) | (b) | (c) | (d) | (e) | (f) | (g) | (h) |
|---------------------------|-----|-----|-----|-----|-----|-----|-----|-----|
| Conceptual EER | X | X | X | X | - | X | X | X |
| Logical IMS | - | X | X | - | - | X | - | X |
| Logical Relational | - | - | X | - | X | X | X | X |
| Logical Object-relational | X | - | X | - | X | X | X | X |

One can observe that nearly all structures of the *sec* are compatible with the EER model. The only exception is the foreign key. The IMS model is the most restrictive, as it imposes a strongly constrained tree-like structure between entity types (called *segment types* in IMS vocabulary). The relational model accepts five constructs. Its restrictions come from unsupported object types: relationship types, is-a relations and complex attributes. Finally the object-relational model is able to represent all *sec* constructs but relationship types and compound attributes.

We can classify the constructs of the four *sec* resulting from these projections according to a *fitness* criterion combining *expressivity* (specially for EER) and *ease of implementation* (specially for logical models). The goal of such ranking is to associate a score to the constructs, that will help for the evaluation of the schema and in the choice between alternatives. There are different ways to determine the scores. Among them, we are turning toward the collection of expert opinions. It aims at placing the structure on a quality scale. We apply the *fitness* criterion after a projection on the logical relational model. As a rough guide, we place the remaining structures in a 5 grades ordinal scale (very bad, bad, average, good and very good) as follow: **(h)** < **(f)** < **(c)** < **(g)** < **(e)**. Obviously, the quality rating can be based on a numerical scale, in order to provide a more precise classification.

5 Framework Application

An important issue in software engineering is the evaluation of the quality of a system. Software metrics have long been the favourite techniques to evaluate this quality. Though they are not always easy to interpret, metrics are easy to compute and they provide an immediate quantitative result [3,4,5]. However, when applied to database schemas, they are generally based on counting atomic objects such as entity types, tables, attributes, etc. independently of their intended meaning.

The approach described in this paper makes it possible to integrate semantic aspects in qualitative and quantitative evaluations. A qualitative evaluation requires a classification based of a simple order relation among the members of each *sec*. Such classification is easy to achieve. However it will neither provide a global score for the schema quality, nor allow one to quantify the difference between two schemas. It allows one to identify construct instances, that are not a *best practice* and that can be improved, together with the possible changes. The quantitative evaluation requires a classification based on a more or less precise numeric scale. The grading will obviously require more effort from experts.

Though defining metrics, their computation and their interpretation are beyond the scope of this paper, we will suggest two metrics to develop quantitative evaluations based on this framework.

The first suggested metrics is the **Individual sec evaluation**. It computes the average quality score of a schema S for a particular sec , defined by its intention I . We note $sec(I)$ the semantic equivalence class, M the model of S and D the criterion according to which $sec_M(I)$ has been classified. The metrics is defined as follows:

$$IndividualScore_{sec_M}(S, I) = \frac{\sum_{c \in sec_M(I)} \#\{inst(c, S)\} \times score_D(c)}{\sum_{c \in sec_M(I)} \#\{inst(c, S)\}}$$

where $inst(c, S)$ is the set of all instances of the construct c in the schema S and $score_D(c)$ is the score of c according to criterion D .

The second metric is the **Global sec evaluation**. It evaluates the average score of a schema in a particular context using the sec of the most significant constructs of M . We note II the set of these significant constructs. We associated a weight to each construct I (and its corresponding sec) in order to define their importance. We define normalized weights $weight(I)$ in the range $[0 - 1]$ so that their sum is equal to 1. The metric is defined as:

$$GlobalScore_M(S) = \sum_{I \in II} weight(I) \times IndividualScore_{sec_{(M)}}(S, I)$$

The evaluation of the schema through the identification of deviations from best practices naturally leads to its improvement according to a definite criterion D . The improvement process takes into account the classification of the sec of the most significant constructs (II) in order to maximize the schema quality. Though the specification of a complete improvement method is beyond the objective of this paper, we will sketch a tentative heuristics: for a specific $I \in II$, we transform each instance of I into the instance of $sec_M(I)$ that has the highest score. Figure 5 represents the sec of *is-a* relation expressions of fig. 4 projected on the SQL2 model, that is, $sec_{SQL2}(C_{is-a})$. Edges represent standard reversible transformations, acting as improvement paths, and scores translate numerically from 1 to 5 the *very bad* to *very good* scale of section 4. In some cases, the improvement path uses transformations that lead out of sec_M . In this case, the transformation chain has to form a path that goes back in the projected sec (e.g., transform (c) into (e)).

In the remainder of this section, we apply this approach on a small schema. In fig. 6, we represent 2 equivalent EER schemas. We identify in schema (a) two significant constructs which, considering the expressiveness criterion, are of poor quality (they have a low quality score in their respective sec). The optional attribute `DiscontinuedDate` is the date from which the product has been discontinued. Its sec contains, among others, a representation based on an *is-a* relation, suggesting that the attribute represents a specific category of product. We note C_{Att} the attribute construct and C_{isa} its *is-a* alternative. The construct comprising entity types `DETAIL`, `DET_ORD` and `DET_PRO`, together with their relationship types, is a complex but valid expression of a single many-to-many relationship type very common in some legacy IMS databases. We note C_{RT_s} this complex construct and C_{MM} the *many-to-many* relationship type.

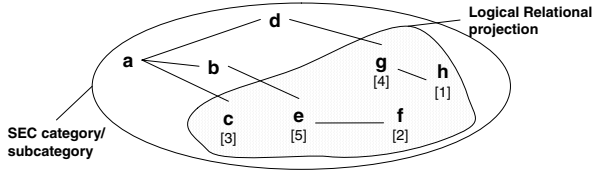


Fig. 5. Transformation associated to a semantic equivalence class

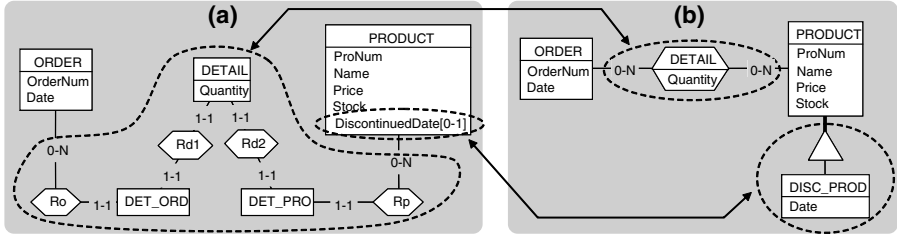


Fig. 6. Evaluation and improvement of a conceptual EER schema

For the purpose of the example, we consider the same 1 to 5 coarse-grained classification as above. The classification assigns the score 2 to C_{Att} and 5 to C_{isa} . C_{RTs} receives the score 1, while C_{MM} gets a 5. C_{isa} and C_{MM} are the only constructs in their *sec* with the maximum score. These scores allow to compute the *IndividualScore* metric. The results are trivial since they correspond to the score of the only construct of the *sec* present in the schema. As a rough guide, we assign C_{Att} and C_{RTs} respectively scores 0.4 and 0.6. The results are the following: $GlobalScore_{(EER, expressiveness)}(a) = 0.4 \times 2 + 0.6 \times 1 = 1.4$ and $GlobalScore_{(EER, expressiveness)}(b) = 0.4 \times 5 + 0.6 \times 5 = 5$. The expressiveness of (a) tends to be very bad while the score of (b) is maximum. The improvement process transforms (a) into (b).

6 Limits

The framework described in this paper is under evaluation through a collection of case studies. Though it is too early to draw definitive conclusions and to specify precisely its application domain, we have identified some issues that will be addressed in the near future. We will mention two of them.

Interactions are possible between the constructs of two distinct *sec*. We distinguish two types of interactions. In the first one, a construct appears in different SEC. In this case, an in-depth analysis has to be done in order to determine the very purpose of the structure. This can be performed using reverse engineering methods, and particularly the conceptualization step, through which the semantics of a technical construct is elicited. This implies that the identification of some *sec* construct instances may not be fully automated and requires human

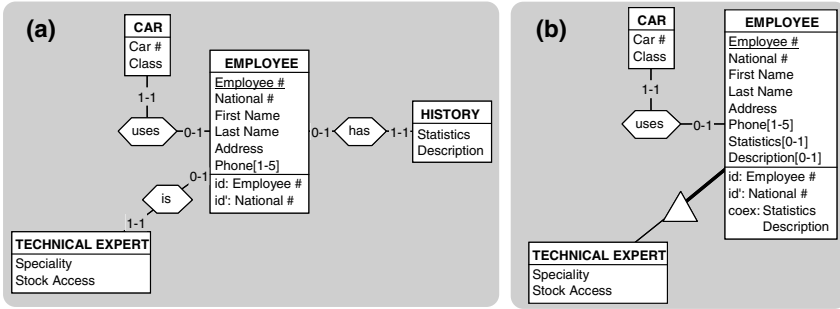


Fig. 7. Relations between constructs of different semantic equivalence classes

intervention. This problem is illustrated in the Fig. 7, where the 1-to-1 relationship type can be involved in an category/subcategory relation, a relation between two different concepts (employee-car) or a concept fragmentation.

The second type of interactions concerns structures that have a common part or structures included in others. We use the structure (a) in the fig. 4 in order to illustrate the problem. The attribute *Att*sA can be transformed into an entity type independently of entity type A and the is-a relation. Such transformation is used to extract a concept included in an other. The construct involved in this transformation should only take into account entity type A and its internal components (excluding the is-a relation) and is included in (a). As in the previous case, such problems may require some interactions with a designer.

As a second example of challenge already identified, it appears that addressing several criteria may lead to a conflicting situation. For example, the *minimality* criterion may contradict “expressivity”. *Normalization* versus *optimization* is another popular example.

7 Validation of the Framework

The validation and tuning of an evaluation framework is such a complex, costly and time-consuming task that it is seldom carried out according to standard validation methodologies. Ideally, one should collect dozens of real schemas of realistic size, that is, including at least 40-50 entity types or tables, and submit them to dozens of experts in data modeling who are asked to evaluate them according to various quality criteria such as expressivity, readability (for different stakeholders), maintainability, evolvability, time/space performance, DBMS-independence, etc. Considering that good experts are both scarce and very busy, proceeding in this way is quite unrealistic. As a consequence, many proposals are tested on closed systems comprising a teacher in IS design together with his/her students, a procedure sometimes considered unreliable.

The framework described in this paper makes it possible to lower the cost of the validation process.

1. First, the framework relies on the concept of *sec*, the generation of which can be, to a large extent, automated. However, the ordering of their members must be performed by experts. We can formulate two observations. (1) Though K can be an infinite set of patterns (depending on T), practice has shown that only a subset of about 20 constructs is sufficient to build most real schemas. (2) This provides us with about 20 *sec*, most of them comprising from 5 to 10 members. Ordering these sets of constructs is much easier and more deterministic than evaluating complete schemas. According to a first experiment involving four high level industrial experts⁵, assigning value scores to one *sec* according to one criterion (e.g. expressiveness) takes about 15 minutes and the results show very little variation among experts. Processing the most useful *sec* for one major criterion requires less than one day per expert, so that the complete parametrization of the framework can be performed in a matter of one or two weeks.
2. Evaluating a schema according to a set of ordered *sec* is an automatic task. Identifying constructs and their intention depends on the (still unknown) quality of the schema, so that a conceptualization step may be necessary. This process is automatic to a large extent⁶.
3. It remains to check the validity of the framework. Here, relying on teachers and students makes sense. (Last year) students form a realistic sample of designers of various skills, ranging from desperately inapt to experienced and ingenious. On the other hand, teachers are expected to be expert in evaluating the quality of medium size schemas. Therefore, comparing and aligning academic and automated evaluations allow the tuning of the evaluation framework. These validation and alignment processes still are under investigation.

8 Related Work

In the context of data schemas, very few authors seem to have explored the use of reversible transformations to deal with schema quality. Among them, Codd proposed the concept of relational normalization [6]. The normalization process relies on the use of transformations in order to eliminate problematic functional dependencies. Compared with our framework, it deals with a *no redundancies* criterion. An early synthesis of the existing normal forms was proposed by Kent [7]. Another proposal was made by Assenova and Johansson [8], who used reversible transformations to increase the understandability of the conceptual models, a criterion they decomposed into smaller quality criteria. In their work, a qualitative quality indicator is associated to transformations for each criterion. In our framework, we choose to relate the quality to the structure itself and develop more precise indicators. Burton and Weber [9] realized a study on the use of attributes in relationship types and its impact on schema clarity. Their observations were

⁵ From the ReveR company, specialized in database reengineering.

⁶ The DB-MAIN CASE tool we have been developing since 1993 includes programmable assistants that support this process.

based on equivalent schemas. A similar work was carried out by Gemino and Wand [10] on the use of mandatory properties and subtypes on ER schemas. Both proposals deal with schema quality and propose solutions to increase it, but none highlights the use of reversible transformations. Outside the context of data schemas, Bouhours et al. proposal focuses on the transformation of software architecture according to quality requirements [11]. Their transformations consist in applying design patterns that best satisfy the requirements (another name for design criteria). Finally, Kurtev [12] uses the concept of transformation space for dealing with schema quality. Such space represents a transformation by its initial and resulting structures and allows to link it with quality indicator. However, studied objects are atomic, while we consider semantically richer constructs.

9 Conclusion

The framework described in this paper intends to improve the precision and the automation of the evaluation of a database schema according to a definite criterion. Built on the transformational paradigm, it provides a sound and rigorous basis to develop evaluation strategies (including metrics-based ones) and improvement techniques. In particular, it makes explicit and implements the idea that a designer chooses, among a collection of candidate constructs (semantic equivalence class), which best fits its intention, that is, the fact type from the application domain. A defect in a schema occurs when this choice does not prove to be optimal. The framework makes it possible to identify this collection and the best choice, called *best practice*.

The framework also shows the importance of the three components of the context of a schema: the level of abstraction, the paradigm (that both form the data model) and the design criterion.

At the present time, we are parameterizing and validating the framework through practical case studies and with the help of a community of expert designers. We intend to compare the results of our framework with synthetic metrics-based approaches. In addition, we are developing a tool, built on the DB-MAIN platform, to identify significant patterns in a schema, to associate with each of them a quality score according to a definite criterion and to suggest improvement transformations.

References

1. Hainaut, J.L.: The transformational approach to database engineering. In: Lämmel, R., Saraiva, J., Visser, J. (eds.) GTTSE 2005. LNCS, vol. 4143, pp. 95–143. Springer, Heidelberg (2006)
2. Cleve, A., Lemaitre, J., Hainaut, J.L., Mouchet, C., Henrard, J.: The role of implicit schema constructs in data quality. In: Proc. of the International Workshop on Quality in Databases and Management of Uncertain Data, Auckland, New Zealand, pp. 33–40 (2008)

3. Genero, M., Piattini, M., Manso, M.E.: Finding "early" indicators of uml class diagrams understandability and modifiability. In: ISESE, pp. 207–216. IEEE Computer Society, Los Alamitos (2004)
4. Manso, M.E., Genero, M., Piattini, M.: No-redundant metrics for uml class diagram structural complexity. In: Eder, J., Missikoff, M. (eds.) CAiSE 2003. LNCS, vol. 2681, pp. 127–142. Springer, Heidelberg (2003)
5. Si-Said Cherfi, S., Akoka, J., Comyn-Wattiau, I.: Perceived vs. measured quality of conceptual schemas: An experimental comparison. In: ER (Tutorials, Posters, Panels & Industrial Contributions), pp. 185–190. Australian Computer Society (2007)
6. Codd, E.F.: Normalized data structure: A brief tutorial. In: SIGFIDET Workshop, pp. 1–17. ACM, New York (1971)
7. Kent, W.: A simple guide to five normal forms in relational database theory. *Commun. ACM* 26(2), 120–125 (1983)
8. Assenova, P., Johannesson, P.: Improving quality in conceptual modelling by the use of schema transformations. In: Thalheim, B. (ed.) ER 1996. LNCS, vol. 1157, pp. 277–291. Springer, Heidelberg (1996)
9. Burton-Jones, A., Weber, R.: Understanding relationships with attributes in entity-relationship diagrams. In: ICIS 1999: Proc. of the 20th international conference on Information Systems, Atlanta, GA, USA, pp. 214–228 (1999)
10. Gemino, A., Wand, Y.: Complexity and clarity in conceptual modeling: comparison of mandatory and optional properties. *Data Knowl. Eng.* 55(3), 301–326 (2005)
11. Bouhours, C., Leblanc, H., Percebois, C.: Alternative models for a design review activity. In: Proc. of the 2nd workshop on Quality in Modeling, Nashville, TN (USA), pp. 65–79. Springer, Heidelberg (2007)
12. Kurtev, I.: Adaptability of model transformations. PhD thesis, University of Twente, Enschede (2005)

Reverse Engineering User Interfaces for Interactive Database Conceptual Analysis

Ravi Ramdoyal¹, Anthony Cleve², and Jean-Luc Hainaut¹

¹ Laboratory of Database Application Engineering - PReCISE Research Center
Faculty of Computer Science, University of Namur, Belgium

{rra,jlh}@info.fundp.ac.be

² INRIA Lille-Nord Europe, LIFL CNRS UMR 8022

University of Lille 1, France

anthony.cleve@inria.fr

Abstract. The first step of most database design methodologies consists in eliciting part of the user requirements from various sources such as user interviews and corporate documents. These requirements formalize into a conceptual schema of the application domain, that has proved to be difficult to validate, especially since the visual representation of the ER model has shown understandability limitations from the end-users standpoint. In contrast, we claim that prototypical user interfaces can be used as a two-way channel to efficiently express, capture and validate data requirements. Considering these interfaces as a possibly populated physical view on the database to be developed, reverse engineering techniques can be applied to derive their underlying conceptual schema. We present an interactive tool-supported approach to derive data requirements from user interfaces. This approach, based on an intensive user involvement, addresses a significant subset of data requirements, especially when combined with other requirement elicitation techniques.

Keywords: Information systems engineering, Requirements engineering, Database engineering, Human-computer interfaces reverse engineering.

1 Introduction

Data modeling plays a pivotal role in Requirements engineering, as it defines the semantic core of the future application. Accurately eliciting and validating user requirements are vital to build reliable specifications of the data application domain. Database engineering precisely focuses on data modeling, where these requirements are typically expressed by means of a conceptual schema, which is an abstract view of the static objects of the application domain. Designing databases often relies on various requirements elicitation techniques such as the analysis of corporate documents and interviews of stakeholders. However beyond the initial collection of the requirements, these techniques usually do not actively and interactively involve end-users.

Still, the necessity to associate end-users of the future system with its specification and development steps has long been advocated [1]. End-users can perceive the qualities and the flaws of the information systems currently used, and since they know “how business is done” in the developing environment, they have the ability to state what could be done to improve it [2]. Involving them in the expression of their needs and in the definition of an appropriate solution can therefore reduce their resistance toward a new information system infrastructure and stimulate productivity [3]. As for the validation of these data requirements, their formal graphical representation is often difficult to grasp for the end-users. Indeed, while analysts focus on building requirements meeting various expectations (correctness, completeness, consistency, ...), requirements from the end-users standpoint need to be understandable and expressive enough.

In order to tackle this issue, we present an approach to elicit and validate database requirements, based on end-users involvement through interactive prototyping, and adapting techniques coming from various fields of study. By taking advantage of their expressiveness and understandability, we adopt form-based user interfaces as a two-way channel to efficiently express, capture and validate data requirements with end-users. In particular, we capitalize on the transformational power of data structure Reverse engineering techniques, which aim at extracting specifications from existing artifacts.

This approach relies on the principles of the ReQuest framework [4,5], which provides a complete methodology and a set of tools to deal with the analysis, development and maintenance of web applications. This approach proved that it is possible to efficiently and swiftly involve end-users in the definition of their needs. Whereas ReQuest deals with data modeling and the dynamic aspects of the future application (such as task analysis, behavior of the application, etc.), while providing generators for several of its components (database, framework skeleton, etc.), here we focus specifically on improving the data requirements process, leading the interfaces to appear as a means rather than an end product.

The remainder of the paper is structured as follows. Section 2 delineates the research context, while Section 3 describes the related works. The main principles of the proposal are detailed in Section 4. In Section 5, we elaborate on key aspects of our approach, namely Reverse engineering, modular refinement, view integration and transformational approach. Section 6 briefly presents the tool kit supporting the proposed methodology. Finally, in Section 7, we discuss the merits and limitations of our proposal and anticipate future work.

2 Research Context

The process of designing and implementing a database that has to meet specific user requirements has been described extensively in the literature [6] and has been available for several decades in CASE tools. It consists of four main sub-processes: (1) *Conceptual design* through which user requirements are translated into a conceptual schema; (2) *Logical design*, which produces an operational logical schema that translates the constructs of the conceptual schema according

to a specific technology family; (3) *Physical design*, which augments the logical schema with performance-oriented DBMS-specific constructs and parameters; (4) *Coding*, which translates the physical schema (and some other artifacts) into the DDL (*Data Definition Language*). Transformational and generative techniques allow one to automate the production of logical and physical counterparts of the conceptual schema [7], as well as artifacts of the final application [8], such as database code, program fragments, interface forms, etc.

In this paper, we focus on the primordial conceptual design, for which the Entity-Relationship (ER) model has long been the most popular medium to express conceptual requirements [9]. Its simplicity, its graphical representation, the availability of numerous CASE tools that include an ER schema editor (should) make it the ideal communication medium between designers and users. However, despite its merits, the ER formalism often fails to meet its objectives as an effective end-users communication medium. The reason is easy to grasp: a conceptual ER schema is actually a graphical presentation of a large and complex set of 1st and 2nd order predicates, and implicitly conveys non trivial concepts such as sets, non-1st normal form relations (NF²), algebraic operators, candidate keys and functional dependencies. Fig. 1(a) shows a small conceptual schema and its NF² relational interpretation according to the GER formalism [7]. The intrinsic complexity of the requirements has been concealed by the apparent intuitiveness of the ER graphical notation but has not disappeared.

On the other hand, most users are quite able to deal with complex data structures, provided they are organized according to familiar layouts. In particular, electronic forms have proved to be more natural and intuitive than usual conceptual formalisms to express data requirements [10], while making the semantics

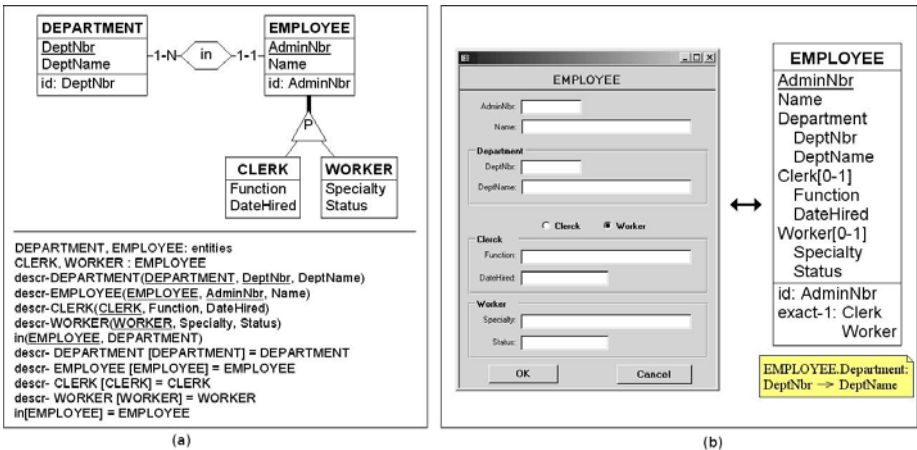


Fig. 1. (a) An ER schema and its formal expression. (b) An almost equivalent electronic form and its informational contents.

of the underlying data understandable [11]. This naturally leads to ponder the idea of using such forms as the preferred way to describe data structures in requirements engineering processes.

3 State of the Art

The strong link existing between graphical interfaces and data models is usually exploited in forward engineering. Indeed, as we have seen, it is relatively straightforward to produce artifacts such as form-based interfaces from a conceptual schema. Conversely, a form contains data structures that can be seen as a particular *view* of the conceptual schema. The transition from one to another has been shown to be tractable [12], so that database Reverse engineering techniques can be applied to recover a fragment of the conceptual schema. These techniques can be combined with prototyping, which may also act as a basis for interviews or group elicitation [13], while providing early feedback [14].

Deriving requirements from prototype artifacts has a long tradition. In 1984, Batini et al. studied paper forms as a widely used means to collect and communicate data in the office environment. Later on, Choobineh et al. [10] explored a form-based approach for database analysis and design, and developed an analyst-oriented Form Definition System and an Expert Database Design System that incrementally produced an ER diagram based on the analysis of a set of forms. Kösters et al. [15] introduced a requirements analysis method combining user interface and domain analysis. Rollinson and Roberts studied the problem of non-expert customization of database user interfaces in [12], and developed a set of graph-oriented transformations to extract an Extended Entity-Relationship schema describing an interface's information content. More recently, Yang et al. inquired about the WYSIWYG user-driven development of Data Driven Web Applications, while transparently generating their underlying application model on the fly [16]. We can observe that all these approaches rely on the same core principles: (1) build a set of form-based interfaces; (2) extract the underlying form model; (3) translate the form model into a working data model; (4) progressively build an integrated data model by looking for structural redundancies as well as constraints and dependencies.

We also notice that the number of studies on the subject is limited (especially recently), and that several limitations must be underlined in most of them. First of all, the tools provided for the drawing of the interfaces are not dedicated to this purpose and/or not convenient for end-users. Secondly, the underlying form model of the interfaces must often be constructed by analyzing the physical composition (layout) before the informational composition (content) of the form, and in parallel, the prototypical form-based interfaces do not use a generic language that would enable GUI generation of an application on any target platform. Regarding the coherence of the interfaces, it is assumed that the labels are used consistently through out the different forms, and little care is given to possible lexical variation (paronymy, feminine, plural, spelling, mistakes, etc.) and ontological ambiguity (polysemy, homography, synonymy). The use of examples

(either through static statements or dynamic interaction) is not systematically used to elicit constraints and dependencies. And last but not least, the final output is limited to the prototype itself and the final version of underlying integrated data model is not systematically submitted to the end-users in a way enabling easy validation.

4 Proposal

To alleviate understandability limitations of the ER model, we propose to use form-based user interfaces as a two-way channel to efficiently capture and validate static data requirements with end-users by providing the latter with adequate techniques to draw the interfaces describing the underlying key concepts of their application domain. This approach benefits from the advantages of rapid prototyping (such as visual expressiveness, early feedback and clarification) [17], while making the user a central actor of the process.

Our RAINBOW approach involves end-users in a simple and interactive way while providing the analysts with semi-automatic tools. The approach is formalized into a seven step process whose aim is to support the development of future applications and answer most of the concerns raised in Section 3. To illustrate these steps, let us consider the example of a small sales company wanting to develop a simple IT solution to manage its customers and their orders. Each order is created in a given shop and specifies a list of products.

1. *Represent*: End-users are invited to draw a set of form-based interfaces to perform usual tasks of their application domain. Such interfaces are typically entry forms to capture data on, say, a new customer or a new product. The end-users must at least provide basic properties regarding the interface elements (typically a label and description). Advanced users may also provide other properties such as the size of a field, the expected type of values, default or predefined values, existence constraints, as well as links between the concepts. Note that the objective is *not* to let end-users draw the interfaces of a future application¹, but to capture requirements through a medium they are familiar with. A dedicated drawing tool provides them with a limited set of primitive widgets, namely *interfaces*, *group boxes*, *tables*, *input fields*, *selection fields* and *button panels* (Fig. 2). These simple but usual form widgets can indeed be used to express any other complex widget. Fig. 3(a) illustrates the key concepts (**Customer**, **Order**, **Product**, **Shop**) that the end-users might draw for the running example.

2. *Adapt*: Once the interfaces are drawn, *database Reverse engineering techniques* are applied to recover the underlying conceptual schema of the domain. The interfaces (Fig. 3(a)) are automatically analyzed to extract data models using mapping rules, a subset of which is presented in Fig. 2. The stereotype <R> indicates an attribute associated to a button panel. Then, each individual entity type (Fig. 3(b)) is transformed into a primitive conceptual schema by transforming complex attributes (Fig. 3(c)). The structure of the data models is very simple, but, as we will discuss it, there is no semantic loss.

¹ Which is the case in the ReQuest framework.








| Widget | Visual Representation | ER Counterpart | Widget | Visual Representation | ER Counterpart |
|-----------|---|--------------------------------|-----------------|---|---|
| Interface |  | Entity Type | Input Field |  | Monovalued Mandatory Simple Attribute |
| Group box |  | Monovalued Compound Attribute | Selection Field |  | Monovalued Simple Attribute with Value Domain |
| Table |  | Multivalued Compound Attribute | Button Panel |  | Monovalued Role of a Relationship type |
| | | | Button Panel |  | Multivalued Role a Relationship type |

Fig. 2. Available graphical widgets with their mapping rules to data structures

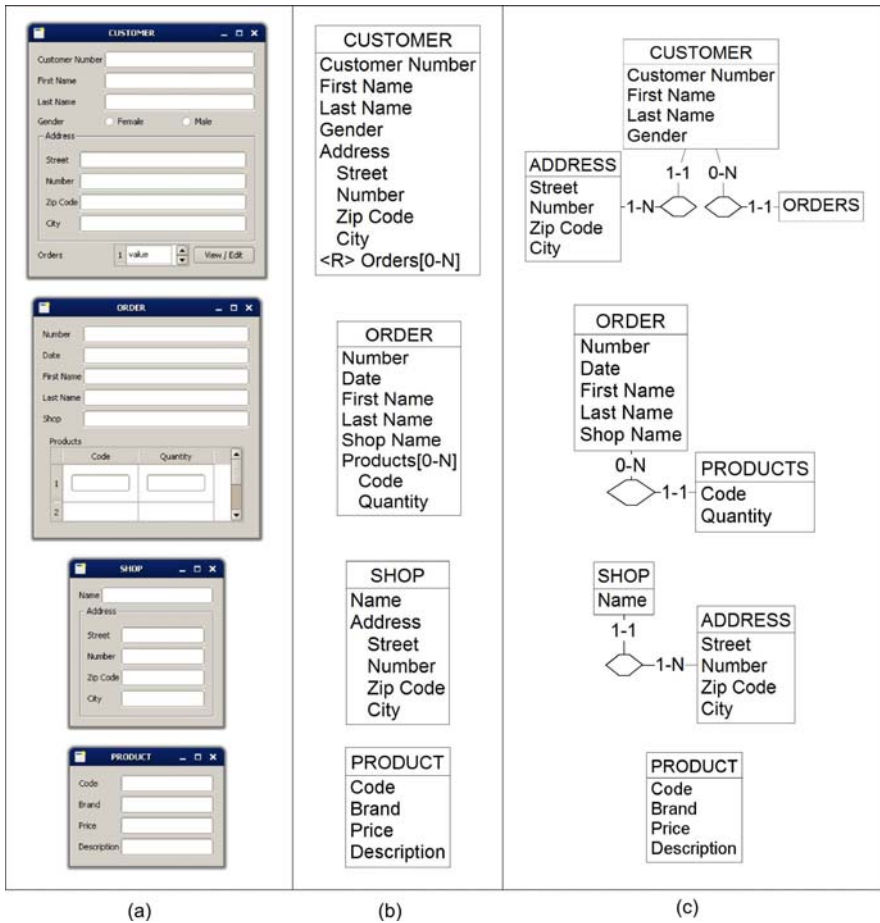


Fig. 3. (a) Example of user-drawn interfaces. (b) Translation of the interfaces into raw entity types. (c) Translation of the raw entity types into independent schemas.

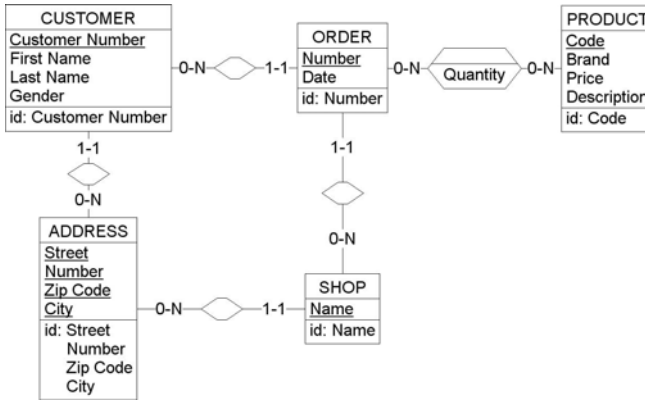


Fig. 4. Integrated schema of our running example

3. *Investigate*: Cross-analyzing each individual schema usually brings to light possible ambiguities as well as redundant information in the interfaces. In particular, semantic and syntactic redundancies are automatically identified and presented to end-users for manual validation.

4. *Nurture*: To elicit possible constraints and dependencies among data structures, induction techniques are applied on positive and negative data samples provided by end-users.

5. *Bind*: Validated redundancies, constraints and dependencies are processed to integrate the individual schemas into a pure conceptual schema that represents the data requirements. A dedicated approach has been developed, based on transformational techniques. Fig. 4 illustrates the result of the integration for our running example.

6. *Objectify*: A lightweight prototype application is generated from the integrated conceptual schema. It comprises a simple data manager that uses the interfaces drawn by the end-users and allows them to manipulate the concepts that have been expressed, typically to inspect, create, modify and remove data.

7. *Wander*: Finally, the end-users are invited to “play” with the prototype in order to ultimately validate the requirements, or identify remaining flaws.

5 Methodological Specificities

In this section, we describe in more detail four basic techniques at the core of the RAINBOW approach, namely Reverse engineering, Modular refinement, View integration and Transformational approach. We will show how the context in which they are used has lead to specialize them.

5.1 Reverse Engineering

Reverse engineering consists, among other things, in recovering or reconstructing the functional specifications from a piece of software, starting mainly from the

source code of the programs [18,19]. Such a process is typically required when an existing database has to be refactored or migrated toward a different technology. Reverse engineering hence aims at recovering a conceptual schema that is the most faithful to the original one, working from multiple system artifacts, such as documentation (when available), the DDL code of the database, data instances, screens, reports and forms, source code of application programs.

However, our objective is here to “build the truth” rather than “find the truth”, as in traditional Reverse engineering situations. In particular, the interfaces are used as a specification language as opposed to the usual Reverse engineering of existing screens. This requires to significantly adapt the usual database Reverse engineering (DBRE) methodology [20]. Indeed, as depicted in Figure 5 (a), DBRE typically comprises the following four sub-processes: (1) *Physical extraction*, which consists in parsing the DDL code in order to extract the raw physical schema of the database; (2) *Refinement*, which enriches the raw physical schema with additional constructs and constraints elicited through the analysis of the application programs and other sources; (3) *Cleaning*, which removes the physical constructs (such as indexes) for producing the logical schema; (4) *Conceptualization*, which aims at deriving the conceptual schema that the logical schema implements.

Such a methodology is not applicable as is in the context of the RAINBOW approach, as shown in Figure 5 (b). Starting from a set of user interfaces

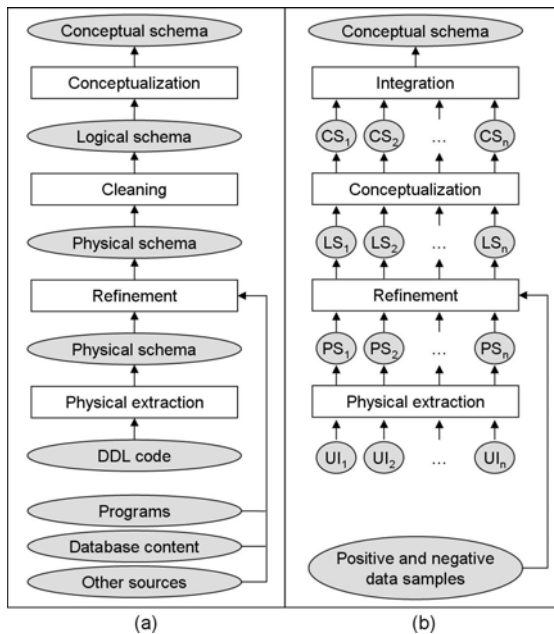


Fig. 5. (a) Standard database Reverse engineering methodology. (b) Reverse engineering methodology of the RAINBOW approach.

$(UI_1, UI_2, \dots, UI_N)$, the physical extraction does not allow one to derive a complete physical schema, but a *set of partial views* of this schema $(PS_1, PS_2, \dots, PS_N)$. Similarly, the refinement process may not rely on additional available artifacts such as application programs or database contents. However, it can take benefit from data samples provided by the users through the interfaces they have drawn, leading to the identification, among others, of candidate dependency constraints and attribute domains. The recovered constraints, once validated, are used to enrich the physical schemas PS_i in order to obtain a set of logical schemas LS_i . The cleaning phase, as defined above, does not make sense in the absence of an initial DDL code. Instead, the conceptualization step allows one to derive a set of partial conceptual schemas (CS_i) from the logical schemas obtained so far. In particular, the logical schemas are *normalized* in order to ease the identification of similarities between them. This important process relies on transformation techniques that will be developed in Section 5.4. During the integration phase, the partial conceptual schemas are merged, based on structural and semantic similarity criteria. This process, further described in Section 5.3, produces a single complete conceptual schema.

5.2 Modular Refinement

One of the key assets of our approach is its flexibility, especially regarding the enrichment of the data models. As we have seen, proficient end-users can already provide technical properties (such as size and type of attributes, domain of values, existence constraints, reference keys, ...) during the drawing phase. For non expert end-users, such properties can be discovered later on, as well as functional dependencies, from a set of positive and negative data samples [21,22] provided by end-users. Several algorithms have been proposed to support such a task [23]. However, a major limitation of these techniques comes from the fact that they rely on massive preexisting data sets, whereas we are working in a context where there is potentially no pre-existing data (or the (re)encoding cost would be too high). We therefore use a simplified version of such a mining algorithm to build Armstrong relations [24], by first asking end-users to provide data samples through the interfaces they have drawn. From these data, candidate constraints and dependencies are identified. Subsequently, using additional data samples and generated examples that the end-users may validate or reject, the constraints and dependencies are progressively enforced, discarded or refined for each entity type.

5.3 View Integration

In this section, we focus on the non standard integration process of our method. First of all, elements of integration (ambiguities, similarities, ...) are gradually collected, in order to be resolve later on.

The first element that retains our attention is the *semantic redundancy and ambiguity*. This issue arises due to the previously mentioned limitations of written natural language and possible mistakes, which lead to unclear labels in the

interfaces. In our example, one can for instance notice the closeness of the labels **Orders** (from the interface **Customer**) and **Order**. Moreover, consider for instance the strings “primary provider” and “alternative supplier”, which could be used as labels. They are not lexically close, but one may notice the nearness of meaning of the words “provider” and “supplier”.

Identifying similar labels and words is a well-known problem that can be dealt with using *String Metrics* [25], ontologies and similarity dictionaries. First of all, we extract the set W_i of relevant words contained in each label l_i of the interfaces elements, by typically casting away articles and conjunctions. Then, by combining these techniques, we identify ambiguously labeled elements within user-drawn interfaces.

For a given string distance metric s and threshold t , we declare two labels l_1 and l_2 as *lexically similar* iff :

$$s(l_1, l_2) \leq t \vee \exists w_{1_i} \in W_1, w_{2_i} \in W_2 : s(w_{1_i}, w_{2_i}) \leq t$$

Within a set of labels $L = \{l_1, l_2, \dots, l_n\}$, we also define a *subset of lexically similar labels* L_i as a subset $\{l_{i_1}, l_{i_2}, \dots, l_{i_m}\} \subseteq L$ verifying:

$$\begin{aligned} & \forall l_{i_j} \in L_i, \exists l_{i_k} \in L_i : “l_{i_j} \text{ and } l_{i_k} \text{ are lexically similar}” \\ & \wedge \forall l_{i_j} \in L_i, l_k \in L \setminus L_i : “l_{i_j} \text{ and } l_k \text{ are not lexically similar}” \end{aligned}$$

Among the wide variety of reliable string distance metrics, Jaro-Winkler’s distance d_w [26] has proved to be a good fit for short strings. It uses a prefix scale which gives more favorable ratings to strings that match from the beginning for a set prefix length. We use a variant s_w of this distance, also taking the longest suffix into account. We define it for two strings s_1 and s_2 as:

$$s_w(s_1, s_2) = 1 - \max(d_w(s_1, s_2), d_w(s_1', s_2'))$$

where s_i' is the reversed version of the string s_i

and $t_w = 0.2$ was found to be a reasonable threshold.

Since we want to partition the set L of all the labels available in the user-drawn interfaces into subsets of semantically similar labels, we confront each label to the others using s_w for string distance and the lexical reference system WordNet [27] for synonymy. We hence build the set of the lexically similar subsets, then visually point out the discovered similarities between concepts in the user-drawn interfaces in order to ask the end-users to validate or reject them.

Another issue concerns *structural redundancy*, which occurs when two entity types share a *pattern*. We define a pattern as a bijection between two sets of attributes belonging to different entity types. For instance, **Customer** and **Order** share a pattern based on the labels **First Name** and **Last Name**. The similarity between pairs of attributes from each set is measured using several indicators (e.g., the label). For each indicator, we define a *similarity index*, the values of which fall between 0 (strictly different) and 1 (strictly identical). The similarity of two attributes is computed as a weighted average of the similarity indicators.

Given the hierarchical structure of the interfaces, and thus the tree-like structure of the underlying models, the problem of extracting structural redundancies constitutes a particular case of *frequent embedded subtrees mining in rooted unordered trees*, which is similar to pattern mining in XML documents. This complex issue is described by Jimenez *et al.* [28], who also list related algorithms such as Zaki's SLEUTH or Asai *et al.*'s UNOT. In this context, entity types can be seen as root nodes, compound attributes as intermediary nodes, simple attributes as leaves, and the attribute order is irrelevant, as we explored in [5].

While tree-based approaches are suitable for complex and deep graphs, we observe that the structure of user-drawn interfaces is usually quite simple (no more than four levels of imbrication), if only by concerns of legibility and usability [10]. Instead of considering such heavy algorithms, we use a simple algorithm that consists in comparing one by one each entity type to elicit patterns and visually point them out. The end-users are then invited to arbitrate them by classifying the relation between the concepts sharing a pattern among one of these most usual cases: equality (the entity types represent the same concept), union (the entity types partially represent the same concept, which may translate the specialization of a higher-level concept non explicitly expressed), comprehension (one of the entity types is a specialization of the other), complementarity (one of the entity types actually refers to the other) or difference (the entity types fortuitously share a set of attributes).

Once the knowledge of the domain is enriched thanks to the end-user input, that validates or rejects ambiguities, similarities and dependencies, the integration process can take place. Transformational techniques have proved to be particularly powerful to carry out this task, which is a typical case of *database schema integration* [29]. They enable the integration of similar objects into a unique, non-redundant structure, without any loss of semantics. For simplicity, we consider in this paper that two similar objects refer to the same concept and can therefore be merged. Hence, the entity types of the logical schemas are integrated pairwise, and whenever needed, end-users are invited to choose which attributes of the two objects are relevant and should thus be kept during the merging process. However, as mentioned in the structural analysis, there is not always a strict identity between two concepts and other integration techniques must then be used to resolve integration conflicts [30]. At this point, it is clear that this process cannot be fully automated and that the analysts must be actively involved.

5.4 Transformational Approach

Finally, our approach heavily relies on the *transformational engineering* paradigm, according to which most (if not all) database engineering processes can be modeled as a chain of schema transformations [7]. A transformation operator is defined by a rewrite rule that substitutes a target schema construct for a source construct. The most interesting operators are said *semantics-preserving*, in that the source and target constructs convey the same semantics. Recall for instance Fig. 1. The schema on top of subfigure (a) is claimed to be a more

expressive but equivalent version of the schema on the right of subfigure (b), the latter representing the information contents of the electronic form on the left of subfigure (b). Two questions naturally arise: (1) how has the schema with entity types **Department**, **Employee**, **Clerk** and **Worker** been produced from the sole **Employee** aggregate and (2) what guarantee do we have that both structures are equivalent?

Fig. 6 illustrates two important operators in the context of user interface Reverse engineering, namely *attribute to entity type mutation* and *upward inheritance*. Both are *reversible* or *semantics preserving*, so that they can be applied from left to right and from right to left. The first one (T1) transforms an entity type into an equivalent attribute (and conversely). The second transformation (T2) integrates the subtypes of an entity type as complex attributes of the latter (and conversely). They lack some necessary pre- and post-conditions to be fully semantics preserving, but they are sufficient considering the scope of this paper.

When applied successively to the schema of Fig. 1(a), transformation T1 (on entity type **Department**) and transformation T2 (on subtypes **Clerk** and **Worker**) yield the schema of Fig. 1(b). Since the transformations are semantics-preserving, their inverse can be applied to the schema of Fig. 1(b), which is transformed in that of Fig. 1(a). This simple scenario illustrates the use of transformations in the Conceptualization process. It also shows how the expressive schemas of Fig. 3(c) can be extracted from raw physical schemas of Fig. 3(b) by a chain of semantics-preserving transformations.

Besides, ER schemas, be they conceptual or logical, can be given a formal semantics in several ways. Fig. 1(a) suggests an approach based on an extended NF² relational model. This semantics makes it easy to demonstrate the equivalence of two schemas by building a chain of algebraic operators, such as project/join and nest/unnest [7]. This formal framework is essential to evaluate the applicability of transformation in specific contexts, notably to identify missing parts in pre-conditions. For example, the functional dependency **DeptNbr** → **Location** in Fig. 1(b) is a part of the pre-condition to recover the **Department** entity type. Should this property be missing, the resulting schema would have been different (as suggested by Fig. 6(a)).

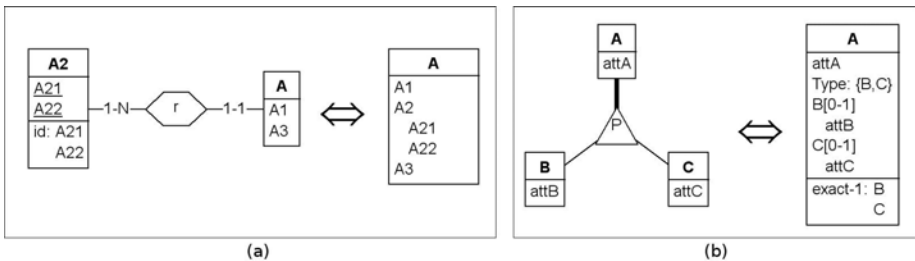


Fig. 6. T1 (a) and T2 (b), variants of two (almost) semantics-preserving transformations

6 Tool Support

The RAINBOW Tool Kit is a user-oriented development environment, intended to assist end-users and analysts in the definition and validation of database requirements through prototyping. It therefore supports the first steps of our approach by offering ready-to-use widgets and mapping rules with the ER model. The tool kit interacts with the repository of DB-Main, a database engineering CASE Tool [31] providing all the necessary functionalities to support a complete database design process (from conceptual analysis to DDL code generation). It provides transformation tools and supports database Reverse engineering. The interaction between these tools allows one to cover the whole database engineering process from both the end-user and the analyst perspectives.

7 Concluding Remarks

7.1 Contributions

This paper has presented a comprehensive interactive approach to bridge the gap between end-users and analysts during the requirements analysis phase of database engineering. This approach supports the elicitation and validation of static data requirements with end-users, while overcoming several limitations of existing prototyping methods. It relies on the expressiveness and understandability of form-based user interfaces, used jointly with tailored Reverse engineering techniques to acquire data specifications from existing artifacts. We offer a very simple interface model, inspired by high level abstract models such as UsiXML [32]. Since any interface widget dedicated to data representation can be expressed using these basic components, limiting the number of available widgets appears to ease the user interaction without restricting it. The process of drawing and specifying the interfaces takes in account possible labeling variations, and offers an incremental and flexible enrichment of the underlying data models.

The RAINBOW approach lies within broader perspectives, such as the Request Framework. While offering a precious user empowerment and involvement in the data requirements elicitation, it provides an expressive and interactive part of user requirements thanks to the RAINBOW Tool Kit and its connection to DB-Main. The requirements are materialized as a documented application domain (the conceptual model) and a documented database (DDL, queries, etc.). The simple yet operational generated prototype can serve as a basis for further application development over the database.

7.2 Limitations and Future Work

Although our approach addresses a significant subset of data requirements, it does not cover all of its aspects, typically the dynamic ones. Therefore, our approach does not replace more traditional task and information analysis approaches, but rather complements them. For instance, the form-based graphical representation of the underlying data model can be used during interviews to

stimulate the discussion. As for the generated prototype, it can be used during the task analysis to capture real-time use cases and define the expected behavior of the system. In addition, analyzing how the tasks are performed using the prototype in comparison to the legacy information system (if any), can help to support the Reverse engineering of existing artifacts and even induce more general considerations on the definition of the target information system.

This work assumes that end-users are able, with proper training, to represent simple and intuitive concepts. However, while limiting the approach to basic data structures can be seen as a “positive” simplification, we intend to push our investigation further to allow the users to also express complex structures such as temporal or semi-structured data.

A more general issue concerns the feasibility of involving different levels of users in our approach. A software engineering process may indeed involve stakeholders ranging from casual (or even novices) to experts users, such as analysts, designers or programmers, all of which may lack proper expertise in Database engineering. Still, to ensure the quality of the requirements, none of these users can be cast away, and each of them should be provided with adequate means to express their needs. We therefore intend to continue working on our approach and our tools, in order to make them even more intuitive and adaptative, according to the category of all our potential users.

Acknowledgments. This work was carried out during the tenure of an ERCIM “Alain Bensoussan” Fellowship. Partial support was also received from the Région Wallonne (through the ReQuest project) and the Interuniversity Attraction Poles Programme of the Belgian State, Belgian Science Policy (through the MoVES project).

References

1. Rosson, M.B., Carroll, J.M.: Usability Engineering: Scenario-Based Development of Human-Computer Interaction. Morgan Kaufmann, San Francisco (October 2001)
2. Fischer, G.: Beyond ‘couch potatoes’: From consumers to designers and active contributors. *First Monday* 7 (2002)
3. Vosburgh, J., Curtis, B., Wolverton, R., Albert, B., Malec, H., Hoben, S., Liu, Y.: Productivity factors and programming environments. In: ICSE, pp. 143–152 (1984)
4. Brogneaux, A.F., Ramdoyal, R., Vilz, J., Hainaut, J.L.: Deriving user-requirements from human-computer interfaces. In: Proc. of 23rd IASTED Int’l Conf., pp. 77–82 (2005)
5. Vilz, J., Brogneaux, A.F., Ramdoyal, R., Englebert, V., Hainaut, J.L.: Data conceptualisation for web-based data-centred application design. In: Dubois, E., Pohl, K. (eds.) CAiSE 2006. LNCS, vol. 4001, pp. 205–219. Springer, Heidelberg (2006)
6. Batini, C., Ceri, S., Navathe, S.B.: Conceptual database design: an Entity-relationship approach. Benjamin-Cummings Publishing Co., Inc. (1992)
7. Hainaut, J.L.: The transformational approach to database engineering. In: Lämmel, R., Saraiva, J., Visser, J. (eds.) GTTSE 2005. LNCS, vol. 4143, pp. 95–143. Springer, Heidelberg (2006)

8. Pizano, A., Shirota, Y., Iizawa, A.: Automatic generation of graphical user interfaces for interactive database applications. In: Proc. of the 2nd Int'l Conf. on Information and Knowledge Management (CIKM 1993), pp. 344–355. ACM, New York (1993)
9. Shoval, P., Shiran, S.: Entity-relationship and object-oriented data modeling—an experimental comparison of design quality. *Data Knowl. Eng.* 21(3), 297–315 (1997)
10. Choobineh, J., Mannino, M.V., Tseng, V.P.: A form-based approach for database analysis and design. *Communications of the ACM* 35(2), 108–120 (1992)
11. Terwilliger, J.F., Delcambre, L.M.L., Logan, J.: The user interface is the conceptual model. In: Embley, D.W., Olivé, A., Ram, S. (eds.) ER 2006. LNCS, vol. 4215, pp. 424–436. Springer, Heidelberg (2006)
12. Rollinson, S.R., Roberts, S.A.: Formalizing the informational content of database user interfaces. In: Ling, T.-W., Ram, S., Li Lee, M. (eds.) ER 1998. LNCS, vol. 1507, pp. 65–77. Springer, Heidelberg (1998)
13. Nuseibeh, B., Easterbrook, S.: Requirements engineering: a roadmap. In: Proc. of the Conf. on The Future of Software Engineering, pp. 35–46. ACM Press, New York (2000)
14. Davis, A.M.: Operational prototyping: A new development approach. *IEEE Softw.* 9(5), 70–78 (1992)
15. Kösters, G., Six, H.W., Voss, J.: Combined analysis of user interface and domain requirements. In: Proc. of the 2nd Int'l Conf. on Requirements Engineering, pp. 199–207. IEEE Computer Society, Los Alamitos (1996)
16. Yang, F., Gupta, N., Botev, C., Churchill, E.F., Levchenko, G., Shanmugasundaram, J.: WYSIWYG development of data driven web applications. *Proc. of the VLDB Endowment* 1(1), 163–175 (2008)
17. Ravid, A., Berry, D.M.: A method for extracting and stating software requirements that a user interface prototype contains. *Requir. Eng.* 5(4), 225–241 (2000)
18. Chikofsky, E.J., Cross, J.H.: Reverse engineering and design recovery: A taxonomy. *IEEE Software* 7(1), 13–17 (1990)
19. Hall, P.A.V.: *Software Reuse and Reverse Engineering in Practice*. Chapman & Hall, Ltd., Boca Raton (1992)
20. Hainaut, J.L.: Introduction to database reverse engineering. LIBD Publish (2002), <http://www.info.fundp.ac.be/~dbm/publication/2002/DBRE-2002.pdf>
21. Tseng, V.P., Mannino, M.V.: Inferring database requirements from examples in forms. In: Proc. of the 7th Int'l Conf. on ER Approach, pp. 391–405 (1988)
22. Ram, S.: Deriving functional dependencies from the entity-relationship model. *Commun. ACM* 38(9), 95–107 (1995)
23. Yao, H., Hamilton, H.J.: Mining functional dependencies from data. *Data Min. Knowl. Discov.* 16(2), 197–219 (2008)
24. De Marchi, F., Petit, J.M.: Semantic sampling of existing databases through informative armstrong databases. *Information Systems* 32(3), 446–457 (2007)
25. Cohen, W.W., Ravikumar, P., Fienberg, S.E.: A comparison of string distance metrics for name-matching tasks. In: Proc. of IJCAI 2003 Workshop on Information Integration on the Web (IIWeb 2003), pp. 73–78 (2003)
26. Winkler, W.E.: String comparator metrics and enhanced decision rules in the fellegi-sunter model of record linkage. In: Proc. of the Section on Survey Research Methods, pp. 472–477 (1990)

27. Fellbaum, C.: WordNet: An Electronic Lexical Database. MIT Press, Cambridge (1998)
28. Jiménez, A., Berzal, F., Cubero, J.C.: Mining induced and embedded subtrees in ordered, unordered, and partially-ordered trees. In: An, A., Matwin, S., Raś, Z.W., Ślęzak, D. (eds.) ISMIS 2008. LNCS (LNAI), vol. 4994, pp. 111–120. Springer, Heidelberg (2008)
29. Batini, C., Lenzerini, M., Navathe, S.B.: A comparative analysis of methodologies for database schema integration. *ACM Computing Surveys* 18(4), 323–364 (1986)
30. Spaccapietra, S., Parent, C., Dupont, Y.: Model independent assertions for integration of heterogeneous schemas. *The VLDB Journal* 1(1), 81–126 (1992)
31. DB-Main: The official website, <http://www.db-main.be>
32. Limbourg, Q., Vanderdonckt, J., Michotte, B., Bouillon, L., López-Jaquero, V.: Usixml: A language supporting multi-path development of user interfaces. In: *EHCI/DS-VIS*, pp. 200–220 (2004)

Towards Automated Inconsistency Handling in Design Models

Marcos Aurélio Almeida da Silva^{1,*}, Alix Mougénot¹,
Xavier Blanc¹, and Reda Bendraou¹

LIP6, UPMC Paris Universitatis, France

Abstract. The increasing adoption of MDE (Model Driven Engineering) favored the use of large models of different types. It turns out that when the modeled system gets larger, simply computing a list of inconsistencies (as provided by existing techniques for inconsistency handling) gets less and less effective when it comes to actually fixing them. In fact, the inconsistency handling task (i.e. deciding what needs to be done in order to restore consistency) remains largely manual. This work is a step towards its automatization. We propose a method for the generation of *repair plans* for an inconsistent model. In our approach, the depth of the explored search space is configurable in order to cope with the underlying combinatorial characteristic of this problem and to avoid overwhelming the designer with large plans that can not be fully checked before being applied.

1 Introduction

As an effect of the increasing adoption of MDE (Model Driven Engineering), large-scale industrial projects are currently being developed by hundreds of people and make use of several models instance of different meta-models (e.g. SysML, UML, Petri nets, architecture, work, business process) [1]. In such a context, it turns out that inconsistencies that may exist in models have been revealed as one of the main development problems [2] and that is why developing techniques for inconsistency management becomes so important.

A model is considered to be inconsistent if and only if it contains undesirable patterns, which are specified by the so called inconsistency rules [3]. Even if there are several variants of inconsistency rules such as well-formedness rules of [4], structural rules of [5], detection rules of [6], syntactic rules of [7], and inconsistency detection rules of [8], approaches that deal with detection of inconsistencies irremediably consists in browsing the model in order to detect undesirable patterns. However, as defined by [4], inconsistency management not only consists in the detection of inconsistencies but also in their handling. Indeed, once inconsistencies have been detected on models, they have to be resolved.

The work we present in this paper focuses on the *handling of inconsistencies*, which consists in automating the modification of a model in order to make

* This work was partly funded by ANR project MOVIDA Convention N° 2008 SEGI 011.

it consistent again. The first challenge we address, in order to make a model consistent, is to identify the potential changes that can be applied to fix detected inconsistencies. Regarding this challenge, Nentwich has clearly shown that, for any given set of inconsistencies, there exists an infinite number of possible ways of fixing it [9]. Therefore for efficiency reasons, this challenge is more related to narrowing the scope of the identification rather than to enumerate all possible resolutions.

The second challenge we address is to measure the impact of a given identified change. Indeed, as Mens has shown in [6], a change that fixes one inconsistency may introduce new ones and therefore will be counter productive. This second challenge consists then in filtering out non productive changes in order to keep only productive ones.

The last challenge we address consists in computing the execution order of a set of productive changes that do solve a set of inconsistencies and that do not introduce new ones. This last challenge consists in choosing an appropriate order of execution of the potential productive changes and is therefore a combinatorial challenge. To sum up, we argue that the *handling of inconsistencies* aims at providing what we call *repair plans*, which are sequences of concrete changes to be performed over a given model and that fix existing inconsistencies without introducing new ones.

In this paper we propose an approach for automatic generation of *repair plans*. Our main motivation is to assist the developers while they build their models. We argue that in such context being able to generate partial short plans very quickly is a desirable feature.

Our proposal is based on Praxis [8], our model inconsistency detection approach. In Praxis, the model is represented as the sequence of actions executed by the user in order to build it. The *repair plans* we propose to generate are then sequences of Praxis actions. The key contribution of our work is leveraging a search algorithm that is optimized to find the best plan (the shortest plan that fixes the bigger number of inconsistencies) by exploring a limited and configurable subset of all possible plans. Our approach has been prototyped into the Praxis environment that runs on top of the Eclipse EMF platform. It proposes *repair plans* to UML developers while they build their UML models.

This paper is organized as follows: in Section 2, we present a motivating example of the inconsistency handling task. In Section 3, we present the Praxis formalism. In Section 4, we present our approach for automated inconsistency handling. In Section 5, we present our prototype implementation of this approach and the runtime results in applying these techniques in a series of randomly generated UML2 models. In Section 7, we conclude this paper after the section 6 that presents the related works.

2 Motivating Example

This section illustrates the problem of inconsistency handling in an example. All models in this paper are instances of the subset of the UML2 meta-model [10] displayed in Figure 1.

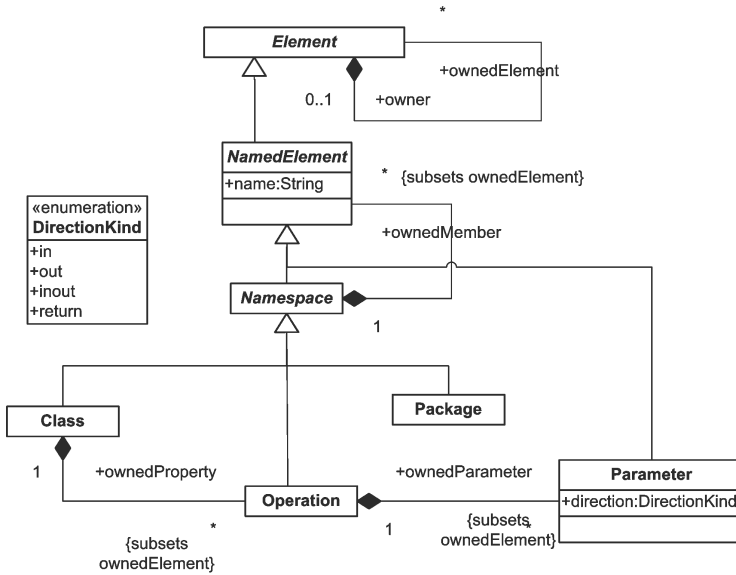


Fig. 1. A simplified fragment of the UML2 meta-model

The root concept of UML21 is the one of *model element* that is represented by the meta-class *Element*. Everything in a UML2 model is an instance of *Element*. An *Element* may contain a set of other elements. The *NamedElement* meta-class represents the elements that have a name (stored in the string attribute *name*). A *Namespace* is a named model element that serves as a space of names for the named elements it contains. Three classes inherit from *Namespace*: *Package*, *Class* and *Operation*. A *Package* is a container for other elements. A *Class* represents the object-oriented concept of a class of objects and serves as a namespace for its operations. An *Operation* represents an operation that is provided by the instances of a class and represents a namespace for its parameters. Every *Parameter* has a direction of one of four *kinds*: *in*, *out*, *inout* and *return*.

Figure 2 presents a UML model used as an example in this paper. This model contains a package called *Azureus* that owns two classes: *Client* and *Server*. The *Server* class defines an operation called *send()*. Figure 3 presents this model after some changes have been done on it in order to make it inconsistent. Those changes break the following well-formedness rule defined in the UML2 specification: “All the members of a *Namespace* are distinguishable within it”.

In the changed model, a new class named *Client* has been added. This creates an inconsistency since there is now two classes with the same name. In the rest of this paper, this first inconsistency is named **namespace-1**. In the changed model, a new operation named *send()* has been added in the class *Server*. This creates another inconsistency since there is now two operations with the same name. In the rest of this paper, this second inconsistency is named **namespace-2**.

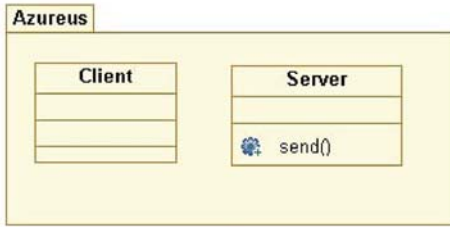


Fig. 2. Sample UML2 model

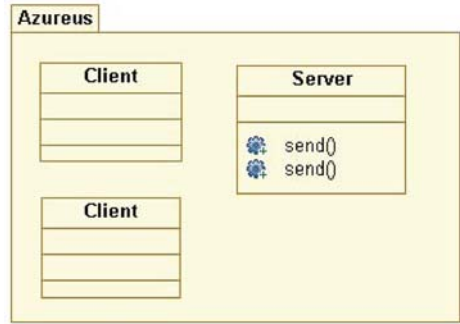


Fig. 3. Sample modified UML2 model

The first step in handling these inconsistencies is enumerating the repair changes that could solve the detected inconsistencies (i.e. deciding *how* to solve them). In this example, at least four sample repair changes¹ can be enumerated:

1. Rename one of the *Client* classes to *class1*.
2. Rename one of the *Client* classes to *Server*.
3. Rename one of the *send()* operations to *operation1()*.
4. Rename one of the *send()* operations to *operation2()*.

Each of the four repair changes fix some particular inconsistency by renaming one of the offending elements: repair changes 1 and 2 fix **namespace-1** and repair changes 3 and 4 fix **namespace-2**. The next step is thus to analyze *what* are the impacts of each repair change. Notice that the repair change 2 introduces a new inconsistency of the same kind by making the old class *Client* indistinguishable with the the old class *Server* (let us name this inconsistency **namespace-3**).

Finally, a analyze has to be done in order to decide *when* each of the repair change has to be executed, i.e. which change will be executed to fix each inconsistency and in which order. For example, if the repair changes 2 and 4 are executed in this order, this will solve both original inconsistencies, but will lead to a new one: **namespace-3**. This analyze will result that, for this sample, four possible repair plans can be executed: 1 – 3, 1 – 4, 3 – 1, and 4 – 1.

3 The Praxis Formalism

As described in [8], in the Praxis formalism, models are represented as sequences of elementary actions needed to construct each model element. Each action is

¹ Notice that these are just four among the infinite number of repair changes that actually solve one of the inconsistencies in the model. For instance, for every string *s* different from *Client* the plan “Rename one of the *Client* classes to *s*.” is a valid one.

```

create(p1,package,1)
addProperty(p1,name, 'Azureus',2)
create(c1,class,3)
addProperty(c1,name, 'Client',4)
create(c2,class,5)
addProperty(c2,name,'Server',6)
addReference(p1,ownedMember,c1,7)
addReference(p1,ownedMember,c2,8)
addReference(p1,ownedElement,c1,9)
addReference(p1,ownedElement,c2,10)
create(o1,operation,11)
addProperty(o1,name, 'send', 12)
addReference(c2, ownedProperty, o1, 13)
addReference(c2, ownedElement, o1, 14)

```

Fig. 4. Model construction operation sequence

annotated with a timestamp which indicates the moment when it was executed by the user. The actions are inspired by the MOF reflective API [11].

The $create(me, mc, t)$ and $delete(me, t)$ actions respectively create and delete a model element me , that is an instance of the meta-class mc at the timestamp t . The $addProperty(me, p, value, t)$ and $remProperty(me, p, value, t)$ add or remove the value $value$ to or from the property p of the model element me at timestamp t . Similarly, the actions $addReference(me, r, target, t)$ and $remReference(me, r, target, t)$ add or remove the model element $target$ to or from the reference r of the model element me at timestamp t .

Figure 4 presents the sequence of basic actions that construct the model presented in Figure 2. The action at timestamp 1 creates the package $p1$. The second action sets its name to *Azureus*. The actions with the timestamps 3 and 4 and those with the timestamps 5 and 6 create the classes $c1$ and $c2$ and set their names to *Client* and *Server* respectively. The actions with the timestamps 7 and 10 state that both classes are owned by the package $p1$. The actions with the timestamps 10 and 11 create the operation $o1$ whose name is *send* and the actions with timestamp 13 and 14 add it to the class $c2$.

Notice that this sequence is not unique, in the sense that there are usually many sequences that construct the same model. For example, changing the places of the actions with timestamps 7 and 10 and the actions with the timestamps 11 and 14 would result in the same model.

3.1 Inconsistency Detection in Praxis

In Praxis, inconsistencies are detected by the means of logic rules over the sequence of model construction actions that identify undesired patterns in it. The actions in the sequence are referenced by the means of logical predicates (expressed in Prolog in our prototype) that match them. For example, the Prolog

```

namespaceOCL1(ME1, ME2) :-
    lastAddReference(NS,ownedmember,ME1),
    lastAddReference(NS,ownedmember,ME2),
    ME1 \== ME2,
    not(distinguishable(ME2,ME1)).

```

Fig. 5. Inconsistency detection rule in Praxis

query `create(X, package, 1)` would result in the answer `X=p1` in the model represented in Figure 4.

As syntactic shortcuts, the ‘last’ prefix denotes actions that are not followed by other actions canceling their effects. Moreover, for each predicate referring to a Praxis operation, there is a similar one without the timespamp parameter. For instance, a `lastCreate(me, class)` operation is defined as a `create(me, class,t)` operation that is not followed by a `delete(me,t)` operation; and a `lastAddProperty(me, name, val)` operation is defined as a `addProperty(me, name, val,t)` operation for which the value of the *name* property of *me* in the model corresponds to *val*.

Figure 5 displays an example of inconsistency detection rule. This rule defines the predicate `namespaceOCL1(ME1, ME2)` that detects pairs of model elements *ME1* and *ME2* that are owned by the same namespace *NS* but are not distinguishable. The rules defining the predicate `distinguishable(ME1, ME2)` were omitted for the sake of brevity. Let us just consider that it holds if and only if *ME1* and *ME2* are instances of different meta-classes or if they are instances the same meta-classes but have different names.

4 An Approach for Inconsistency Handling in Praxis

In Praxis, it is worth to assume that model inconsistencies are *introduced* by user’s actions that violate some of its constraints or by *not executing* actions required by these constraints.

In order to generate a repair plan to fix these inconsistencies (either by undoing undesired actions or by adding omitted ones), our approach answers the three following questions (i) how to detect the actions that caused inconsistencies? (ii) how to enumerate the possible ways of changing a given inconsistent action? and (iii) how to generate a repair plan for the model sequence from the list of possible ways of changing the model?

4.1 How to Detect the Actions That Caused Inconsistencies?

Our approach to this problem is to adapt the inconsistency rules presented in Section 3.1 so that, instead of pointing out the problematic *elements* in the model they are going to identify the problematic *actions* which have (probably) *caused* the problem. This is done by the means of the *cause detection rules*.

```

namespaceOCL1(Cause) :-
    lastAddReference(NS,ownedmember,ME1, TS1),
    lastAddReference(NS,ownedmember,ME2, TS2),
    lastAddProperty(ME1, name, NM1, TS3),
    lastAddProperty(ME2, name, NM2, TS4),
    ME1 \== ME2,
    not(distinguishable(ME2,ME1)),

    Causes = [ addReference(NS,ownedmember,ME1, TS1),
                addReference(NS,ownedmember,ME2, TS2),
                addProperty(ME1, name, NM1, TS3),
                addProperty(ME2, name, NM2, TS4)          ],
    member(Cause, Causes).

```

Fig. 6. Cause detection rule in Praxis

Figure 6 presents the cause detection rule written from the inconsistency detection rule initially displayed in Figure 5. This new rule identifies four possible causes for the detected inconsistencies: the actions that added the model elements ME1 or ME2 to the same namespace NS or the actions that defined their names. Notice that, for this rule, the version of the Praxis predicates with explicit timestamps has been used because there is a need to identify unambiguously the action that caused the inconsistency.

In most cases a trivial *cause detection rule* can be generated from an inconsistency detection rule: just point all actions used to prove an inconsistency as possible causes of it. This is a possible strategy, although suboptimal, since not all involved actions are necessarily responsible for an incoherence. For example, let us take the following well-formedness rule from the UML2 metamodel: “*The visibility of all features owned by an interface must be public.*”

This rule is specified as the following cause detection rule:

```

interfaceOCL1(Cause) :-
    lastCreate(X,interface),
    lastAddReference(X,feature, F),
    lastAddProperty(F,visibility,V,TS),
    not(V='public'),
    Cause = addProperty(F,visibility,V, TS).

```

This rule states that if X is an interface and F is one of its features and its visibility is not `public` then the cause of this inconsistency is the action that defines its visibility. For each inconsistency of such kind that is proved, three actions need to be inspected: a *create*, an *addReference* and an *addProperty*; although only one of them is actually responsible for the inconsistency. Therefore, our manually written rule would prune two thirds of the search space, by avoiding looking for repair plans that fix actions that did not caused the problem.

4.2 How to Enumerate the Possible Ways of Changing a Given Inconsistent Action?

We propose to use generator functions that determine a set of lists of actions that cancel the effects of a given inconsistency causing action in the model. These manually written rules are mostly independent of the cause detection rules, since they do not have to consider impact of the changes they propose, neither to other inconsistencies that might exist in the model nor to the set of cause detection rules.

Let us analyze the following partial definition of the generator function:

```
generate(addProperty(E, name, OldName, TS),
         [remProperty(E,name, OldName),
          addProperty(E, name, NewName)]) :-
         lastCreate(E, C),
         randomNameGenerator(C, NewName).

generate(addProperty(E, visibility, 'public', TS),
         [remProperty(E, visibility, 'public'),
          addProperty(E, visibility, 'private')]).

generate(addProperty(E, visibility, OldVisibility, TS),
         [remProperty(E, visibility, OldVisibility),
          addProperty(E, visibility, 'public')])
         :- not(OldVisibility='public').
```

The first rule cancels inconsistencies in a `addProperty` action to the `name` field of a model element `E`. It says that every time a `addProperty(E, name, OldName, TS)` action is a source of inconsistency there is a simple plan that may fix it: removing the old value of the property by executing the action `remProperty(E,name, OldName)` and setting its name to another value `NewName` by the means of the action `addProperty(E, name, NewName)`.

Observe that this new name is generated using a random name generator accessible by the predicate `randomNameGenerator(C, Name)` such that it generates a name `Name` taking the meta-class `C` as a reference (e.g. generating a `metaclassX` name for a metaclass named `metaclass`).

The second and third rules fix inconsistencies in the `visibility` property of a model element `E`: if it was `public` the second rule suggests changing it to `private`, otherwise the third rule changes it to `public`.

As it is shown in [12], brute force generation of choices is not scalable, therefore, a well-written generator function needs to be custom-tailored in order to reduce the number of proposed choices while not introducing obvious inconsistencies (i.e. removing all values of a non-optional property).

4.3 How to Generate a Repair Plan for the Model Sequence?

The Iterative Deepening Depth-first Search Strategy (IDDFSS) [13] is a depth-first tree search algorithm that allows exploring the tree of possible repair plans

by taking into consideration the first suggested change for the first cause of inconsistency and going thus deeper and deeper in the search three until either a consistent model is found or until there is no possible action and thus backtracking.

The case when no action is possible happens when the algorithm reaches the maximum allowed depth in the search tree (i.e. when a previously defined maximum number n of execution steps has been reached, this is called a *depth-limited search*). This strategy is therefore capable of finding any repair plans that can be constructed after executing at most n steps. Indeed, if no complete repair plan exists in this limited search space, *partial* plans may also be constructed by recording the *best* (e.g. the one that generates a final model with less inconsistencies) plan found during the search.

In IDFSS, the maximum number of steps n is iteratively incremented, e.g. if the max number of steps is defined to be m , then a depth-limited search will be executed for every n between 1 and m until a repair plan that fixes all inconsistencies is found. This makes sure that the final plan was obtained with the least number of steps as possible.

On top of IDFSS we propose to adopt an heuristics that decides the next action to be fixed by ordering the inconsistency causes from the most recent to the least recent one. This design decision is based on the assumption that the user tries to maintain the model as consistent as possible at most of the time. Taken into account that inconsistencies are *introduced* in the model by actions executed later in the design process, it is reasonable to assume that the actions that were executed more recently are more likely to have introduced new inconsistencies that were not there before.

Finally, if there exists a repair plan that can be constructed with less than m execution steps the IDDFSS algorithm is guaranteed to find it. The remaining problem is thus determining the optimal value of m that is going to find a complete repair plan.

In fact, this problem is simpler then it seems at first sight since defining a precise value for m is *not necessary*. The value of m just needs to be set to be *big enough* to allow us to find a solution. Choosing a value that is *bigger* than needed has no impact on the actual execution time, since the depth-limited search will be repeated for all values from 1 to m until a complete solution is found, i.e. if a complete repair plan is found for some $n = k$ before reaching m the rest of the possible depths will not be tested.

4.4 Running Example

This section highlights our approach on the sample model displayed in Figure 3. Let us suppose that its model sequence is composed by the sequence displayed in Figure 4 appended with the following sequence:

```
create(c3, class, 15)
addReference(p1, ownedMember, c3, 16)
addReference(p1, ownedElement, c3, 17)
addProperty(c3, name, 'Client', 18)
```

```
create(o2, operation, 19)
addReference(c2, ownedProperty, o2, 20)
addReference(c2, ownedElement, o2, 21)
addProperty(o2, name, 'send', 22)
```

This sequence defines an inconsistent model, because `c2` and `c3` and `o1` and `o2` and indistinguishable in their respective namespaces. Let us detail the execution of one iteration of depth-limited search in the IDFS algorithm in which the depth of the search tree is limited to 2.

Step 1

At this point, the cause detection rules are used to compute the list of causes of inconsistencies in the model. The list is organized in the inverse order of timestamps:

```
addProperty(o2, name, send, 22)
addReference(c2, ownedmember, o2, 21)
addProperty(c3, name, 'Client', 18)
addReference(p1, ownedmember, c3, 17)
addReference(c2, ownedelement, o1, 14)
addProperty(o1, name, send, 12)
addReference(p1, ownedmember, c1, 7)
addProperty(c1, name, 'Client', 4)
```

The search tree is going to be explored in a depth-first search, this means that each step fixes the possible causes of inconsistencies from the most recent to the older one. If no consistent model was found after exploring one possibility, the next cause should be tried. At this point, the first cause of inconsistency (`addProperty(o2, name, send, 22)`) is taken. The generator function is then used to obtain a list of possible actions to fix it.

In this case, the generator function returns only one possibility composed of two actions:

```
remProperty(o2, name, send, 23)
addProperty(o2, name, operation1, 24)
```

Those actions are then appended to our model sequence.

Step 2

Like in the step 1, this step starts by computing the new ordered list of inconsistencies. The following list is then computed:

```
addProperty(c3, name, 'Client', 18)
addReference(p1, ownedmember, c3, 17)
addReference(p1, ownedmember, c1, 7)
addProperty(c1, name, 'Client', 4)
```

Then, this process is repeated by using the generator function in order to fix the first cause of inconsistency (`addProperty(c3, name, 'Client', 18)`) and getting the following list of actions:


```
remProperty(c3, name, 'Client', 25)
addProperty(c3, name, class1, 26)
```

At this point there are no more inconsistencies left on the model sequence so, the final repair plan is:

```
remProperty(o2, name, send, 23)
addProperty(o2, name, operation1, 24)
remProperty(c3, name, 'Client', 25)
addProperty(c3, name, class1, 26)
```

Two depth levels were explored in the search tree and it was enough to find a repair plan that fixed all inconsistencies in the model. If at the end of this limited exploration no solution is found, the execution would restart from scratch, but would explore a larger search space (e.g. a search limited to depth 3) and so on.

5 Prototype Implementation

In [8], we present the Praxis prototype. It is composed of two components: the *Sequence Builder* (which integrates to Eclipse EMF Framework and builds the model sequence from the actions executed by the user while creating a model) and the *Check Engine* (which is responsible for detecting inconsistencies).

This prototype has been extended in order to support the generation of repair plans. In particular, the *Model Fixing Agent* component has been integrated within Praxis. This component is an intelligent agent that proposes real time repair plans in order to fix the inconsistencies found in the model. The core functionalities of this component is entirely implemented in a set of Prolog rules that are packaged into an Eclipse Plug-in that interfaces with the existing Praxis plug-ins.

Figure 7 displays a screenshot of this integration. In (1) we show the class diagram presented in Figure 3 drawn using the Papyrus UML Tool integrated to Praxis. While the user is building the model, the Sequence Builder component in Praxis builds the sequence of actions. The Model Fixing Agent then watches this sequence and checks for inconsistent actions in it regularly. In (2) we see that it displays the list of inconsistencies found in the current model.

After detecting and showing the list of inconsistent actions, the Model Fixing Agent computes a repair plan for them. In (3) the four actions needed to repair the current model are listed in the order they need to be executed.

5.1 Case Study

Our approach has been stress tested with models that have been automatically generated by a mathematically grounded random model sampler [14]. These tests were executed with models of different sizes (varying from 20 to 10,000 model elements), and using different depths in the explored search space (1, 5, 10 and 15 levels) to show the impact of this parameter on the plan generation time

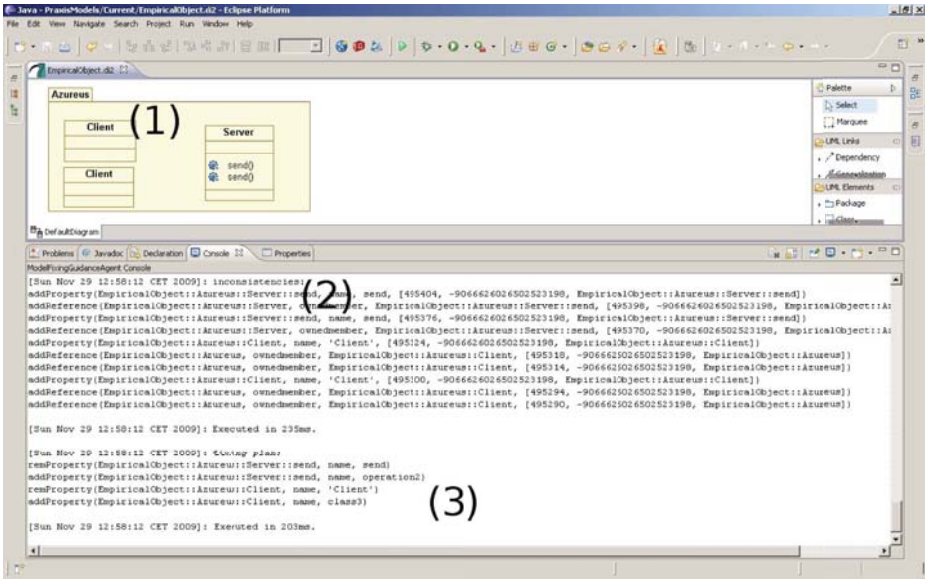


Fig. 7. Screenshot of integration with Praxis

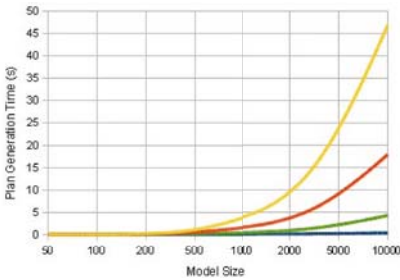


Fig. 8. Timing results in seconds

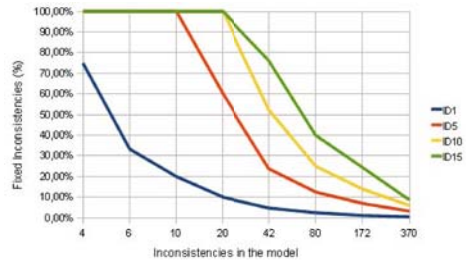


Fig. 9. Fixed inconsistencies results

and in the number of fixed inconsistencies. We manually implemented a subset of 4 of the UML2 well-formedness rules as cause detection rules and a set of 7 generator functions. Each test was executed 10 times and the average time was recorded.

Figure 8 shows the timing results of our tests. The graph clearly shows the exponential characteristic of our problem. Notice that the ID1 line (the results for exploring just one level in the search tree) is equal to current existing inconsistency detection approaches that are only capable of suggesting fixes for one inconsistency at a time.

Figure 9 shows the relative number of inconsistencies that is solved by our approach for each model for each maximum level of exploration. This graph

shows that the deeper is the level of the exploration the bigger is the proportion of solved inconsistencies. It also shows that for each level of exploration, there is a number of inconsistencies, such that, beyond that, only partial repair plans can be found.

6 Related Work

In [12], an approach for fixing inconsistencies in UML models is presented. In short, this approach uses a model profiler that monitors the parts of the model that were touched when consistency rules are evaluated. Those parts are then considered to be natural candidate for fixing actions when the rules are evaluated to be inconsistent. For each inconsistency, the approach explores all the possibilities of changing the monitored model parts in order to make the model consistent. The identification of possible changes are guided by the *choice generation functions*. The choices that turn the model into a consistent state are then presented to the user. Instead of trying to automatically identify the causes of the inconsistencies, our approach asks for their definition thanks to *Cause detection rule*. This definition may be automatically generated from existing inconsistency detecting rules, however, from our experience, it seems that it needs to be manually optimized afterwards. The choice generation functions of Egyed’s approach inspired our generator functions. However, instead of generating just options for changing the model graph, our generator functions deliver alternative fixing plans for given actions in the model sequence.

In [9], Nentwich et al describe a framework for repairing inconsistent documents in a distributed setting. Their approach consists of defining a mapping from the logical language used to describe the inconsistency rules into a set of *repair actions* that, after being executed, will make the model consistent again. In [6], Mens et al present an approach for inconsistency management on top of graph transformation tool AGG. They detect the inconsistencies in the model by the means of the inconsistency detection rules (that tags model elements as *conflicted*) and fix inconsistencies by the means of the *resolution rules* (for each possible resolution of every kind of inconsistency there is one rule that describe how should be the model after fixed). They use then a critical pair analysis algorithm to infer dependencies between rules and aid the user in the task of fixing the model. Both approaches automate the process of defining *how* to deal with the inconsistency by proposing a set of actions that fix each of them, and they automate the definition of *what* are the impacts of the suggested repairs: they detect inconsistencies among different plans and thus discourage their application at the same time. However, the decision on the *order* of the execution of the proposed repair actions (or plans) is left to the user. In our proposal, we cope with this problem by exploring the set of generated choices and actually delivering a plan of execution of the proposed actions.

It is still noticeable that, as pointed out by [12], the use of choice functions reduces significantly the programming effort when compared to the resolution rules approach used in [6]. In that case, the number of “repair rules” is bounded

by $O(\#R * \#L_T)$ (where $\#R$ is the number of consistency rules and $\#L_T$ is the number of location types) where in our approach it reduces to $O(\#L_T)$. Notice also that the generator functions are *manually* custom-tailored to the syntactical constraints of the particular meta-model. This avoids the problem of non-scalability of approaches that compute all possible choices (such as [9] and [15]) as pointed out in [12].

7 Conclusion

In this paper we propose an approach for obtaining automatic generated *repair plans* for a given inconsistent model. Our approach is based on three main mechanisms, which are the *Cause detection rules*, the *Generator functions* and the *search algorithm*. *Cause detection rules* are used to identify actions that make the model inconsistent. Once those actions are identified, they can be repaired. It should be noted that *Cause detection rules* are directly derived from inconsistency detection rules employed in Praxis and can substitute them. Even if, for the sake of efficiency, they currently have to be manually written, they should be automatically generated from detection rules. This automatical generation is however left as a future work. *Generator functions* are used to drive the generation of repair actions and are therefore loosely coupled with the *Cause detection rules*. The *search algorithm* is used to efficiently generate repair plans, which are composed of a sequence of repair actions. It is fitted to build repair plans that start by fixing the most recent causes of inconsistencies. This algorithm is also parameterizable in such a way that the size of the search space to be explored during its execution can be decided beforehand.

Thanks to these three mechanisms, the modeler can, besides obtaining repair plans that correct all inconsistencies in his model, get partial plans that start by proposing fixes to the inconsistencies that were more recently introduced in the model. We argue that this capacity helps the developer when correcting models that have too much inconsistency and which would therefore require too much time to compute a complete repair plan.

Our approach is integrated into the existing Praxis environment on top of Eclipse EMF and thus is accessible to modelers using any compatible EMF based Eclipse UML Editor (such as Papyrus, that was used in this study). We are currently elaborating an empirical study in order to measure the effect of our approach for industrial developers who work on UML models.

References

1. Selic, B.: The pragmatics of model-driven development. *IEEE Software* 20(5), 19–25 (2003)
2. Hessellund, A., Czarnecki, K., Wasowski, A.: Guided development with multiple domain-specific languages. In: Engels, G., Opdyke, B., Schmidt, D.C., Weil, F. (eds.) *MODELS 2007*. LNCS, vol. 4735, pp. 46–60. Springer, Heidelberg (2007)
3. Balzer, R.: Tolerating inconsistency. In: *Proc. Int' Conf. Software engineering (ICSE 1991)*, vol. 1, pp. 158–165 (1991)

4. Spanoudakis, G., Zisman, A.: Inconsistency management in software engineering: Survey and open research issues. In: Handbook of Software Engineering and Knowledge Engineering, pp. 329–380. World Scientific, Singapore
5. Van Der Straeten, R., Mens, T., Simmonds, J., Jonckers, V.: Using description logics to maintain consistency between UML models. In: Stevens, P., Whittle, J., Booch, G. (eds.) UML 2003. LNCS, vol. 2863, pp. 326–340. Springer, Heidelberg (2003)
6. Mens, T., et al.: Detecting and resolving model inconsistencies using transformation dependency analysis. In: Nierstrasz, O., Whittle, J., Harel, D., Reggio, G. (eds.) MoDELS 2006. LNCS, vol. 4199, pp. 200–214. Springer, Heidelberg (2006)
7. Elaasar, M., Brian, L.: An overview of UML consistency management. Technical Report SCE-04-18 (August 2004)
8. Blanc, X., Mougnot, A., Mounier, I., Mens, T.: Detecting model inconsistency through operation-based model construction. In: Robby (ed.) Proc. Int'l Conf. Software engineering (ICSE 2008), vol. 1, pp. 511–520. ACM, New York (2008)
9. Nentwich, C., Emmerich, W., Finkelstein, A.: Consistency management with repair actions. In: Proc. Int'l Conf. Software Engineering (ICSE 2003), Washington, DC, USA, pp. 455–464. IEEE Computer Society, Los Alamitos (2003)
10. OMG: Unified Modeling Language: Super Structure version 2.1 (January 2006)
11. OMG: Meta Object Facility (MOF) 2.0 Core Specification (January 2006)
12. Egyed, A., Letier, E., Finkelstein, A.: Generating and evaluating choices for fixing inconsistencies in UML design models. In: Proc. ACM/IEEE Int'l Conf. Automated Software Engineering (ASE 2008), pp. 99–108. ACM, New York (2008)
13. Russell, S.J., Norvig, P.: Artificial Intelligence: A Modern Approach. Pearson Education, London (2003)
14. Mougnot, A., Darrasse, A., Blanc, X.: Uniform random generation of huge meta-model instances. In: Paige, R.F., Hartman, A., Rensink, A. (eds.) ECMDA-FA 2009. LNCS, vol. 5562, pp. 130–145. Springer, Heidelberg (2009)
15. 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)

Dynamic Metamodel Extension Modules to Support Adaptive Data Management

Michael Grossniklaus¹, Stefania Leone²,
Alexandre de Spindler², and Moira C. Norrie²

¹ Dipartimento di Elettronica e Informazione, Politecnico di Milano
I-20133 Milano, Italy

`grossniklaus@elet.polimi.it`

² Institute for Information Systems, ETH Zurich
CH-8092 Zurich, Switzerland

`{leone,despindler,norrie}@inf.ethz.ch`

Abstract. Databases are now used in a wide variety of settings resulting in requirements which may differ substantially from one application to another, even to the point of conflict. Consequently, there is no database product that can support all forms of information systems ranging from enterprise applications to personal information systems running on mobile devices. Further, domains such as the Web have demonstrated the need to cope with rapidly evolving requirements. We define dynamic metamodel extension modules that support adaptive data management by evolving a system in the event of changing requirements and show how this technique was applied to cater for specific application settings.

1 Introduction

Nowadays, database systems support everything from enterprise applications to personal information management on mobile devices. These systems vary greatly, not only in terms of the scale, but also data management requirements, which may differ dramatically from one application to another. Currently, database products tend to dedicate themselves more or less explicitly to a given domain with major vendors offering either a family of products or means for design-time configurability. For example, the open-source database MySQL addresses a segment of the database market that differs substantially from that of products such as Oracle Database or IBM DB2, which were designed to support large-scale enterprise applications. In contrast, MySQL was originally developed to support Web applications and its kernel optimised for a subset of SQL to meet the requirements of that domain. MySQL also provides less support for transactional query processing since it is less relevant in typical Web applications.

In some cases, the requirements of one domain may even conflict with the requirements of another, making it impossible for one database product to support all domains. One reason for very different and even contradicting requirements is the fact that some of the features that are desirable during the design and development phase of a database application may be undesirable during operation.

For example, features such as schema evolution and versioning facilitate prototyping but often have adverse effects on system performance and may not be required for system operation. Moreover, many Web applications evolve rapidly and as a result the requirements may change dramatically during the operation.

The need for adaptation has already been recognised by the database community [1] and various proposals exist for *tailor-made data management* [2]. Traditional DBMS tended to be monolithic in structure and static in functionality. Over the last decades, there have been different approaches to allow a DBMS to be tailored to the requirements of particular applications and settings. The general goal was to find ways in which database systems could be made extensible and more configurable. While most of these approaches provide support for design-time adaptation, more recently approaches based on service-oriented architectures have been proposed that also support run-time adaptation.

Most existing approaches support adaptation through adaptable database architectures. In this paper, we propose a different approach, showing how adaptation in database systems can be supported by basing their implementation on a well-defined system metamodel that can be evolved through dynamic metamodel extension modules. Metamodel extension modules can be loaded and unloaded dynamically, thus uniformly providing database system adaptation at design-time and run-time. While our approach is generally applicable, we demonstrate it in the setting of an object database system.

We begin in Sect. 2 with a discussion of related work. Our approach together with the specification of metamodel extension modules is introduced in Sect. 3. The application of the proposed approach is demonstrated in Sect. 4 and 5 where the core metamodel and an example metamodel extension module are presented, respectively. Implementation details are presented in Sect. 6. Concluding remarks are given in Sect. 7.

2 Related Work

Adaptation of data management systems has been addressed in terms of lightweight, configurable database management architectures. The emergent trend of service-oriented architectures (SOA) has led to a number of proposals for service-oriented database architectures (SODA), e.g. [3,4], that allow application developers to configure a DBMS by linking together all services required for a given application domain. The general ideas for these architectures are very close to that of earlier configurable DBMS [5].

In [3], a service-oriented DBMS (SDBMS) architecture is described based on the layered architecture proposed in [6]. In addition to being able to extend functionality, it allows for the selection of an alternative service or service composition in the case of service failure. This could also involve introducing a wrapper to adapt the interface of a service. While the proposed architecture promises the flexibility required, there are no details about the mechanisms used to achieve the different kinds of flexibility and the implementation of the architecture. Further, it is not clear if and how run-time adaptation would be supported.

The CoBRA DB project [7] aims at providing run-time adaptation for DBMS. The focus lies on modularizing a DBMS and supporting module exchange at run-time in a transparent and atomic way. The authors experimented with two methods of enabling dynamic adaptation, namely dynamic aspect-oriented programming (d-AOP) and a second approach where a component implementation, or part of it, is exchanged in order to adapt the component while the interface remains valid. In [8], they present how a Transaction Manager can be added and removed at run-time using the d-APO approach which the authors argue in [7] has several disadvantages in terms of performance, code maintenance, limited functionality and testing. Therefore the approach of component replacement has been adopted in the CoRBA framework. In this approach, the adaptation manager is responsible for registering services offered by the components as well as for managing and triggering all adaptation requests.

In mobile applications, computing resources may be limited and it is therefore important that a DBMS can be configured and optimized for a particular application setting. COMET [9] is a component-based real-time database for automotive systems that represents a typical example of a DBMS that can be statically configured for a given application or target device. FAME-DBMS [10] is an approach to configurable DBMS in the area of embedded systems that follows the idea of software product lines with static system composition. DBMS functionality is tailored after the application has been developed to provide the minimal functionality required based on code analysis. For example, if the join operation is never used, the configured DBMS will not provide the operator. This approach is a design-time approach and run-time adaptation is not supported.

In summary, the ability to support various types of adaptation in database systems has been recognised as desirable, if not essential, in modern DBMS. Architectural solutions such as configurable DBMS are capable of supporting design-time adaptation, but they are not suited to run-time adaptation. While approaches based on SOA do support run-time adaptation, their motivation has mainly been to develop self-healing and self-managing systems in order to improve reliability. In contrast, our work focuses on supporting different and/or changing requirements of information systems brought about by wide variation in application settings and system evolution. We believe that in a setting where a DBMS is not distributed, for example applications on mobile devices, services are not ideally suited to solving the particular problem of adaptation since the fact that they are loosely coupled can have an adverse impact on system performance. Further, since many proposals adopt service-based approaches in which specific services are replaced or extended, it is difficult to support the types of adaptation that require adaptation across services such as storage, query processing and constraint management. For example, the kinds of adaptation that require changes to the basic structures of the data model to support spatial information or versioning may require changes to many parts of the system, especially if all data is to be handled uniformly. We therefore decided to investigate how a DBMS could be adapted through changes to the metamodel rather than to specific services.

3 Approach

The development of a database application typically involves defining a model of the application domain and implementing the means to create, retrieve, update and delete instances of the application domain concepts. The application model is itself defined in terms of the DBMS metamodel that specifies the core constructs supported by the system. In the case of relational technologies, the metamodel includes the concepts of relations and attributes, while object metamodels describe object types and their properties. Therefore, a DBMS must offer the basic database operations to create, retrieve, update and delete instances of the metamodel concepts in order to specify an application model. Also, most DBMS offer a database language (DBL) including data definition, data manipulation and data retrieval components.

Our approach assumes that data and metadata are handled uniformly, so that both the data model that defines the functionality of the DBMS and the application model are represented explicitly as data. This means that, not only may all database functionality such as storage management, query processing and constraint management be applied to metadata as well as data, but also they can be updated dynamically at run-time. The key to our approach to DBMS adaptation is to allow the core metamodel, corresponding database operations and the DBL to be changed or extended. This can be done either through a configuration process at design-time or by extending functionality dynamically at run-time to allow the DBMS to adapt to new requirements.

We claim that many requirements imposed by database applications can be met by extending the metamodel with additional concepts. Therefore, we decided to address the requirement of adaptive data management through a modular system metamodel. An overview of this approach is shown in Fig. 1.

The core database module $Module_{core}$ is shown on the left-hand side of Fig. 1. It comprises the core metamodel MM_{core} as well as the component to create, retrieve, update and delete core metamodel concepts ($CRUD_{core}$) and the core database language DBL_{core} . MM_{core} is a set of meta concepts $\{MC_1, \dots, MC_n\}$.

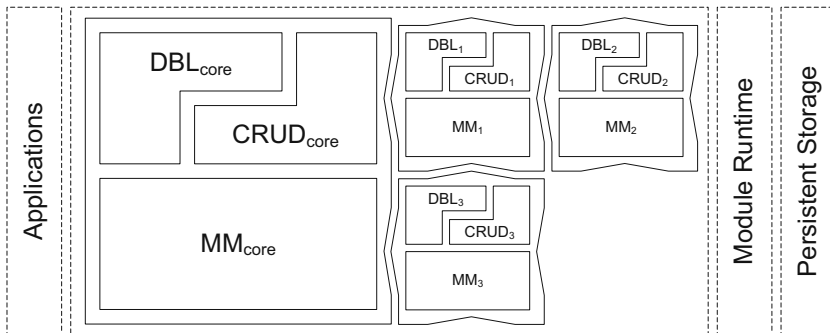


Fig. 1. Metamodel extension modules

As seen in Fig. 1, the definition of metamodel extension modules follows the design of the core system in the sense that each module also provides metamodel concepts, operations to manipulate them and an extension to the database language. The additional manipulation operations and database language extension for the new metamodel concepts are required to make the new functionality available to other parts of the system as well as to the application developer or end-user. In summary, database extensions consist of three components which, together, form what we refer to as a metamodel extension module.

More formally, a module can be defined as a triple

$$Module_{ext} = \langle MM_{ext}, CRUD_{ext}, DBL_{ext} \rangle$$

where MM_{ext} is a set of additional concepts made available to the application developer to model the application domain according to application-specific requirements, $CRUD_{ext}$ refers to the database operations and DBL_{ext} refers to the extensions to the database language.

In general, such an extension is a set of concepts $\{C_1, \dots, C_m\}$ where each concept $C_i \in MM_{ext}$ is an instance of a meta concept $MC_j \in MM_{core}$, formally written as $C_i \triangleleft MC_j$. Note that, once the metamodel extensions have been added to the metamodel by loading the module, they become part of it and remain indistinguishable from the perspective of an application or developer. Therefore, the application developer can easily take advantage of these additional concepts in order to model the application domain.

The term $CRUD_{ext}$ refers to those facilities required for an application or developer to manage the instances of all concepts defined by the module.

$$\forall C_i \in MM_{ext} \exists CRUD_{ext}(C_i) \in CRUD_{ext}$$

where $CRUD_{ext}(C_i)$ allows for instances of the concept C_i to be created, retrieved, updated and deleted. When a module is loaded, the set of its operators $CRUD_{ext}$ are registered with the database in such a way that they can be retrieved and used by the developer or application. In a more general sense, the database operations can be seen as forming the API of the module.

The component DBL_{ext} is a set of symbols extending the core database language DBL_{core} to allow access to the operations offered by $CRUD_{ext}$. In general, a database language is defined by a grammar G consisting of a set N of non-terminal symbols, a set Σ of terminal symbols and the set P of production rules where each rule maps from one string of symbols to another. In short, the grammar can be written as $G = (N, \Sigma, P)$. Consequently, with each module loaded, the core language DBL_{core} defined by the core grammar G_{core} is extended by DBL_{ext} by unifying its grammar with the core grammar as

$$G_{core} \cup G_{ext} = (N_{core} \cup N_{ext}, \Sigma_{core} \cup \Sigma_{ext}, P_{core} \cup P_{ext}).$$

Moreover, there may exist dependencies among modules. By default, all modules are dependent on the core module. However, a module may additionally be dependent on other modules which means that they must be loaded first. Similarly, a module cannot be unloaded if other loaded modules depend on it. In order

for the module run-time to check dependencies, a list $[Module_1, \dots, Module_n]$ of all dependent modules is defined as part of each module declaration.

Since the core components of our system also include a metamodel, database operations and a language, our system is built so that the core itself is defined as a module and loaded accordingly. In contrast to all other modules, the core module cannot be unloaded at run-time since all other modules depend on it. Nevertheless, the core module can be configured at design-time to adapt it, for example, to a mobile environment requiring lightweight databases or a heavily-used Web application relying on additional concepts to increase performance.

4 Core Metamodel

In this section, we present an example of a core database module consisting of a core metamodel MM_{core} , core management functionality $CRUD_{core}$ and a core database language DBL_{core} . Based on this, we will show the use of metamodel extension modules in the next section. As initially stated, our approach works independently of the given data model of a system as long as it is defined through a metamodel. Therefore, the approach can be equally applied to relational, XML and object databases. We have implemented the approach in the object database system OMS Avon [11] and therefore will present the details of the approach using this as an example.

We begin by introducing the concepts of the core metamodel MM_{core} shown in Fig. 2, which is based on the OM data model [12]. Essentially, the OM data model is an integration of entity-relationship (ER) and object-oriented models. In contrast to ER models where the concepts of entity types and entity sets are often merged, OM introduces a clear separation between the typing and classification of entities by using a two-level model. Each object has at least one object type that specifies the representation and behaviour of the object in terms of attributes and methods. Note that OM supports subtyping and also multiple instantiation which means that an object can be said to have multiple types. Objects are classified through membership in collections and each collection has a membertype that restricts membership to objects of a particular type. A collection is represented graphically as a shaded box with the membertype specified in the shaded part.

Just as types can be specialised through subtyping, classifications can be specialised through subcollections. A collection may have multiple subcollections and classification constraints such as *disjoint*, *cover*, *partition* and *intersect* may be placed over these. Note that for reasons of legibility, not all classification constraints are shown in Fig. 2.

Relationships in OM are represented by bi-directional associations that are defined in terms of a source and a target collection. Associations are a first-order concept of the model and are represented by binary collections. As in some extended ER models, cardinalities over associations are specified in terms of a minimum and maximum value that expresses the number of objects to which an object can be linked. Associations can also be specialised over collections. Associations are represented graphically as shaded ovals.

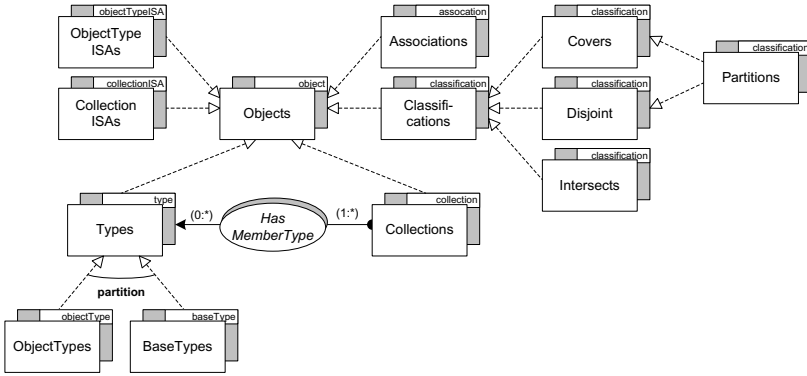


Fig. 2. Graphical representation of the core metamodel

| ObjectTypes | Collections | Associations |
|--|---|---|
| create(Name): ObjectType retrieve(Name): ObjectType getName(ObjectType): Name addAttribute(ObjectType, Name, Type) remove Attribute(ObjectType, Name) getAttributes(ObjectType): Collection delete(ObjectType) | create(Name, MemberType): Collection retrieve(Name): Collection getName(Collection): Name getMemberType(Collection): MemberType addMember(Collection, Member) removeMember(Collection, Member) delete(Collection) | create(Name, Domain, Range, Relation, ...): Association retrieve(Name): Association getName(Association): Name getDomain(Association): Collection getRange(Association): Collection addMember(Association, Member, Member) removeMember(Association, Member, Member) delete(Association) |

Fig. 3. UML class definitions of the core system operators

As can be seen in Fig. 2, the instances of all constructs of the core model—types, collections, associations, ISA relationships (subtypes and subcollections) and also classification constraints—are represented as objects. These objects are classified through membership of the corresponding metadata collections.

Formally, the concepts defined by the core metamodel MM_{core} are given by its set of types, its set of collections and its set of associations.

$$\begin{aligned}
 MM_{core} = & \{ \{ object, type, collection, association, \dots \}, \\
 & \{ Objects, Types, Collections, Associations, \dots \}, \\
 & \{ HasMemberType, \dots \}
 \end{aligned}$$

It is beyond the scope of this paper to describe all aspects of the OM model in detail.

In Fig. 3, we show UML type definitions of the manipulation operators for these concepts. These diagrams show the methods to create, retrieve, update and delete instances of the corresponding type. For example, the creation of a collection takes the name and member type of the collection as argument, internally creates an object, dresses it with the collection type, sets the name and member type attributes and returns this object. Given a collection object, its name and member type can be retrieved using the `getName` and `getMemberType`

methods. An object dressed with this membertype can be added to or removed from the collection using the `addMember` and `removeMember` methods. Finally, the `delete` method is used to delete a collection object.

The third part of the core module is the database language DBL_{core} . Associated with the OM data model, we have defined the OML language [13] which encompasses a data definition, data manipulation and query language. The query language is based on a collection algebra that defines a set of operators to manipulate and process collections and associations. Apart from being used for data definition, manipulation and querying, OML also serves as a declarative object-oriented implementation language for the methods of database objects as well as for stored procedures and triggers. An example of an OML script is given below.

```
/* data definition language */
create type contact ( name : string, phone : string );
create collection Contacts as set of contact;
/* data manipulation language */
$obj := create object;
dress $obj with contact ( name = "Fred Bloggs", phone = "555-2223344" );
insert [ $obj ] into Contacts;
/* query language */
$fred := first(all $c in Contacts having ($c.name like "(F|f)red.*"));
```

In the data definition section, the application developer creates an object type `contact`, which is used as membertype for collection `Contacts`. The first statement creates an object of type `objectType` in the core metamodel, while the second statement creates a collection object. The data manipulation section demonstrates how an object is created and instantiated with the `contact` type using the `dress` operation. Then the object is inserted into the `Contacts` collections. Finally, a simple selection query over the `Contacts` collection is shown that selects the previously created object. Formally, OML is defined by a grammar expressed as a set of productions P_{core} . For reasons of space, only a subset of P_{core} is given below.

```
statements → statement { ";" statement }
statement → [ ddl_statement | dml_statement | query_expression ]
ddl_statement → create_statement
create_statement → "create" [ create_object | create_objecttype | ... ]
create_object → "object"
create_objecttype → "type" name "(" attribute_list ")"
...
```

Correspondingly, the DBL_{core} component is given by

$$DBL_{core} = \{ \{ \text{statements, statement, ddl_statement, ...} \}, \{ \text{"create", "object", "type", ...} \}, P_{core} \}.$$

5 Metamodel Extension Module

In this section, we show how we produced an object database with integrated support for Web content management by defining the appropriate metamodel extension module based on previous work [14] that established four information concepts for content management: content, view, structure and layout. Content elements represent the objects that are to be published on the Web, view elements define which attributes and relationships of these objects are displayed, structure elements provide support for arbitrarily nested content hierarchies, and layout elements govern the presentation of content and structure. Due to space limitations, we will omit further discussion of view elements here. As all of these elements are context-aware, the resulting system is very flexible and well-suited to support personalisable and multi-channel Web applications. Context-awareness is supported based on a version model [15] that manages each object as a set of variants which are defined for specific context states. At runtime, the client context state is matched against the variant context states and the best matching variant is selected to represent the object.

The metamodel of the extension module is shown in Fig. 4. On the left, the hierarchy of content management concepts is shown. On the right, the object variants for context-awareness are shown. Note that we support object variants based on multiple instantiation similar to the approach presented in [16]. To publish an existing data object on the Web, it is simply dressed with an instance of type *variant* that augments the object with context state information. Then it is associated with an instance of type *content* that assigns a resource name to the content for referencing it based on a URL. Elements and variants are linked with two associations. *HasVariants* captures all context variants of an element, while *DefaultVariant* designates a fallback representation of the object that can be used, for example, in the absence of a client context state. In order to make the application of layout elements to content elements type-safe, both concepts are associated with a type. Since *Types* is a concept of the core metamodel, the content management metamodel is an extension of the core.

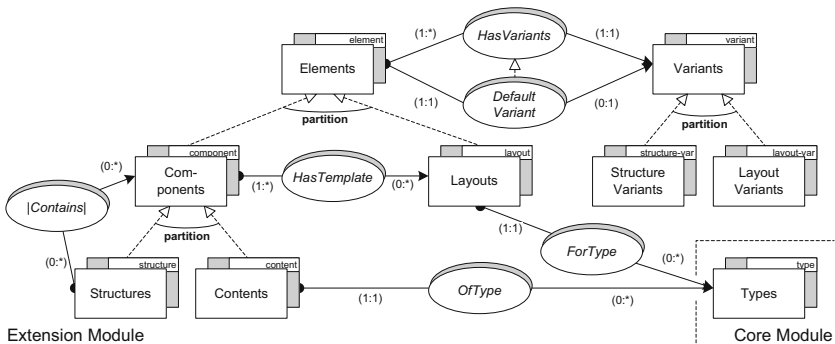


Fig. 4. Graphical representation of the content management metamodel

To implement this system as a module, the three module components have been defined as follows. The metamodel is a definition of the types, collections and associations introduced above.

$$MM_{cm} = \{ \{ element, variant, component, layout, structure, \dots \}, \\ \{ Elements, Variants, Components, Layouts, Structures, \dots \}, \\ \{ HasVariants, DefaultVariant, HasTemplate, \dots \} \}$$

Note that the collections **Elements** and **Variants** are subcollections of the **Objects** collection (not shown in the figure) of the core metamodel. The system operators providing the creation, retrieval, update and deletion of these metamodel concepts are defined as

$$CRUD_{cm} = \{ Content, Structures, Layouts, Variants \}.$$

Those operators relevant to the client of the content management system are shown in Fig. 5. Note that the operators shown implement just the management operations required in order to interact with the content management system. The part of the system that publishes the content on the Web is outside the scope of this paper. The presented operations make use of the core operators in $CRUD_{core}$ in order to implement their functionality. For example, the create operation in **Variants** calls the create operation in **Objects**, takes the returned object and dresses it with the variant type before returning it.

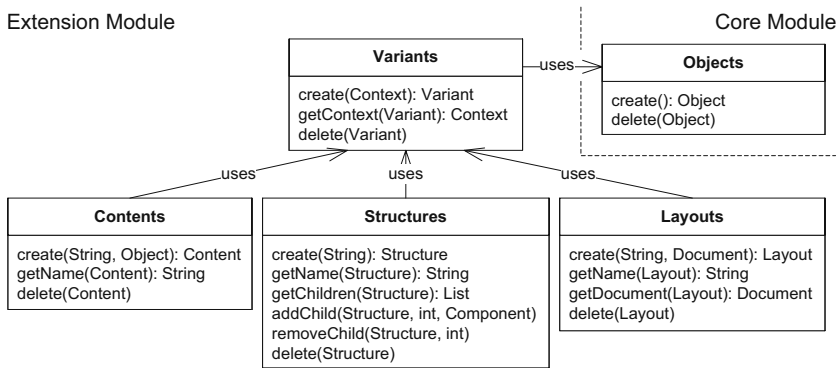


Fig. 5. UML class definitions of the content management system operators

Finally, the database language extension DBL_{cm} provides the vocabulary allowing the operations offered by $CRUD_{cm}$ to be invoked. The example below shows how to setup the content management to publish a created object $\$obj$ on the Web.

```

/* Create content element*/
create content fred from $obj context ( lang = "english" ) default;
/* Create main structure */
create structure index context ( lang = "english" ) default;
insert [ fred ] into index;
/* Create a template for person content */
create layout contact_layout for content context ( lang = "english" )
  [ <xsl:template match="..."> ... </xsl:template> ];

```

The extensions to the core language as defined in the previous section are formally defined by the set of productions P_{cm} . Again, we limit ourselves to a subset due to space limitations.

```

create_statement → "create" [ ... | create_content | create_structure | create_layout ]
  create_content → "content" name "for" object_ref [ context ]
  create_structure → "structure" name [ context ]
  create_layout → "layout" name "for" type_ref [ context ] [" template "]"
  context → "context" value_list [ "default" ]
insert_statement → "insert" [ collection_insert | structure_insert ]
  structure_insert → component_ref "into" structure_ref
  ...

```

These productions lead to the definition of the DBL_{cm} component as

$$DBL_{cm} = \{\{\text{create_content, create_structure, create_layout, context, \dots}\}, \{\text{"content", "structure", "layout", "context", \dots}\}, P_{cm}\}.$$

To conclude this section, we note that the approach has also been used to develop an object database that allowed personal information to be integrated with Web 2.0 data sources, to extend an object database to support event-based programming [17] and to develop a platform for peer-to-peer data sharing in mobile applications [18].

6 Implementation

As shown in Fig. 1, the module runtime of our adaptive database system is built on top of a low-level persistent storage designed to provide flexible data management. The flexibility is achieved by means of the data model outlined in Fig. 6. We distinguish the notion of an *object* which strictly identifies a real-world object and an *instance* which bears the attribute values declared by an *object type*. An *extent* is a bulk of values that are described by an *extent type* and used to support collections and associations. Note that attribute values and extent members may be objects, extents or built-in values such as integer or string.

The persistent storage implements persistent data management according to this data model and exposes the API shown in Fig. 7. Object types are created with a list of attribute definitions, each declaring the name and type of an attribute. An extent type is created by providing the membertype. In both cases, an object must be provided which will serve as an identifier referring to the created type.

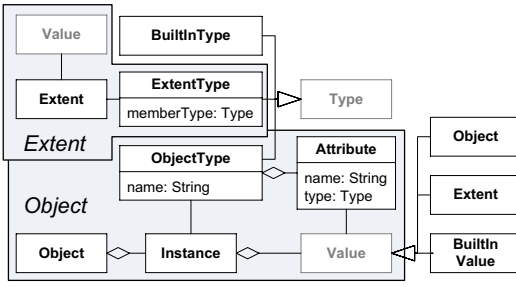


Fig. 6. Data model of persistent storage

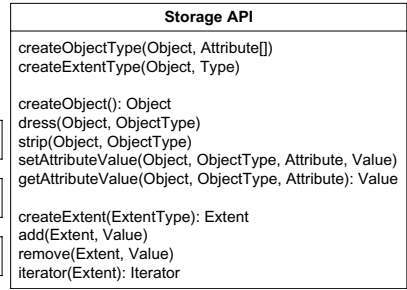


Fig. 7. Persistent storage API

The API offers methods to manage objects and extents. Once an object has been created, instances can be added or removed using the `dress` and `strip` methods, respectively. Attribute values can be set and retrieved by providing the object, the object type declaring the attribute to be accessed and the attribute itself. An extent is created by providing the extent type. Given an extent, values can be added and removed as well as accessed by means of an iterator. Methods for the deletion of types, objects and extents as well as a query facility are also provided, but not shown in the figure.

The module runtime provides a single Java class `OMObject` that represents all metamodel concepts uniformly in terms of the notion of an object as defined by the persistent storage data model. It offers means to add and remove instances as well as to access and manipulate attribute values. Another class `OMExtent` wraps the notion of an extent and allows for members to be added, accessed and removed. By separating the metamodel concepts from the actual concept representation within the programming language, we achieve the flexibility of being able to alter and extend the metamodel at runtime. Therefore, altering and extending the metamodel in the database does not require any changes to the in-memory Java representations of metamodel concepts.

To manage modules, the module runtime provides a `ModuleManager` that offers methods to load and unload modules. When the manager is initialised, it reads a configuration file where modules may be specified at design-time. The module manager requires that a module implementation follows the `Module` interface that is also defined by the runtime. This interface defines four methods that correspond to the lifecycle of modules. To load a module, the module concepts are created by the `bootstrap` method, then the operators are initialised with `registerCRUDs` and, finally, `generateDBL` loads the database language. If no longer required, the manager disposes of modules by invoking the `unload` method.

In the remainder of this section, we discuss the implementation of the core and the extension modules presented in Sects. 4 and 5, respectively. Due to the fact that all extension modules are implemented using the core module, the implementation of the core module differs from that of an extension module. For both modules, we will present an excerpt of the bootstrap and the operator implementation. To illustrate the discussion, Fig. 8 gives a UML interaction diagram that shows the communication between all involved actors.

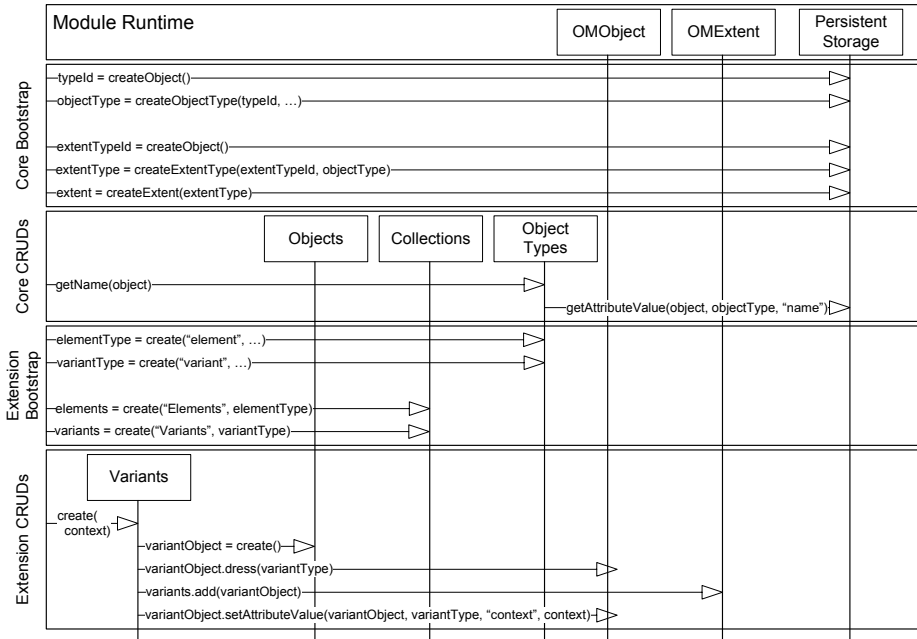


Fig. 8. UML interaction diagram of core and extension bootstrap and operators

The core metamodel is implemented entirely using the persistent storage shown in Fig. 7. The bootstrap process creates the core concepts represented in terms of types and collections and their relationships using the methods exposed by the persistent storage. As shown at the top of Fig. 8, the core database module creates the type `objectType` using the `createObjectType` method. The collection `ObjectTypes` is created by first creating the extent type using the method `createExtentType` and then creating the extent using `createExtent`. All other types and collections are created similarly.

Below the excerpt of the core bootstrap, Fig. 8 shows how the core operators make use of the persistent storage API to implement their concept-specific services. They take an `OMObject` instance as argument—or return one—and encapsulate the creation, deletion, attribute access and manipulation implemented using the methods of the persistent storage. For example, the object type operator to create an object type gathers the required arguments and calls the `createObjectType` method. The read and manipulation operations make use of the `setAttributeValue` and `getAttributeValue` methods. In the figure, we give an example of the latter, showing how the `getName` method of `ObjectTypes` uses the storage to retrieve that name of an object type. Finally, retrieval is implemented using the querying facility.

Extension modules are completely decoupled from the persistent storage as they only interact with the module runtime and the core module. The extension bootstrap procedure shown near the bottom of Fig. 8 consists of using the

core operators to create instances of the core metamodel representing extension concepts. For example, the content management module uses the core operator `ObjectTypes` to extend the metamodel with object types such as `element` and `variant`. In the same way, it uses the `Collections` operator to create the corresponding collections, i.e. `Elements` and `Variants`.

Finally, at the bottom of Fig. 8, we present how the extension operators use the core operators to provide the creation, attribute access and manipulation, retrieval and deletion facilities of the extension concepts. We use the example of the `Variants` operator to show how its `create` method has been implemented based on the core operator `Objects` as well as the module runtime classes `OMObject` and `OMExtent`.

To generate the database language of a module, we currently use JavaCC¹ and, therefore, the grammar is expressed in terms of the JavaCC syntax. Each module provides its grammar as a single file that is merged with the corresponding grammar files of other modules to obtain a comprehensive grammar before generating the parser. As a consequence, we can only support design-time adaptation for modules that require a language extension since parser generation requires an additional compilation step. To the best of our knowledge, there are no compiler compilers available at the moment that overcome this limitation and we are investigating how to engineer a solution that also provides runtime adaption of the database language.

7 Conclusions

We have motivated the need for adaptive database management systems to support configuration at design-time and evolution at run-time and proposed an approach that is based on revisions to the system metamodel. We have shown how this can be implemented and also demonstrated the use of the approach by means of an example. Even though the paper describes the application of the approach in the setting of an object database, it is important to emphasise that the approach generalises to all systems that use well-defined metadata to describe the data that they manage. While our main focus to date has been on achieving the desired functionality, we recognise that such flexibility of adaptation comes at a price in terms of performance and are now investigating exactly what the overhead is and how it can be reduced.

References

1. Stonebraker, M., Cetintemel, U.: “One Size Fits All”: An Idea Whose Time Has Come and Gone. In: Proc. Intl. Conf. on Data Engineering, ICDE (2005)
2. Apel, S., Rosenmüller, M., Saake, G., Spinczyk, O. (eds.): Proc. EDBT Workshop on Software Engineering for Tailor-made Data Management. University of Magdeburg (2008)

¹ <http://javacc.dev.java.net/>

3. Subasu, I.E., Ziegler, P., Dittrich, K.R., Gall, H.: Architectural Concerns for Flexible Data Management. In: Proc. EDBT Workshop on Software Engineering for Tailor-made Data Management, SETMDM (2008)
4. Tok, W.H., Bressan, S.: DBNet: A Service-Oriented Database Architecture. In: Proc. Intl. Conf. on Database and Expert Systems Applications, DEXA (2006)
5. Dittrich, K.R., Geppert, A. (eds.): Component Database Systems. Morgan Kaufmann, San Francisco (2001)
6. Härder, T.: DBMS Architecture – New Challenges Ahead. *Datenbank-Spektrum* 14 (2005)
7. Irmert, F., Fischer, T., Meyer-Wegener, K.: Runtime Adaptation in a Service-Oriented Component Model. In: Proc. Intl. Workshop on Software Engineering for Adaptive and Self-managing Systems, SEAMS (2008)
8. Irmert, F., Lauterwald, F., Neumann, C.P., Daum, M., Lenz, R., Meyer-Wegener, K.: Semantics of a Runtime Adaptable Transaction Manager. In: Proc. Intl. Database Engineering & Applications Symposium, IDEAS 2009 (2009)
9. Nyström, D., Nolin, M., Norström, C., Hansson, J.: COMET: A Component-Based Real-Time Database for Automotive Systems. In: Proc. Workshop on Software Engineering for Automotive Systems (2003)
10. Rosenmüller, M., Siegmund, N., Schirmeier, H., Sincero, J., Apel, S., Leich, T., Spinczyk, O., Saake, G.: FAME-DBMS: Tailor-Made Data Management Solutions for Embedded Systems. In: Proc. EDBT Workshop on Software Engineering for Tailor-made Data Management, SETMDM (2008)
11. Norrie, M.C., Grossniklaus, M., Decurtins, C., de Spindler, A., Vancea, A., Leone, S.: Semantic Data Management for db4o. In: Proc. Intl. Conf. on Object Databases, ICODB (2009)
12. Norrie, M.C.: An Extended Entity-Relationship Approach to Data Management in Object-Oriented Systems. In: Elmasri, R.A., Kouramajian, V., Thalheim, B. (eds.) ER 1993. LNCS, vol. 823. Springer, Heidelberg (1994)
13. Lombardoni, A.: Towards a Universal Information Platform: An Object-Oriented, Multi-User, Information Store. PhD thesis, ETH Zurich, Zurich, Switzerland (2006)
14. Grossniklaus, M., Norrie, M.C.: Information Concepts for Content Management. In: Proc. Intl. Workshop on Data Semantics in Web Information Systems (2002)
15. Grossniklaus, M., Norrie, M.C.: An Object-Oriented Version Model for Context-Aware Data Management. In: Benatallah, B., Casati, F., Georgakopoulos, D., Bartolini, C., Sadiq, W., Godart, C. (eds.) WISE 2007. LNCS, vol. 4831, pp. 398–409. Springer, Heidelberg (2007)
16. Beech, D., Mahbod, B.: Generalized Version Control in an Object-Oriented Database. In: Proc. Intl. Conf. on Data Engineering (1988)
17. Grossniklaus, M., Leone, S., de Spindler, A., Norrie, M.C.: Unified Event Model for Object Databases. In: Proc. Intl. Conf. on Object Databases, ICODB (2009)
18. 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)

Supporting Runtime System Evolution to Adapt to User Behaviour^{*}

Estefanía Serral, Pedro Valderas, and Vicente Pelechano

Centro de Investigación en Métodos de Producción de Software (ProS)
Universidad Politécnica de Valencia, Valencia, Spain
{eserral,pvalderas,pele}@dsic.upv.es

Abstract. Using a context-aware approach, we deal with the automation of user routines. To do this, these routines, or user behaviour patterns, are described using a context model and a context-adaptive task model, and are automated by an engine that executes the patterns as specified. However, user behavior patterns defined at design time may become obsolete and useless since users needs may change. To avoid this, it is essential that the system supports the evolution of these patterns. In this work, we focus on supporting this evolution by confronting an important challenge in evolution research: raise the level in which evolution is applied to the modelling level. We develop mechanisms to support the pattern evolution by updating the models at runtime. Also, we provide end-users with a tool that allows them to carry out the pattern evolution by using user-friendly interfaces.

Keywords: Context adaptation, models, tasks, user behaviour pattern automation.

1 Introduction

Context-aware systems are those capable of adapting its behaviour according to context and performing actions on behalf of users without being intrusive. Using a context-aware approach, we deal with the automation of user routines. A routine, or behaviour pattern, is a set of tasks characterized by habitual repetition in similar contexts [1]. Some patterns are determined by our lifestyle, e.g. reading electronic mail and opening certain web pages as soon as we have access to internet; others are reactions to things happening around us, e.g. lowering every blind and winding up every awning when it starts to rain.

Several works [2, 3] have dealt with inferring these patterns from user action observation by using machine-learning algorithms. However, they require a lot of training data and may automate actions that users do not want to be automated because algorithms do not know the semantics of user-performed actions. Moreover, note that many patterns are well known before the system is implemented. These patterns can

^{*} This work has been developed with the support of MEC under the project SESAMO TIN2007-62894 and co financed by FEDER, in the grants' program FPU.

be described at design time and automated under the adequate context conditions. Our approach [22] achieves this. It presents a context-adaptive task model that allows each behaviour pattern to be specified through a set of tasks that have to be carried out to automate it. These tasks are modelled according to context, which is previously specified in a context model. Both the context model and the task model are also used at runtime. A Model-based Automation Engine (MAE) interprets them to execute the patterns as specified.

However, this is not enough to automate user behaviour patterns since user needs may change in the future. It is essential that the system supports the evolution of the designed patterns to adapt to user behaviour changes; otherwise, their automation may become a burden on users instead of a way of helping them. In this work, we focus on supporting this evolution. To do this, we confront one of the most important challenges identified in software evolution according to works such as [4, 5]: raise the level of abstraction in which evolution is applied to the modelling level, i.e. perform system evolution by evolving the models that specify it. Since the research in model-driven development started, this challenge becomes increasingly more relevant, and new approaches and tools for dealing with it are urgently needed. However, although many advances have been done in software evolution research, almost all existing approaches for supporting it are primarily targeted to source code, including the proposed approaches for automating user routines that supports evolution. In contrast, we propose applying runtime evolution at modelling level, which allows us to manage it at a high level of abstraction.

To do this, we take advantage of the design of our approach. Since the task model and the context model are interpreted at runtime to execute the corresponding behaviour patterns, as soon as the models are modified, the changes are applied by the system. Thus, we design and implement mechanisms to support the update of both models at runtime [6] using the same high level concepts used for building the models. Furthermore, we provide end-users with a user-friendly tool to evolve the patterns. This end-user tool uses the provided mechanisms to carry out the evolution. Note that these mechanisms could also be automatically applied, for instance, by using machine-learning algorithms that detect changes in user behaviour. However, this may lead to applying changes that do not correspond to user desires and to automating tasks that users do not want to be automated. This may be annoying to users and may cause users to feel loss of control of the system.

The remainder of the paper is organized as follows. Section 2 gives an overview of our approach. Section 3 describes real examples of behaviour patterns. Section 4 describes the context model and the task model. Section 5 presents the mechanisms to address pattern evolution. Section 6 explains the end-user tool. Section 7 evaluates our approach. Section 8 presents the related work. Section 9 presents the conclusions and the future work.

2 Automating User Behaviour Patterns: An Overview

Our approach, which is shown in Fig. 1, deals with automating the routines that users want to be automated, according to their desires and demands. To achieve this, analysts first interview end-users to determine the tasks that they perform and identify

behaviour patterns in these tasks. Also, the analysts determine the context in which the pattern has to be triggered and the context in which its tasks are performed. Thus, the analysts describe the needed context information by using a context model. With the participation of users, the analysts then specify the context situation that triggers the execution of the pattern, the tasks to be executed for each one of the identified patterns, and the temporal relationships that must be accomplished for the execution of these tasks. The behaviour pattern tasks and their relationships are specified using the context information of the context model, in such a way that the task execution automatically adapt according to context.

In addition, the tasks to be executed are related to a pervasive service that can carry it out. A pervasive service is a piece of software that is in charge of controlling devices in order to achieve a task goal. For instance, if a task whose goal is to lower the blinds has been defined, this task is associated to a pervasive service that executes this action by interacting with the blind device. These services are also in charge of updating the context model when context changes arise (e.g. indicating the current state of blinds: open or closed). To implement these pervasive services, a MDD method that generates them in Java/OSGi [7] technology from models is used. As an example, Figure 1 shows a partial view of a Java/OSGi-based pervasive service implementation. This method is out of the scope of this paper; more information about it can be found in previous works [8, 9].

Once the behaviour patterns have been specified using the Context Model and the Task Model, a Model-based user task Automation Engine (MAtE) is in charge of automating the patterns when needed. To do this, MAtE uses a Context Monitor that monitors the context changes reflected by changes on the context model. When a context change is arisen, MAtE checks whether some context situations are fulfilled by interpreting the task model and querying the context model at runtime. If so, MAtE interprets the task model to perform the tasks of the corresponding pattern according to their specification. To perform each task, MAtE executes the pervasive service related to it.

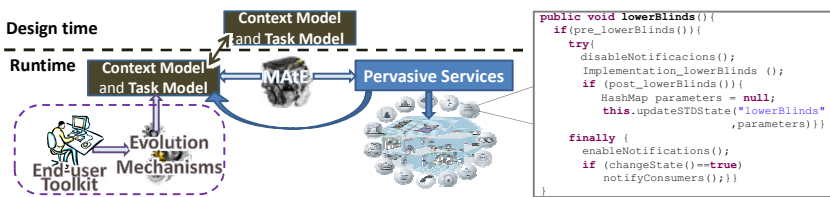


Fig. 1. Overview of the approach

However, without explicit support for pattern evolution, system may become progressively less satisfactory. In this work, we deal with this problem by allowing end-users to be able to evolve the behaviour patterns according to their needs. To do this, our approach provides a set of mechanisms to allow behaviour patterns to be evolved by updating the task model and the context model at runtime. Moreover, we provide end-users with a tool that allows them to evolve these patterns by using user-friendly interfaces. This tool uses the provided mechanisms to carry out the evolution.

3 User Routines: A Case Study

We present some user examples in the smart home domain, since it is close to readers. A married couple (we will refer to them as Bob and Sarah), was first interviewed to determine their domestic behaviour patterns. We next present four representative examples of the behaviour patterns identified for the case study:

1. **WakingUp:** At 7.00 a.m. on working days, the system wakes Bob up with his preferred music and raises the bedroom blinds to the middle.
2. **GoingOut:** When Bob and Sarah left the house, the system closes all the windows and doors, and switches off all the lights. If it is a working day, the system puts the air and heating conditioner in energy-saving mode; otherwise, the system switches it off, turns the water off at the mains and turns the gas off. Finally, the system reminds users to enable security (they prefer to activate the security themselves; therefore, the system only sends them a notification to remind them to do this).
3. **Recording:** If Sarah is not at home at 18.30 p.m., her favourite program is recorded.
4. **StormSecurity:** If it starts to rain, the system lowers all the blinds and winds up all the awnings. If it does not stop raining and the garden sprinklers are switched on, the system switches them off. When it stops raining, the system calculates the cubic meters of rainfall and updates the irrigation timetable according to it.

4 The Context Model and the Task Model

The context model and the task model are used at design time for specifying the behaviour patterns that users want to be automated and their context; and also at runtime for supporting the specified behaviour pattern automation and evolution.

4.1 Ontology-Based Context Model

The context model semantically describes the Context required for properly automating the user behaviour patterns. This model is based on an ontology proposed in previous works [8], which defines classes such as *User*, *EnvironmentProperty* or *Location* to capture context. We used an ontology-based approach because it provides a formal analysis of the domain knowledge and allows common understanding of the structure of context. At design time, we use the EODM plugin [10] to graphically specify the context. EODM provides a tree graphic editor and also stores the model in the Web Ontology Language (OWL) [11]. OWL is an ontology language that greatly facilitates knowledge automated reasoning and is a W3C standard. At runtime, we use the model representation in OWL. In OWL, the classes of the ontology are defined by OWL classes, and the context of the system is represented by OWL individuals, which are instances of these classes. Fig. 2 shows an example of context model in its both representations (design and run time).

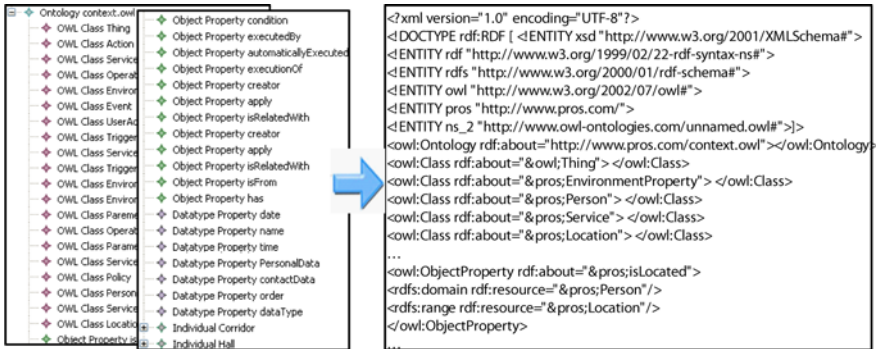


Fig. 2. An example of context model (graphical and OWL representation)

4.2 Context Adaptive Task Model

This model describes the behaviour patterns that have to be automated. Each pattern is specified by a **task hierarchy**. The root task represents the behaviour pattern and has an associated context situation, which defines the set of context conditions whose fulfilment triggers the execution of the pattern. It is broken down into *Composite Tasks* (which are intermediate tasks that can be broken down), and/or *System Tasks* (which are leaf tasks). Composite Tasks are used for grouping subtasks that share a common behaviour or goal. System tasks can be supported by a pervasive service. Both types of task can have a context precondition, which defines the context conditions that must be accomplished so that a task is performed (if the precondition is not accomplished, the task is not executed). Each task also inherits the context preconditions of its parent task.

As examples, the modelling of the *GoingOut* and *StormSecurity* patterns (Section 3) using the task model is shown at the top of Fig. 3. For instance, the *StormSecurity* behaviour pattern is triggered when it starts to rain as indicated in the context situation. The pattern is split into four tasks: the *lower blinds*, *wind up awnings* and *switch sprinklers off* System Tasks, and the *modify irrigation system* Composite Task, which is also split into the *calculate rainfall* and *update irrigation timetable* System Tasks. In addition, the *switch sprinklers off* System Task has a context precondition (which is shown between brackets before the name of the task) that indicates that this task will be only executed when it is raining and the system is watering the garden.

Tasks with the same father task are related by **temporal relationships**. They describe how the tasks are executed. For instance, the temporal relationship between the *wind up awnings* and *switch sprinklers off* tasks indicates that the sprinklers will be switched off 3 minutes after the awnings have been wound up (provided the context precondition fulfils).

At design time, the task model is specified by means of an editor developed using the Eclipse platform and the EMF and GMF plugins [10]. By using this editor, the model can be graphically edited (as the top section of Fig. 3 shows) and is also stored in XMI (XML Metadata Interchange). At runtime, we use its representation in XMI. The bottom section of Fig. 3 shows part of the XMI representation of the

StormSecurity pattern, where the properties of the *lower blinds* and *wind up awnings* tasks are shown. Note that the *service* property is a reference to the service in charge of executing the task; e.g. the *lower blinds* task is related to the *lowerBlinds* method, which was presented in Section 2.

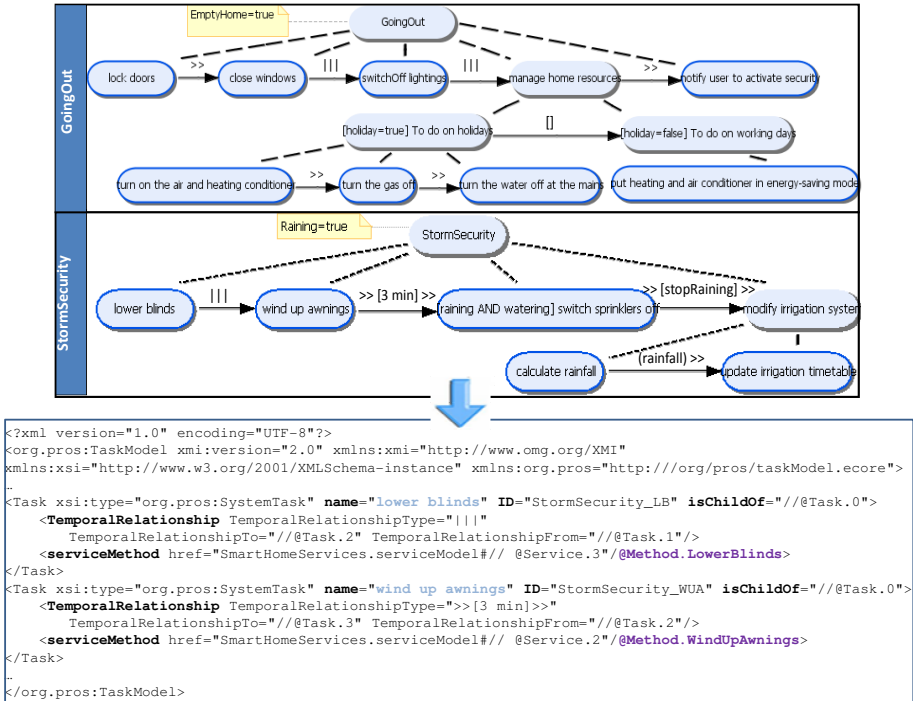


Fig. 3. Examples of behaviour pattern modelling (graphical and XMI representation)

5 Addressing the Evolution of User Behaviour Automation

The design of our approach facilitates to perform system evolution at modelling level, which is one of the top challenges in software evolution research. This is because MATe automates the patterns by interpreting the models (the context model and the task model) where they have been specified, at runtime. Thus, if the models are changed to adapt the patterns, the changes are also taken into account by MATe. Therefore, our approach gives us three immediate benefits to perform system evolution: 1) we do not have to maintain the consistency between the system modelling and system implementation as modifications are applied; 2) the evolution can be managed at a high level of abstraction (modelling level); 3) models can provide us with a richer semantic base for runtime decision-making related to system adaptation.

From these premises, we address the pattern evolution by adapting the models where they are specified. We provide mechanisms capable of evolving the task model and the context model at runtime. These mechanisms use the concepts of the own

language (task, behaviour pattern, user, etc.) used for specifying the models, which facilitates understand and handle the evolution. In addition, the mechanisms ensure that the changes are in accordance with their metamodel definition and, therefore, syntactically correct. In this way, these mechanisms define how the patterns can be changed over time and maintain software quality characteristics. Note that the use of such mechanisms raises the evolution level to the modelling level which allows the system to be evolved at runtime by using high level abstraction concepts instead of by changing lines of code. Furthermore, the mechanisms are implemented in Java applying software design patterns [21] and are provided as APIs; therefore, they can be imported and used by any Java application.

5.1 Supporting the Task Model Evolution

Each automated behaviour pattern is specified by a task hierarchy in the context-adaptive task model. Thus, to support the system adaptation to new user automation requirements, we have implemented a set of Model-Based User Task evolution mechanisms (MUTate) that allows evolving these patterns. For instance, MUTate allows the following: adding new tasks to a pattern; modifying the context precondition that must be accomplished so that a task can be executed; creating a new behaviour pattern; etc. To do this, MUTate provides an API that allows any elements of the specified task model (which are those specified in its metamodel, such as Behaviour Patterns, Tasks, Relationships between tasks, etc.), to be created, modified, or deleted. Specifically, this API consists of a Java class for each one of the elements of the task model metamodel. Each class provides:

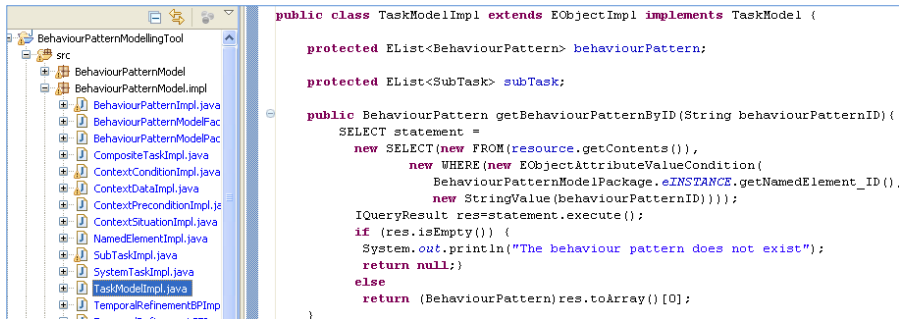
- An attribute for each one of the properties and relationships of the metamodel element that the class represents; e.g., the *BehaviourPattern* class has *name* and *task* as attributes.
- Get, set and delete methods for each one of these attributes; e.g., *getName*.
- An add method for the attributes whose type is a List. This method allows an element to be directly added to the list; e.g., *addTask* method.
- Get and delete methods for access to a certain element of the attributes whose type is a list. These methods allow us to get and delete an element of the list by searching for it by using one of its identifier properties; e.g. *getTaskByID*.

In addition, the API provides a *Factory* class for creating new instances (of the classes defined in the task metamodel) of a task model.

We have used the EMF, EMF Model Query (EMFMQ), and EMF Model Transaction (EMFMT) plugins of the Eclipse Platform [10] to implement MUTate. From the metamodel of the task model in ecore, EMF generates a basic API for managing a task model. This API provides the *Factory* and *Model* classes, as well as a Java interface and an implementation class for each one of the classes of the metamodel. These implementation classes provide get and set methods to access and change the information of the instances specified in the model. We have extended these classes by implementing the methods explained above. EMFMQ is used to search for and get the instances of the model that need to be modified. EMFMT provides us with mechanisms for making transactions, reading and writing models on multiple threads,

and validating the semantic integrity of the modified model by detecting invalid changes. We have also extended the implementation provided classes in order to: 1) add, modify, and delete a complete behaviour pattern as a unique transaction; 2) allow the reading and writing of the task model at the same time; and 3) semantically validate the changes in the task model.

Figure 4 shows a partial view of the source code of MUTate. As an example, the figure shows the *getBehaviourPatternByID* method of the *TaskModel* class. This method returns the behaviour pattern whose ID is the same that the *behaviourPatternID* argument value. To find the pattern, it searches for it by using a query statement build with EMFMQ.



```

public class TaskModelImpl extends EObjectImpl implements TaskModel {

    protected EList<BehaviourPattern> behaviourPattern;

    protected EList<SubTask> subTask;

    public BehaviourPattern getBehaviourPatternByID(String behaviourPatternID) {
        SELECT statement =
            new SELECT(new FROM(resource.getContents()),
                new WHERE(new EObjectAttributeValueCondition(
                    BehaviourPatternModelPackage.eINSTANCE.getNamedElement_ID(),
                    new StringValue(behaviourPatternID))));
        IQueryResult res=statement.execute();
        if (res.isEmpty()) {
            System.out.println("The behaviour pattern does not exist");
            return null;
        }
        else
            return (BehaviourPattern)res.toArray()[0];
    }
}

```

Fig. 4. Partial view of MUTate source code

5.2 Supporting the Context Model Evolution

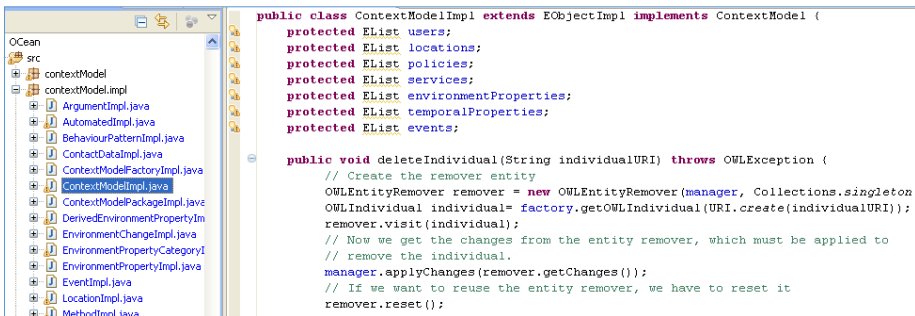
New automation requirements may require new context information to support them. In our approach, Context is captured in the OWL context model as OWL individuals. To change the context model, we have implemented a set of Ontology-based Context model Evolution mechanisms (OCEan) that allows evolving these individuals. For instance, if a change in a pattern requires knowing the preferred environmental temperature of Bob and Sarah, OCEan allows us to create this new user preference (e.g. *idealTemperature*) to the User individuals that represent Bob and Sarah. To do this, OCEan provides an API that allows any individual of the context model to be created, obtained, modified, and deleted. Specifically, this API consists of a *Model* class that allows us to open the context model, save it, and manage its individuals (*addIndividual*, *getIndividual* and *deleteIndividual*) in a generic way. The API also provides an implementation class (and its Java interface) for each one of the OWL classes defined in the context ontology. Each Java class provides:

- An attribute for each one of the properties and relationships of its OWL class; e.g., the *User* OWL class has *DNI* and *preference* as attributes.
- Get and set methods for each one of these attributes; e.g., *getDNI*.
- An add method for the attributes whose type is a List. This method allows an element to be directly added to the list; e.g., *addPreference* method.

- Get and delete methods for access to certain element of the attributes whose type is a List. These methods allow us to get or delete an element of the list by searching for it using one of its identifier properties; e.g. *getPreferenceByName*.
- The new method that creates an individual of the corresponding OWL class and calls the *addIndividual* method of the Model class to add it to the context model.

We have used the OWL API 2.1.1 [11], SPARQL [12] and the Pellet reasoner 1.5.2 [13] to implement Ocean. The OWL API is an open-source API that provides facilities for creating, examining and modifying any element of an OWL ontology. SPARQL is a graph-matching query language recommended by the W3C that allows queries to be built to search for certain individual. Pellet is an open-source OWL reasoner that provides reasoning services. Pellet allows us to launch a SPARQL query against the model.

Figure 5 shows a partial view of the source code of Ocean. As an example, the figure shows the *deleteIndividual* method of the *ContextModel* class. Using the OWL API, this method first creates a *remover* entity, it then gets the individual that is to be deleted, and passes it to the *remover* entity for deletion. Finally, the method applies the changes in the context model.



```

public class ContextModelImpl extendsEObjectImpl implements ContextModel {
    protected EList users;
    protected EList locations;
    protected EList policies;
    protected EList services;
    protected EList environmentProperties;
    protected EList temporalProperties;
    protected EList events;

    public void deleteIndividual(String individualURI) throws OWLException {
        // Create the remover entity
        OWLEntityRemover remover = new OWLEntityRemover(manager, Collections.singleton(
        OWLIndividual individual= factory.getOWLIndividual(URI.create(individualURI));
        remover.visit(individual);
        // Now we get the changes from the entity remover, which must be applied to
        // remove the individual.
        manager.applyChanges(remover.getChanges());
        // If we want to reuse the entity remover, we have to reset it
        remover.reset();
    }
}

```

Fig. 5. Partial view of Ocean source code

5.3 Using the Evolving Mechanisms

In this subsection, we present an example of how the presented mechanisms are used (see Figure 6). In this example, we add a new preference (*wakeUpTime*) to the user Bob. This preference indicates the time he prefers to wake up. To do this, we simply create a new preference, search for the user that represents Bob and add the created preference to his list of preferences. Afterwards, we change the context situation of the *WakingUp* pattern in order to indicate that it must be triggered when the time is the value indicated in the *wakeUpTime* preference. Thus, MUTate and Ocean allow the models to be evolved by using concepts of a high level of abstraction (Preference, ContextSituation, etc.), which are easy for developers to understand and use. However, these mechanisms are still difficult for end-users. For this reason, we have also developed an end-user tool that allows them to evolve the patterns by using intuitive interfaces.

```
//Create a new user preference
Preference preference= new Preference("wakeUpTime", "8:00");
User user= contextModel.getUserByName("Bob");
user.addPreference(preference);
//Using this preference in the context situation of a pattern
BehaviourPattern behaviourPattern = taskModel.getBehaviourPattern("WakingUp");
behaviourPattern.setContextSituation("currentTime=wakeUpTime AND workingDay=true");
```

Fig. 6. Code example for model evolution

6 End-User Toolkit for Evolving the System

To carry out the pattern evolution, we provide end-users with a tool that allows them to use MUTate and OCEan in a user friendly way. This tool provides users with the following functionalities:

- Context Specification: the tool shows users the context information for which they have permission. It also allows a user to add new individuals corresponding to his/her information, modify them, and delete them if they are not used in the task model.
- Pattern Specification: the tool allows users to add, modify, or delete behaviour patterns by facilitating the information necessary to do this. If users do not want certain patterns to be executed during a period of time, they also can enable or disable specified patterns.

To provide user interfaces with this functionality, we have been inspired by the *Natural Programming* and *Visual Programming* end-user development approaches [14]. Based on these approaches, we have developed an interface for each one of the provided functionalities by using the SWT (Standard Widget Toolkit) [10] plugin in Eclipse. All of them follow a similar style. At the top of the interface, we guide users by using tabs that indicate the previous, current, and next steps to perform in order to achieve the corresponding goal. At the bottom of the interface, a text message is shown where help and explanations are provided to end-users. The rest of the interface is divided into two main frames: the left frame, which represents the work area where users specify the corresponding information; and the right frame, which shows users the information that they need for each step. Thus, end-users just need to select the information from the right frame and drag it to the proper location in the left frame.

Figure 7 shows a snapshot of an interface of our prototypical tool. This snapshot shows the first step for creating a new behaviour pattern in the task model: the specification of the context situation whose fulfilment triggers the execution of the pattern. The current tab and the other tabs that allow the user to navigate through the steps to be accomplished are shown at the top of the interface. On the right side, the context information that is available for Bob (who is using the interface) is shown in a tree form, going from more general information to more specific. In the work area, we facilitate the needed operators to form the context situation. Finally, the information area is provided at the bottom of the interface. This area shows what the pattern will do in a language close to natural language.

By using these interfaces, end-users can carry out the changes that they need. However, to preserve software quality characteristics, these changes are validated before they are applied to the system. Up to date, these changes are revised by

analysts; however, we are currently developing a tool to ensure the reliability of the system without the participation of analysts. This tool generates all the possible context situations that may play a part in the behaviour patterns changed by users. Then, it builds a set of tests that have to be passed by the system before these changes can be taken into account. If any of these tests are not passed, the system notifies the users about the possible mistakes so that they can be corrected. Finally, once the changes are validated, the tool updates the task model and the context model accordingly. The tool uses MUTate and OCEan to do this at runtime.

Also, we are currently working in incorporating the possibility of using machine learning algorithms to infer new behaviour patterns from user behaviour observation. These algorithms could automatically apply the mechanisms to evolve the patterns according to the inferred patterns. However, in this way we would not take into account users' desires since the repeated execution of an action does not imply that users want its automation. Thus, instead of this, the tool will present the inferred patterns to users once a month allowing them to modify or add these patterns if users regard them as appropriated.

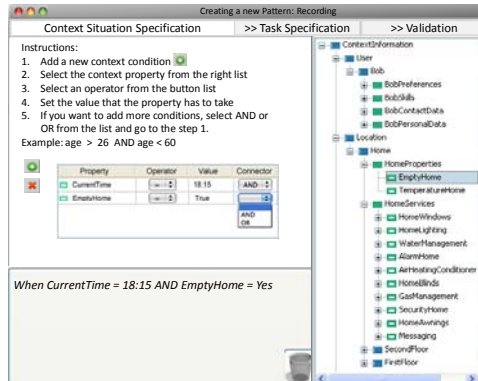


Fig. 7. Snapshot of the end-user tool

7 Evaluation

In order to evaluate our approach, we carried out two evaluation studies. First of all, we evaluated the feasibility of using models at runtime in our approach. Second, we performed a case study-based evaluation to test our approach in supporting user behaviour automation and evolution. To perform these evaluations, we use a Pentium 4, 3.0 GHz processor and 2 GB RAM with Windows XP Professional Edition SP3 and Java 1.5 installed. In addition, we used the implementation of OSGi *Prosynt Embedded Server* 5.2 [15], the EMF 2.3, EMFMQ 1.1, EMFMT 1.1 eclipse plugins, Pellet and the OWL API.

7.1 System Evolution by Using Models at Runtime

We evaluated the feasibility of using models at runtime in the evolution mechanisms. The model operations that they perform have to be efficient enough so that the system

response is not drastically affected. Thus, we performed an experiment to get the temporal cost of the operations of MUTate and OCEan that access to models. We used our context model and an empty task model to be randomly populated by means of an iterative process. The context model was populated with 100 new individuals each iteration, while the task model was populated with one new pattern whose task structure formed a perfect binary tree, varying its depth and the width of the first level each iteration.

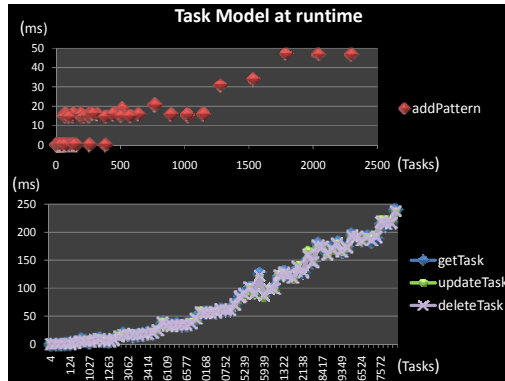


Fig. 8. Temporal cost of task model operations

After each iteration, we tested all the model operations of MUTate and OCEan 20 times and calculated the average temporal cost of each one. As an example, the operation over the context model with the highest temporal cost was the *getIndividual* operation, which took 7 milliseconds with 100 individuals and 10 milliseconds with 6000 individuals. This is because this operation has to get the individual by using a SPARQL query, which determines the temporal cost of the operation. Figure 8 shows the temporal cost of the task model operations with the highest cost. At the top of the figure, we show the time required to add a behaviour pattern according to the number of tasks. This operation took less than 50 milliseconds to add a pattern of 2296 tasks. At the bottom, we show the *getTask*, *updateTask* and *deleteTask* operations. Their costs are very similar since all of them make the same query to get the corresponding task. Even with a model population of 45612 tasks, these model operations provide a fast response (<250 milliseconds). Thus, the response time of using models at runtime is adequate.

7.2 A Case Study-Based Evaluation

As a case study-based evaluation of our approach, we developed a prototype to support the automation of the daily tasks that Bob and Sarah perform in their home (examples of these daily tasks have been presented in Section 3). To develop the case study, we specified the behaviour patterns that the system had to automate (with the participation of Bob and Sarah), using the proposed task model. To facilitate the participation of the users, we briefly explained the main concepts of the model and how we modelled the identified behaviour patterns. We found that this model was intuitive

enough for users and expressive enough to model all the identified automation requirements. However, the end-users found it a little difficult to understand some of the used temporal relationships.

To support the functionality needed to execute the system tasks of the patterns, we used the MDD strategy presented in [8] to obtain the code of the required services used to execute the *system tasks*. To evaluate the feasibility of our approach, we ran the system in real-life deployment sessions using *Prosys*. In the experimental set-up, a scale environment with real devices was used to represent the Smart Home [16].

In addition, we evaluated the usability of the tool for evolving the system automations. To do this, we arranged several sessions in which Bob and Sarah used the tool under our supervision. An observer sat side by side with them and measured (without intervening in their activity) their number of errors and their number of help requests. Since the users had participated in the modelling of the behaviour patterns, it was easy for them to change it. Specifically, 85% of the updates of the patterns were correctly performed; however, they needed particular help to properly establish the temporal relationships between the tasks of a new behaviour patterns. Finally, we measured errors in terms of “wrong clicks” which were, on average, 13% of the overall set of interactions.

8 Related Work

Current approaches for automating users’ behaviour patterns are predominantly machine learning-based approaches. Some examples are: the MavHome project [2] or the iDorm project [3]. The MavHome project uses prediction algorithms to identify common sequential patterns from data captured from the sensors of a smart home. From this learning, Mavhome builds a Markov model of user behaviour in which patterns are specified through a series of states linked by transitions with certain probabilities. If changes in user behaviour are detected by the algorithms or user feedback, the system is rebooted to obtain an improved Markov model using a new set of observations. The iDorm project models user behaviour by learning fuzzy rules that map sensor state to actuator readings representing inhabitant action. If changes in the user behaviour are detected by the algorithms, rules can be added, modified, and deleted. Thus, the evolution of these models is performed at a low level of abstraction. In addition, these techniques present some problems that can cause the loss of user acceptance of the system, e.g.:

- The algorithms used require **a great amount of training** data and make mistakes during the learning process, causing frustration to users;
- Lack of knowledge about user performed tasks may lead to **automating tasks which the user may not want automation** or reach generalizations in such a way that the automation becomes a burden on the user; e.g., the actions predicted by these algorithms for the *WakingUp* pattern (Section 3) would be: switch on the light, raise the blinds and switch the light off again, instead of directly raising the blinds.

In addition, these approaches could not predict some patterns of our case study because users do not perform these actions (Sarah does not record her favourite program when she is not at home), or they are not performed in the same context (the actions

to be automated when it starts to rain are performed when Sarah and Bob are at home, but not when they are out). In our approach, the evolution is performed at modelling level using concepts of a high level of abstraction, such as task or behaviour pattern, instead of adding states or rules as the above presented approaches. Moreover, users participate in the modelling of the system automation and in its evolution; therefore, the system only automates the tasks that users want automated and without requiring a learning process.

Other approaches try to specify proactive behaviour by using rule-based systems. Some examples are the proposal by García-Herranz et al. [17] and the proposal by Henriksen and Indulska [18]. These approaches are not focused on the automation of whole user behaviour patterns and do not cover the evolution of the rules once the system is running. In contrast, our approach automates whole user behaviour patterns by modelling them using a context-adaptive task model. This model considerably improves the expressivity of the above approaches by using the task concept and the temporal relationships among tasks. Thus, our approach allows users to view the automated actions as a whole task and not as isolated actions that are triggered when some conditions arise.

9 Conclusions and Further Work

In this work, we have presented and evaluated a novel approach for confronting the challenge of automating users' behaviour patterns and evolving them at runtime. These patterns are specified in a context-adaptive task model and automated by an engine (MAtE) that interprets the models to execute the patterns as specified. To deal with the evolution of these behaviour patterns, we have implemented MUTate and OCEan that are mechanisms capable of updating these models at runtime [6]. Since these models are interpreted at runtime by MAtE, as soon as they are modified, the changes are applied by the system. In addition, we provide an end-user tool that, by using MUTate and OCEan, allows users to change the patterns by using user-friendly interfaces. Thus, users can update the automated behaviour patterns without having to stop the system using this tool.

As further work, we plan to extend our end-user toolkit to provide interfaces that adapt according to the user preferences, skills and knowledge of the system [20]. According to this information (stored in the context model), the interfaces will provide users with the appropriate end-user techniques to change the automated behaviour patterns.

References

- [1] Neal, D.T., Wood, W.: Automaticity in Situ: The Nature of Habit in Daily Life. In: Bargh, J.A., Gollwitzer, P., Morsella, E. (eds.) *Psychology of action: Mechanisms of human action* (2007)
- [2] Cook, D.J., Youngblood, M., Heierman, I.E.O., Gopalratnam, K., Rao, S., Litvin, A., et al.: MavHome: An agent-based smart home. In: *PerCom 2003*, pp. 521–524 (2003)
- [3] Hagaras, H., Callaghan, V., Colley, M., Clarke, G., Pounds-Cornish, A., Duman, H.: Creating an Ambient-Intelligence Environment Using Embedded Agents. *IEEE Intelligent Systems* 19(6), 12–20 (2004)

- [4] Mens, T.: The ERCIM Working Group on Software Evolution: the Past and the Future. In: IWPSE-Evol 2009 (2009)
- [5] Bennett, K., Rajlich, V.: Software Maintenance and Evolution: A Roadmap. In: 22nd International Conference on Software Engineering, pp. 75–87 (2000)
- [6] Blair, G., Bencomo, N., France, R.B.: Models@run.time. *IEEE Computer* 42, 22–27 (2009)
- [7] OSGI, <http://www.osgi.org/>
- [8] Serral, E., Valderas, P., Pelechano, V.: Towards the Model Driven Development of context-aware pervasive systems. In: *Pervasive and Mobile Computing* (2009)
- [9] Serral, E., Valderas, P., Pelechano, V.: A Model Driven Development Method for developing Context-Aware Pervasive Systems. In: Sandnes, F.E., Zhang, Y., Rong, C., Yang, L.T., Ma, J. (eds.) *UIC 2008*. LNCS, vol. 5061, pp. 662–676. Springer, Heidelberg (2008)
- [10] Eclipse Platform, <http://www.eclipse.org>
- [11] Smith, M.K., Welty, C., McGuinness, D.L.: OWL Web Ontology Language Guide. W3C Recommendation February 10 (2004), <http://www.w3.org/TR/owl-guide/>
- [12] SPARQL Query Language (2008), <http://www.w3.org/TR/rdf-sparql-query/>
- [13] Sirin, E., Parsia, B., Grau, B.C., Kalyanpur, A., Katz, Y.: Pellet: A practical OWL-DL reasoner. *Journal of Web Semantics* (2007)
- [14] Pérez, F., Valderas, P.: Allowing End-users to Actively Participate within the Elicitation of Pervasive System Requirements through Immediate Visualization. In: *REV* (2009)
- [15] Prosys, <http://www.prosys.com/>
- [16] EIB, <http://www.knx.org/>
- [17] García-Herranz, M., Haya, P.A., Esquivel, A., Montoro, G., Alamán, X.: Easing the Smart Home: Semi-automatic Adaptation in Perceptive Environments. *Journal of Universal Computer Science* 14 (2008)
- [18] Henricksen, K., Indulska, J.: A Software Engineering Framework for Context-Aware Pervasive Computing. In: *PerCom* (2004)
- [19] Lieberman, H.: Programming by example (introduction). *Commun. ACM* 43(3), 72–74 (2000)
- [20] Pribeanu, C., Limbourg, Q., Vanderdonckt, J.: Task Modelling for Context-Sensitive User Interfaces. In: Johnson, C. (ed.) *DSV-IS 2001*. LNCS, vol. 2220, pp. 49–68. Springer, Heidelberg (2001)
- [21] Shalloway, A., Trott, J.R.: *Design Patterns Explained: A New Perspective on Object-Oriented Design*. Addison-Wesley, Reading (2004)
- [22] Serral, E.: Automating User Behaviour Patterns. Technical Report, PROS - UPV (2009), <http://oomethod.dsic.upv.es/labs/media/techreports/TechnicalReport-AutomatingBP.pdf>

Interaction-Driven Self-adaptation of Service Ensembles

Christoph Dorn and Schahram Dustdar

Distributed Systems Group
Vienna University of Technology
1040 Vienna, Austria
lastname@infosys.tuwien.ac.at

Abstract. The emergence of large-scale online collaboration requires current information systems to be apprehended as service ensembles comprising human and software service entities. The software services in such systems cannot adapt to user needs based on autonomous principles alone. Instead system requirements need to reflect global interaction characteristics that arise from the overall collaborative effort. Interaction monitoring and analysis, therefore, must become a central aspect of system self-adaptation. We propose to dynamically evaluate and update system requirements based on interaction characteristics. Subsequent re-configuration and replacement of services enables the ensemble to mature in parallel with the evolution of its user community. We evaluate our approach in a case study focusing on adaptive storage services.

1 Introduction

Over the past years we have observed a trend towards online collaboration. Web sites for social networking (e.g., Facebook, LinkedIn), collaborative tagging (e.g., Digg, Del.ici.us), content sharing (e.g., Youtube), or knowledge creation (e.g., Wikipedia) have attracted millions of users. People increasingly utilize such tools to pursue joint interests and shared goals.

The scientific community in particular comes to profit from a tight interweaving of social networks and technological networks [1]. Barabasi [2] highlights the tendency for research teams to grow in size. Guimera et al. [3] describe the impact of social network dynamics on team performance. Scientific teams emerge in an ad-hoc fashion, gather the persons with the required expertise, conduct research, and dissolve again.

The scientific community is one example where collaboration emerges in large-scale, heterogeneous systems. Kleinberg [4] notices the opportunity to observe the dynamics and complexity of such systems that arise from the convergence of social and technical networks in general. We refer to such systems as *service ensembles*.

The scale of online collaborations prevents a single ensemble entity from obtaining a complete picture of the overall service ensemble. Simultaneously, services lack adequate mechanisms that derive global-level interaction characteristics. Services will exhibit poor performance and slow reaction to a changing environment: promising collaborations dissolve prematurely; helpful services remain unavailable as nobody becomes aware of the demand. As a result, enabling interaction-driven adaptivity is a prime concern in evolving information systems.

Several challenges need to be addressed before a service system can adapt to global-level ensemble interaction characteristics. Users apply both direct and indirect interaction means as well as exchange services as they feel suitable. Any interaction metric needs to abstract from these low-level interaction details and has to derive user proximity across activities, services, and shared artifacts. Any distance measurement needs to take into account also the focus and magnitude of interaction among humans, among services, and between humans and services. Existing work on social network analysis (e.g., [516]) provides insight into the community structure. Deriving requirements from this data, however, is non-trivial.

In this paper we provide models and mechanisms to align configuration and provided functionality of software services with the ensemble's interaction structure. The main goal is to establish the set of required service capabilities - i.e., what functionality and adaptability services need to support - but *not* how services achieve specific adaptation. Our mechanisms, for example, discover that ensemble users tend to interact in co-located groups, and derive requirements demanding location-aware services. We do not, however, specify how these services exploit information about the ensemble interaction structure.

Specifically, we extend the traditional autonomic feedback loop with a secondary loop to monitor and analyze interactions across the complete service ensemble (Section 2). Interaction analysis relies on our bipartite interaction graph that tracks relations between humans and services across activities and artifacts. Distance measurements on this graph consider the interaction focus and global significance of individual ensemble entities (Section 3.2). Significant changes in the users' interaction structure trigger an update of the system's configuration via requirements rules. These rules take the interaction structure to derive necessary service capabilities (Section 3.3). Finally, we validate our approach (Section 4) based on the case study presented in the following section (Section 1.1).

1.1 Motivating Scenario

We observe a service ensemble comprising a group of 20 scientists working together closely on a research proposal. They utilize various services to coordinate their work, communicate on- and off-line, manage documents and figures, update and revise financial tables for example. In this scenario, we focus on the requirements and adaptation particular to storage services.

In such a relatively small service ensemble already, individual users find it considerably hard to gain an overview of the underlying interaction structure and the system requirements emerging from that structure. During the proposal drafting phase (Fig. 1a) users will collaborate on the various proposal sections in an ad-hoc manner, giving rise to short-lived interactions between a subset of users that dissolve again. A storage service will, therefore, need to provide maximum cooperation flexibility. A suitable service enables simple file storage and exchange between participants without requiring extensive configuration of file versioning or access control.

Let us assume, the proposal is accepted and the regular project phase commences (Fig. 1b). As the number of project participants rises, we expect stable subgroups to form that work on various project tasks. Subsequently, users require services that

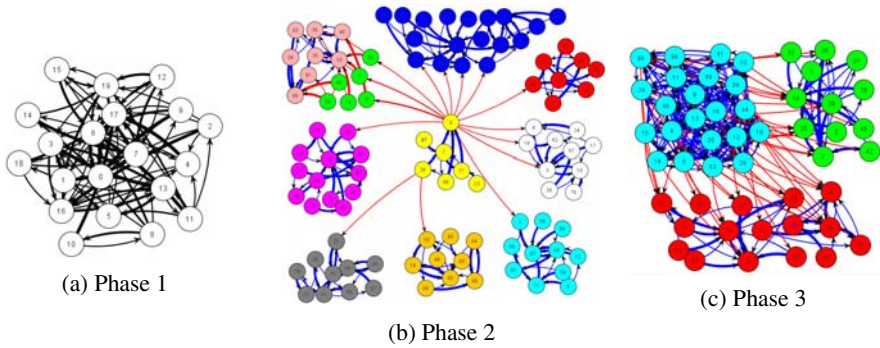


Fig. 1. Scenario: (Colors online) - (a) Phase 1: single cluster, dense interactions. (b) Phase 2: intra organizational interaction with ties to the coordinator. (c) Phase 3: task-centric interaction, feedback from T1 (upper left) to T2 (right) and T3 (bottom). Line thickness indicates the amount of interactions. Red links denote cross-cluster interaction, blue links represent intra-cluster interactions.

reflect this structure. A suitable service, for example, reduces information overload of individual users by focusing the interactions to users within a subgroup. Other adaptation dimensions include services keeping content within user proximity, limiting resource access to users of the same organization, or providing notifications to users with specific roles or skill.

Towards the end of the project (Fig. 1c), users have to collaborate across organizational boundaries again to integrate their research results and prototypes. Communication remains mostly within the scope of the various tasks, with users involved in demonstration activities giving feedback to users working on scientific dissemination as well as to users planning industrial exploitation. Derived requirements demand a shift from organization-centric to task-centric services.

Throughout the evolution of the ensemble, no single user obtains a complete overview of interactions and respective system requirements. Continuous interaction monitoring and analysis is therefore quintessential to enable software services to evolve in line with the overall ensemble structure.

2 From Autonomous to Evolving Service Ensembles

A service ensemble consists of humans (i.e., the users), software services, and a service infrastructure. The infrastructure typically provides a service registry and a graphical user portal. The registry contains a comprehensive set of service descriptions. At any time, however, only a subset of those services is actively deployed in the ensemble. We refer to the complete set of software elements as the *system*. Whereas an autonomous system aims to fulfill a static set of requirements, an evolving system replaces and reconfigures its parts (i.e., services) to match the dynamically changing requirements of the ensemble's user base. Here, dynamic system requirements ultimately cause the replacement, deployment, removal, or reconfiguration of services.

Most traditional autonomous systems follow a Monitor- Analyze- Plan- Execute+ Knowledge (MAPE-K) approach [7,8] - although these phases are sometimes named differently [9,10]. The core cycle in Fig. 2 visualizes the basic MAPE-K steps 1a, 2a, 3a, and 4. In a service ensemble, *System Monitoring* observes changes in services capabilities and QoS parameters, service dependencies, and service load. *System Analysis* compares the current requirements with the current system state. *System Planning* initiates changes when capabilities or QoS degrade, with *System Execution* enforcing the replacement and reconfiguration of the actual service instances.

We introduce a parallel (incomplete) MAPE-K cycle to turn an autonomous system into an evolving system (Fig. 2). *Ensemble Monitoring* (1b) observes the interactions and properties of the user set. *Ensemble Analysis* (2b) extracts interaction patterns and changes thereof. Given the structure of collaboration, user properties, shared artifacts, and service usage, *Ensemble Planning* (3b) updates the system requirements to match the dynamic user characteristics. Direct *Ensemble Control* of human behavior is not possible, thus the outer MAPE-K cycle remains incomplete. Any desirable impact is induced via system reconfiguration. Consequently, the system MAPE-K cycle reacts not only to system events but also to updates of the system requirements.

Interactions play a most important part in our approach. Extraction of the ensemble structure from interactions not only reveals relevant system requirements but also provides the additional information on the system level to analyze service dependencies in more detail. In the following sections we will, therefore, focus on the interaction-centric models and mechanisms for realizing steps 1b, 2b, 3b, and 3a. Specific adaptation and

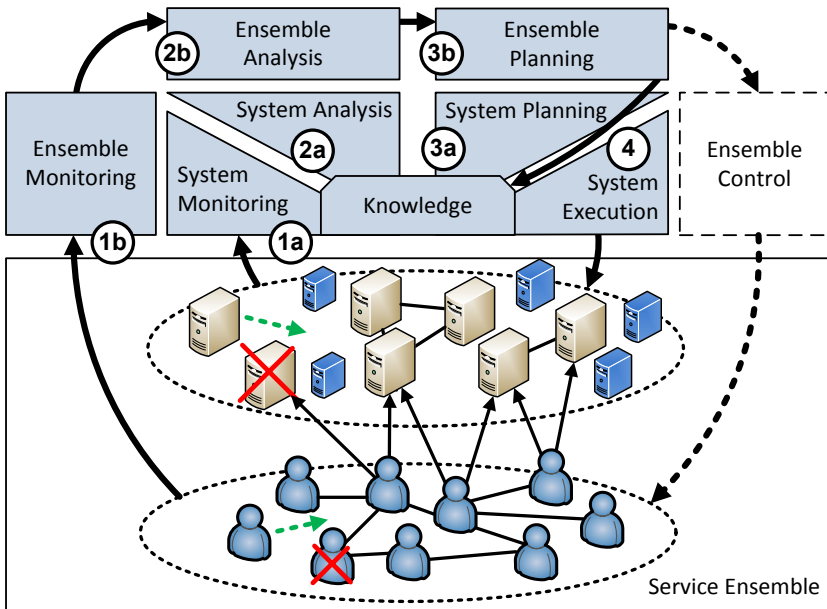


Fig. 2. Approach: from autonomous to evolving service ensembles

reconfiguration mechanisms in step 4 or within individual services remain outside the scope of this paper.

3 Interaction-Based Continuous Adjustment

3.1 Interaction Monitoring

The key to extracting meaningful information from interaction data is monitoring the context in which the various interactions take place (Fig. 2 Step 1b). Introduced in previous work [11], the principle interaction event is an action that describes the co-occurrence of users, services, and artifacts (e.g., documents) in a common activity at the given time. In this manner, actions describe interaction among users, among services, and between users and services.

Additional information sources from the environment such as user profiles available from online social platforms or enterprise directories (in corporate settings) complement action events by putting them in perspective. The raw events are captured by distributed logging [12], monitoring [13], or sensing [14] mechanisms. Within individual services, autonomous toolkits (e.g., [15][16][17]) provide monitoring techniques to provide an up-to-date view on capabilities and QoS values.

3.2 Interaction Structure

We propose to integrate multiple interaction types into a single distance measurement to reflect more accurately the tight inter-dependency between humans and services (Fig. 2 Step 2b). Most existing approaches to interaction analysis observe one interaction type only, for example, either emails, friendship links, or posting threads. Furthermore, little attention is put on the focus and magnitude of links between entities, i.e. the distribution of re-occurring interactions across established links, respectively the amount of re-occurring interactions. The key aspect of our approach is weighting the entity's local interactions according to its global significance.

We map the captured actions into a bipartite interaction graph $\mathcal{AG}_2(V, E)$. Our bipartite graph defines two vertex categories: actions (\mathcal{V}_a) and ensemble entities (\mathcal{V}_e). Edges are undirected and exist only between vertices of different category. Fig. 3a displays an example bipartite interaction graph comprising actions (a1, a2, a3, a4) and ensemble type user (u1, u2, u3, u4), activity (t1, t2), artifact (o1, o2), and service (s1, s2, s3). An action's weight $w(a)$ corresponds to the number of times that action has occurred. Two actions are considered identical when they exhibit the same set of involved ensemble entities.

We measure the distance between two vertices of the same ensemble entity type by aggregating their involvement in shared *and related* actions. The distance, for example, between two users is based on actions involving joint activities, joint resources, and joint artifacts. In the example graph (Fig. 3a) users u2 and u3 become linked to user u4 through the shared artifacts o1 and o2. The process of calculating the distance between two entities requires determining the similarity between any two actions first.

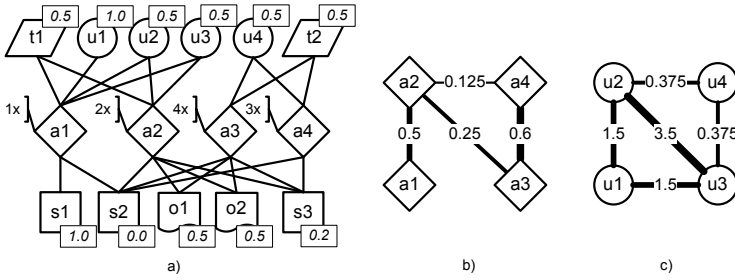


Fig. 3. Bipartite interaction graph (a) combining user, activities, artifacts, service, and actions; action similarity graph (b); user distance graph (c)

Action similarity. For each pair of actions (a_i, a_j) , we consider the amount of shared entities. The set of common entities, however, is insufficient to accurately establish the distance between two actions. We also consider the global significance of a shared entity in contributing to the overall distance measurement. For example, service s_2 has edges to all actions. It should not be considered as it does not add any information to distinguish the similarity of two actions.

We apply Shannon’s entropy definition [18] $H(w) = -\sum(w * \log(w))$ to describe the information content of an entity’s edge set, thus deriving the global significance $sig(v)$. The significance of entity v is defined as¹:

$$sig_v = 1 - \frac{\log(deg(v))}{\log(|a|)} \quad \forall \quad v \in \mathcal{V}_e \text{ and } deg(v) > 1 \quad (1)$$

where $deg(v)$ is the degree of vertex v and $|a|$ denotes the total number of actions. The normalization yields a significance value in the interval $[0, 1]$. When a vertex v links to all actions, it exhibits no focus (i.e., minimum entropy) and thus significance becomes 0. Vertices with one neighbor yield $sig(v) = 1$. Italic numbers in Fig. 3a provide the global significance values for the example entities.

The Jaccard similarity metric determines the similarity of two actions based on their common entities. We consider, however, only those entities which exceed a given significance threshold γ . The Jaccard similarity is defined as 1 minus the difference between set union and set intersection, divided by the set union:

$$J_\delta = 1 - \frac{|A \cup B| - |A \cap B|}{|A \cup B|} \quad (2)$$

where sets A and B contain the references to users, services, activities, and artifacts when comparing two actions a_i and a_j . J_δ becomes 1 when the two sets contain the same entities and yields 0 when the two sets are completely disjoint. The complete action similarity graph is denoted $\mathcal{AG}_{action}(\mathcal{V}_a, E_a)$ shown in Fig. 3b for the example bipartite graph with $\gamma = 0.3$.

¹ Reduced from $sig_v = 1 - \frac{-deg(v) * (\frac{1}{deg(v)} * \log(\frac{1}{deg(v)}))}{\log(|a|)}$.

Interaction distance. The distance measurement between two ensemble entities of same type takes two aspects into account:

- The distance between two entities decreases with increasing number of shared actions. A large amount of shared actions, however, does not necessarily imply low distance.
- The distance between two entities decreases with increasing re-occurrence of interactions. The more often two entities have interacted (i.e., high action weight) the closer they become. Two entities having interacted only once in the scope of three actions, yield higher distance than two entities having interacted 10 times via a single action. Users u_2 and u_3 , for example, yield lower proximity than u_2 and u_1 .

In the case of user distance, we take measurements across services, activities, and artifacts to derive interaction distance for direct and indirect communication. One user, for example, uploads a document, while another user downloads it. This process consists of two actions, each involving a single user, and the same document. Total distance $dist_{total}(u_i, u_j)$ between two users derives, thus, not only from shared actions ($dist_1(u_i, u_j)$), but also related actions ($dist_2(u_i, u_j)$):

$$dist_{total}(u_i, p_j) = dist_1(u_i, u_j) + dist_2(u_i, u_j) \quad (3)$$

We define two action sets. Set A_i contains all neighboring actions of u_i . There exists an analogous set A_j . The shared action distance is defined as the sum of actions weights w :

$$dist_1(u_i, u_j) = \sum_n^{|N|} w(n) \quad (4)$$

where $N(u_i, u_j)$ is the interaction of A_i and A_j .

The related actions distance $dist_2(u_i, u_j)$ defines the edge set $E_a(a_k, a_l)$ in the action similarity graph \mathcal{AG}_{action} such that $a_k \in A_i$, $a_l \in A_j$, and $a_k \neq a_l$.

$$dist_2(u_i, u_j) = \frac{\sum_E (w_e * \min[w(a_k), w(a_l)])}{|E|} \quad (5)$$

The set of pairwise distance measurements generates the entity interaction distance graph $\mathcal{AG}_{user}(\mathcal{V}_P, E_P)$ — here specific to users — required in the subsequent analysis step (Section 3.3). The corresponding graph for the four users in the example is displayed in Fig. 3c.

3.3 Interaction-Driven Requirements Adjustment

System requirements need to reflect the significant changes in the ensemble's interaction structure (Fig. 2 Step 3b). One important structural property of service ensembles is the number and the size of subgroups that emerge from the interaction distance graph and what factors cause this structure. Graph analysis based on interaction affinities (e.g., [19]) or community detection algorithms (e.g., [20] or [21]) describes the underlying

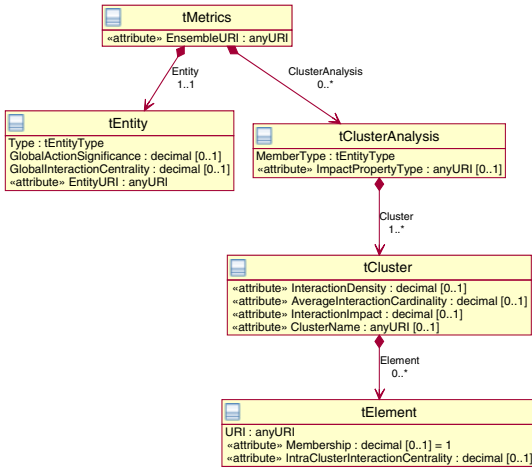


Fig. 4. Excerpt of the ensemble model specifying the interaction analysis result

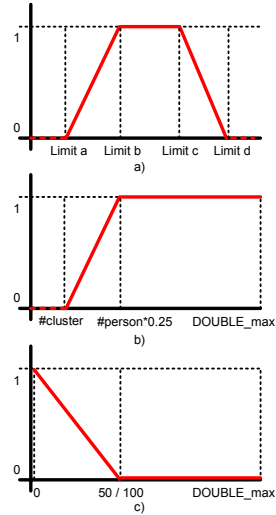


Fig. 5. Utility function: (a) generic form, (b) fct. utilized in the example rule, and (c) fct. used in the evaluation

structure from which we subsequently derive appropriate system requirements. Fig. 4 displays the UML representation of the interaction analysis data model. The model lists the set of involved entities of type users, activities, artifacts, and services. It states for each entity the centrality in the overall interaction graph and the global significance value applied during distance calculation. Each *ClusterAnalysis* element describes the observed entity type and the context property that most likely caused the clusters to form (e.g., location, organization, task, or role). Each detected *Cluster* states the interaction-centric metrics such as density, average cardinality, interaction direction (interaction within the cluster $[0 \rightarrow 1]$ or external $[0 \rightarrow -1]$), and cluster name (e.g., the organization’s name, particular location, or role identifier). The membership degree specifies the degree to which an entity belongs to a particular cluster, and its centrality within that cluster.

Our previous work [22] introduced a capability model for services. The capability model specifies non-functional properties such as storage space per user, cost, or a service’s adaptability of resource access to location or organization context. System requirements are expressed in terms of necessary capabilities. Coupling these system requirements with rules enables us to dynamically adjust the system configuration to match the interaction structure of the service ensemble.

The example requirement rule in Listing 1 demands common storage space for each tightly connected subgroup (i.e., a cluster with interaction impact between 0 and 1). A requirements rule consists of four parts:

1. the interaction-based condition triggering the rule. In the example we wait for two or more clusters of users. Simultaneously, we limit that number to a quarter of the total number of users to mitigate potential errors during the clustering process.
2. the computation of utility function parameters (optional).
3. the generation of the desired capability constraints. A constraint specifies the desired capability, the utility function type, utility function parameters, and the importance of the constraint. Fig. 5 visualizes the general utility function (a), and the instance (b) utilized in requirement *req2*. In this example, we require a storage service to be aware of the source causing the clustering (e.g., clusters mapping to organization, locations, or roles). We also demand that the service supports as many folders as there are clusters, and that each folder can be accessed simultaneously by at least as many users as contained in the largest cluster.
4. the update of the knowledge component of the system's MAPE-K cycle.

```

1 rule "StorageClusterSupport"
2 dialect "java"
3 when
4     em : EnsembleMetrics ((UserCA.clusters.size() >= 2) &&
5                           (UserCA.clusters.size() <
6                             (0.25*em.getEntities().getUsers().size()))
7 then
8     int clusterCount = UserCA.clusters.size();
9     int maxClusterSize = 0;
10    for (int i = 0; i<clusterCount; i++)
11    {
12        int size = UserCA.cluster.get(i).elements.size()
13        if (size > maxClusterSize) maxClusterSize = size;
14        if (UserCA.cluster.get(i).getImpact() < 0) clusterCount--;
15    }
16    URI impactTypeURI = UserCA.getImpactPropertyType();
17    TCapabilitySelectionRequirement req1 = RequirementsFactory.getSelectConstraint(
18        "Require Adaptability to Interaction Cluster Impact",
19        URIs.SELECTCAP_ADAPTABILITY,
20        new String[]{impactTypeURI},
21        ChoiceUtilityHigh.class.getSimpleName(), ChoiceUtilityHigh.UTILITY_TYPE
22        0.8d);
23    TSimpleDecimalConstraint req2 = RequirementsFactory.getConstraint(
24        "Support folders for all clusters",
25        URIs.CAP_ResStorage, URIs.PROP_MaxFoldersPerAccount_ResStorage,
26        ValueUtility.UTILITY_TYPE, ValueUtility.class.getSimpleName()
27        new double[]{ clusterCount, 0.25*em.getEntities().getUsers().size(),
28                      Double.MAX_VALUE, Double.MAX_VALUE},
29        0.5d);
30    TSimpleDecimalConstraint req3 = RequirementsFactory.getConstraint(
31        "Support sufficient concurrent users within folders",
32        URIs.CAP_ResStorage, URIs.PROP_MaxConcurrentUsersPerFolder_ResStorage,
33        ValueUtility.UTILITY_TYPE, ValueUtility.class.getSimpleName()
34        new double[]{ maxClusterSize*0.9, maxClusterSize*2,
35                      Double.MAX_VALUE, Double.MAX_VALUE},
36        0.4d);
37    knowledge.addRequirement( em.getEnsembleURI(), req1);
38    knowledge.addRequirement( em.getEnsembleURI(), req2);
39    knowledge.addRequirement( em.getEnsembleURI(), req3);
40 end

```

Listing 1. Example DROOLS requirement rule generating three storage capability constraints when interaction clusters have emerged

3.4 System Planning and Execution

The new capability constraints need to be matched against the current system configuration (Fig. 2 Step 3a). Along with the capability model, we have introduced a constraint matching and selection algorithm in [22].

The ultimate execution of adaptation plan involves removal, replacement, or deployment of new services. Dynamic invocation frameworks such as VRESCO [23] provide the appropriate mechanisms to exchange service endpoints during runtime. We rely on VRESCO's runtime SOAP message interceptors to perform necessary message transformation operations. This addresses potential service interoperability problems and allows interaction-based requirements analysis to work on a common capability model without having to take heterogeneous service interfaces into account.

4 Case Study Evaluation

We base our evaluation on the three phases of the motivating scenario in Section 1.1. Throughout the evaluation we focus on a limited number of requirements and service specific storage capabilities for sake of clarity. Table 1 (upper part) lists the monitored capabilities and their support by ten example services (S1 ... S10). The underlying interaction data (Fig. 1a-c) for the three phases are synthetic, but yield the same global interaction characteristics we observed in our past research projects. Excerpts from the corresponding bipartite action graphs are visualized in Fig. 6a-c. Table 7 provides the results of the cluster analysis subsequently applied to the rules in Listing 1. We also apply an additional cost constraint. The respective cost utility function yields maximum utility at \$0, and minimum utility from \$50 (Phase 1), respectively \$100 (Phase 2 and 3), onwards. Table 1 (lower part) lists the final service ranking result, while the right part supplies the parameters for the respective utility functions. The ranking process utilizes the Logic Scoring of Preferences (LSP) algorithm [24] - a weighted sum of utility scores.

Phase 1: During the project proposal drafting the 20 initial participants interact closely without being affected by location, their role, or organizational boundaries (Fig. 1a). The bipartite action graph excerpt for phase 1 (Fig. 6a) has most users involved in every action using an email service ($sig(S0) = 0$). S0 is thus not considered for action similarity measurements. The corresponding interaction cluster analysis for users does not reveal emergence of distinct subgroups, thus the only *task* of preparing the proposal becomes the best explanation for the dense interaction graph (Fig. 7). As outlined in the motivation, services which are cheap and offer the minimum required amount of support for users and folders rank highest. Here, we select service S2.

Phase 2: The participant number raises sharply shortly after the project kickoff. In our scenario, ten organizations allocate between 6 and 18 members for a total number of 100 participants. Project coordinator is Organization 10, exhibiting a neutral impact due to a balance of organization internal and external interaction. Most members keep interaction organizations internal, with exception to Organization 2, who's members focus purely on coordinating members in Organization 8 (Fig. 1b). The underlying bipartite

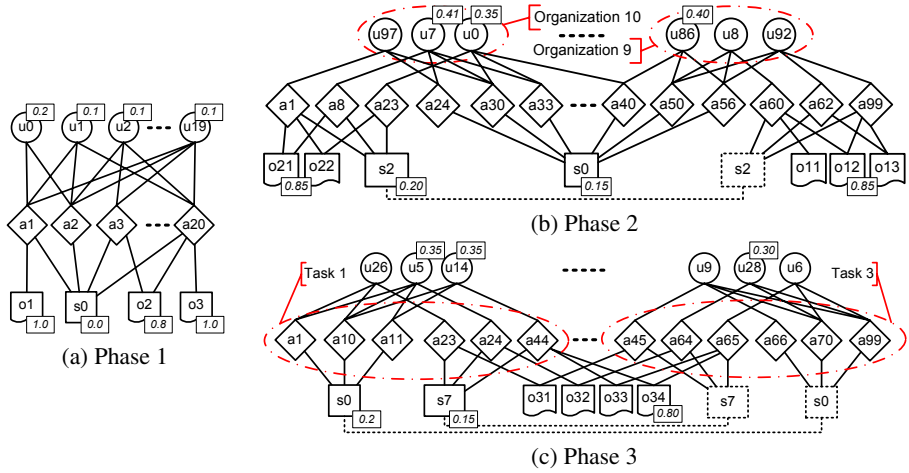


Fig. 6. Bipartite action graph excerpt - activities omitted and services s2 and s7 split for the sake of clarity. Significance values are given only for a subset of entities and are based on the complete bipartite graph.

| Phase | Property | ClusterName | Impact | Size |
|-------|----------|---------------|--------|------|
| 1 | Task | Proposal | 1 | 20 |
| 2 | Org. | Org. 1 | 1 | 9 |
| | | Org. 2 | -1 | 6 |
| | | Org. 3 | 1 | 10 |
| | | Org. 4 | 1 | 10 |
| | | Org. 5 | 1 | 13 |
| | | Org. 6 | 1 | 18 |
| | | Org. 7 | 1 | 11 |
| | | Org. 8 | 1 | 8 |
| | | Org. 9 | 1 | 8 |
| | | Org. 10 | 0 | 7 |
| 3 | Task | Dem. (T1) | 0.83 | 22 |
| | | Dissem. (T2) | 1 | 12 |
| | | Exploit. (T3) | 1 | 16 |

Fig. 7. Interaction Cluster Analysis

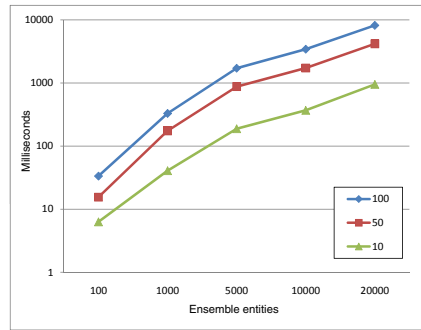


Fig. 8. Interaction analysis performance measurements (in milliseconds) for service ensemble graphs exhibiting 100 to 20000 entities, featuring 10, 50, and 100 properties

action graph (Fig. 6b) emphasizes the uptake of service S2 in Phase 1 as an indirect interaction tool via shared artifacts. Simultaneously, users continue to utilize the email service (S0), albeit, mostly for organization internal communication. Both services thus provide useful (but low significance) information for the calculation of action similarity. The subsequent cluster analysis reveals a strong impact of the organizational topology on the interaction structure (Fig. 7). The same requirement rule of Phase 1 will now request services exhibiting adaptability to the organizational structure, while

Table 1. Storage service capabilities, utility function parameters, constraint weight, and ranking results (top services in bold font)

| | S1 | S2 | S3 | S4 | S5 | S6 | S7 | S8 | S9 | S10 | Para 1 | Para 2 | Para 3 | w |
|-------------------|----|------------|-----------|----|------------|-----|------------|-----------|-----------|-----|--------|---------|---------|-----|
| Org-adaptability | - | - | x | - | - | - | x | - | x | x | - | x | - | 0.8 |
| Loc-adaptability | - | - | - | x | - | - | x | - | - | x | - | - | - | 0.8 |
| Task-adaptability | - | - | - | - | x | - | - | x | x | x | - | - | x | 0.8 |
| Role-adaptability | - | - | - | - | - | x | - | x | x | x | - | - | - | 0.8 |
| FoldersPerAccount | 5 | 5 | 20 | 50 | 50 | 100 | 50 | 250 | 100 | 250 | 1;5 | 9;25 | 3;12.5 | 0.5 |
| UsersPerFolder | 10 | 50 | 100 | 20 | 50 | 10 | 10 | 50 | 100 | 100 | 18;40 | 16.2;36 | 19.8;44 | 0.4 |
| Costs \$ | 0 | 0 | 60 | 50 | 20 | 100 | 50 | 30 | 90 | 200 | 0;50 | 0;100 | 0;100 | 0.5 |
| Rank Phase 1 | 56 | 100 | 44 | 3 | 78 | 0 | 0 | 36 | 44 | 44 | | | | |
| Rank Phase 2 | 0 | 25 | 98 | 49 | 34 | 41 | 100 | 46 | 94 | 76 | | | | |
| Rank Phase 3 | 0 | 19 | 42 | 44 | 100 | 10 | 18 | 98 | 89 | 71 | | | | |

supporting the required number of users within each cluster. In our scenario, we consequently switch from service S2 to S7.

Phase 3: The number of project participants decreases to 50 towards the end of the project (Fig. 1c). The bipartite action graph (Fig. 6c) draws attention to the task internal direct communication, and cross-task indirect interaction via shared artifacts. We also note the switch of storage services as recommended in the previous phase (S7 instead of S2). Interaction analysis now identifies the task structure as the most likely factor to give rise to clusters (Fig. 7). Derived requirements demand a change from organization-centric to task-centric services. Subsequently, we replace service S7 with service S5.

Discussion. During the scenario users switch between different communication types (direct such as email, and indirect such as shared artifacts). They also exchange services. Nevertheless, we are still able to track those interactions and thus derive accurate user proximity as we map all actions onto our bipartite graph. The significance of particular entities changes throughout the scenario. Dynamically assessing their impact on similarity guarantees accurate distance measurements during all scenario phases.

A global-level view on interactions is vital to establish the underlying interaction structure such as emerging clusters. The interaction analysis detects the cause of clustering, such as organizational, spatial, or task-related constraints. The respective requirements demand exactly those services that are able to adapt to and support these constraints. Observation of individual user or service properties alone can never reveal such requirements.

The performance evaluation of the interaction analysis algorithm for various configurations of service ensemble size and property count is displayed in Figure 8. The algorithm exhibits exponential runtime behavior as expected for a growing interaction network. However, even for the most complex scenario of 20000 entities each exhibiting 100 properties, the interaction impact is derived within 10 seconds which is sufficiently fast.

5 Related Work

Interaction-based adjustment of service systems builds on concepts and techniques from the autonomic computing domain and service adaptation community. Current general-purpose autonomic techniques and toolkits (e.g., [15][16][17]) primarily apply context about the software environment. These frameworks adhere to the basic MAPE-K feedback loop, limiting the application of user context to properties such as location or device.

Jennings et al. [25] propose an architecture for autonomic management of communication networks that comes close to our approach of monitoring a large-scale system. No human interaction characteristics, however, are considered.

Colman [26] proposes a hybrid approach to self-organization services through hierarchical structuring of autonomic managers and services. An autonomic manager monitors and controls all composed services, but lacks insight into the interaction characteristic of the environment. In our work, adaptation to the environment is sensed and analyzed in parallel to regular system monitoring. Any effect on the environment, however, can only be achieved through the system MAPE-K cycle.

The main shortcoming of the presented autonomic computing approaches is their fixed set of requirements. These remain valid throughout the system's lifetime or require manual configuration. In large-scale ensembles neither the services' providers nor individual users can grasp the system functionality required by the overall ensemble.

Specifically in service-oriented systems, extensive research efforts focus on service selection based on Quality-of-Service (QoS) attributes (e.g., [27][28][29]), context information (e.g., [30][31]), or trust (e.g., [32][33][34]). Recently, Skopik et al. [35] extended the notion of trust to cover both humans and services in mixed service-oriented systems.

These approaches consider only a subset of an ensemble context during composition. Adaptation and selection criteria usually build upon QoS metrics or context about the service execution environment. Involved user context comprises mostly location, devices, and preferences. None of these efforts evaluates the complex user interaction structure within the service ensemble.

Social network analysis investigates the interaction characteristics of online communities. Information (e.g., [5][6]) that potentially serves as context for adaptation is usually not available in near-realtime, nor does it include aspects beyond human-to-human communication. Our approach derives similarity from users' involvement in joint activities, use of common services, or modification of shared artifacts. Additionally, we take into consideration the global significance of individual elements when computing distance measurements.

6 Conclusion and Outlook

We have demonstrated the importance of monitoring global-level user interaction characteristics to adapt system requirements. Based on captured user actions, our bipartite interaction graph enables proximity measurements across users, activities, resources, and artifacts. Subsequent interaction analysis identifies those properties that determine the emergence of clusters. Clustering results and other interaction metrics provide

meaningful data for interaction-centric requirements rules. These rules generate service capability constraints which serve as system requirements. Final matching against service capability profiles provides recommendations for system reconfiguration.

The current approach evaluates the strongest interaction impact for the complete service ensemble. We plan to introduce context-dependent analysis that observes different parts of the ensemble independently to simultaneously adapt to multiple impact factors. The second line of future research activities focuses on deriving and describing reoccurring global-level interaction patterns. So far changes in the interaction structure cannot be anticipated. We expect that prediction techniques combined with patterns enable the early detection of structural changes and facilitate such transitions through a-priori service reconfiguration.

Acknowledgment

This work has been partially supported by the EU STREP project Commius (FP7-213876).

References

1. Jones, B.F., Wuchty, S., Uzzi, B.: Multi-University Research Teams: Shifting Impact, Geography, and Stratification in Science. *Science* 322, 1259–1262 (2008)
2. Barabasi, A.L.: SOCIOLOGY: Network Theory—the Emergence of the Creative Enterprise. *Science* 308(5722), 639–641 (2005)
3. Guimera, R., Uzzi, B., Spiro, J., Amaral, L.A.N.: Team Assembly Mechanisms Determine Collaboration Network Structure and Team Performance. *Science* 308(5722), 697–702 (2005)
4. Kleinberg, J.: The convergence of social and technological networks. *Commun. ACM* 51(11), 66–72 (2008)
5. Bird, C., Gourley, A., Devanbu, P., Gertz, M., Swaminathan, A.: Mining email social networks. In: *MSR 2006: Proceedings of the 2006 international workshop on Mining software repositories*, pp. 137–143. ACM Press, New York (2006)
6. Valverde, S., Solé, R.V.: Self-organization and hierarchy in open source social networks. Technical report, DELIS – Dynamically Evolving, Large-Scale Information Systems (2006)
7. Kephart, J.O., Chess, D.M.: The vision of autonomic computing. *Computer* 36(1), 41–50 (2003)
8. IBM: An architectural blueprint for autonomic computing (2005)
9. Parashar, M., Hariri, S.: Autonomic computing: An overview. In: Banâtre, J.-P., Fradet, P., Giavitto, J.-L., Michel, O. (eds.) *UPP 2004. LNCS*, vol. 3566, pp. 257–269. Springer, Heidelberg (2005)
10. Dobson, S., Denazis, S., Fernández, A., Gäiti, D., Gelenbe, E., Massacci, F., Nixon, P., Saffre, F., Schmidt, N., Zambonelli, F.: A survey of autonomic communications. *ACM Trans. Auton. Adapt. Syst.* 1(2), 223–259 (2006)
11. Schall, D., Dorn, C., Dustdar, S.: Viacar - enabling self-adaptive collaboration services. In: *34th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA)*. IEEE Computer Society, Los Alamitos (2008)

12. Dorn, C., Truong, H.L., Dustdar, S.: Measuring and analyzing emerging properties for autonomic collaboration service adaptation. In: Rong, C., Jaatun, M.G., Sandnes, F.E., Yang, L.T., Ma, J. (eds.) ATC 2008. LNCS, vol. 5060, pp. 162–176. Springer, Heidelberg (2008)
13. Moser, O., Rosenberg, F., Dustdar, S.: Non-intrusive monitoring and service adaptation for ws-bpel. In: WWW 2008: Proceeding of the 17th international conference on World Wide Web, pp. 815–824. ACM, New York (2008)
14. Akyildiz, I., Su, W., Sankarasubramaniam, Y., Cayirci, E.: A survey on sensor networks. *IEEE Communications Magazine* 40(8), 102–114 (2002)
15. Sterritt, R., Smyth, B., Bradley, M.: Pact: personal autonomic computing tools. In: EASE Workshop at ECBS 2005, pp. 519–527 (2005)
16. Bigus, J.P., Schlosnagle, D.A., Pilgrim, J.R., Mills, W.N., Diao, Y.: Able: A toolkit for building multiagent autonomic systems. *IBM Systems Journal* 41(3) (2002)
17. IBM: Autonomic computing toolkit: Developer's guide (2004), <http://www-128.ibm.com/developerworks/autonomic/books/fpy@mst.htm>
18. Shannon, C.E.: A mathematical theory of communication. *Bell system technical journal* 27 (1948)
19. Dorn, C., Schall, D., Dustdar, S.: A model and algorithm for self-adaptation in service-oriented systems. In: IEEE European Conference on Web Services, ECOWS (November 2009)
20. Capocci, A., Servedio, V., Caldarelli, G., Colaiori, F.: Detecting communities in large networks. *Physica A: Statistical Mechanics and its Applications* 352(2-4), 669–676 (2005)
21. Newman, M.E.J.: Modularity and community structure in networks. *Proc. Natl. Acad. Sci. USA* 103, 8577 (2006)
22. Dorn, C., Schall, D., Dustdar, S.: Context-aware adaptive service mashups. In: IEEE Asia-Pacific Services Computing Conference (APSCC) - (short paper) (December 2009)
23. Michlmayr, A., Rosenberg, F., Platzer, C., Treiber, M., Dustdar, S.: Towards recovering the broken soa triangle - a software engineering perspective. In: 2nd International Workshop on Service-oriented Software Engineering (IW-SOSWE 2007 @ ESEC/FSE 2007) (September 2007)
24. Dujmovic, J.J.: Continuous preference logic for system evaluation. *IEEE Transactions on Fuzzy Systems* 15, 1082–1099 (2007)
25. Jennings, B., van der Meer, S., Balasubramaniam, S., Botvich, D., Foghlu, M., Donnelly, W., Strassner, J.: Towards autonomic management of communications networks. *IEEE Communications Magazine* 45(10), 112–121 (2007)
26. Colman, A.: Exogeneous management in autonomic service compositions. In: ICAS 2007: Proceedings of the Third International Conference on Autonomic and Autonomous Systems, Washington, DC, USA, p. 25. IEEE Computer Society, Los Alamitos (2007)
27. Yu, T., Lin, K.J.: Adaptive algorithms for finding replacement services in autonomic distributed business processes. In: Proceedings of Autonomous Decentralized Systems, ISADS 2005, pp. 427–434 (2005)
28. Wang, X., Vitvar, T., Kerrigan, M., Toma, I.: A qos-aware selection model for semantic web services. In: Dan, A., Lamersdorf, W. (eds.) ICSOC 2006. LNCS, vol. 4294, pp. 390–401. Springer, Heidelberg (2006)
29. Rosenberg, F., Leitner, P., Michlmayr, A., Celikovic, P., Dustdar, S.: Towards composition as a service - a quality of service driven approach, March 29–April 2, pp. 1733–1740 (2009)
30. Yang, Y., Mahon, F., Williams, M.H., Pfeifer, T.: Context-aware dynamic personalised service re-composition in a pervasive service environment. In: UIC, pp. 724–735 (2006)

31. Baresi, L., Bianchini, D., Antonellis, V.D., Fugini, M.G., Pernici, B., Plebani, P.: Context-aware composition of e-services. In: Benatallah, B., Shan, M.-C. (eds.) TES 2003. LNCS, vol. 2819, pp. 28–41. Springer, Heidelberg (2003)
32. Vu, L.H., Hauswirth, M., Aberer, K.: Qos-based service selection and ranking with trust and reputation management. In: Meersman, R., Tari, Z. (eds.) OTM 2005, Part I, LNCS, vol. 3760, pp. 466–483. Springer, Heidelberg (2005)
33. Maximilien, E.M., Singh, M.P.: Toward autonomic web services trust and selection. In: ICSOC 2004: Proceedings of the 2nd international conference on Service oriented computing, pp. 212–221. ACM, New York (2004)
34. Maximilien, E., Singh, M.: Self-adjusting trust and selection for web services, pp. 385–386 (June 2005)
35. Skopik, F., Schall, D., Dustdar, S.: The cycle of trust in mixed service-oriented systems. In: 35th Euromicro Conference on Software Engineering and Advanced Applications, SEAA (August 2009)

On the Semantics of the Extend Relationship in Use Case Models: Open-Closed Principle or Clairvoyance?

Miguel A. Laguna, José M. Marqués, and Yania Crespo

Department of Computer Science, University of Valladolid, Valladolid
{mlaguna, jmcc, yania}@infor.uva.es

Abstract. A use case is a description of the interactions of a system with the actors that use it. The Achilles' heel of use cases is the unclear UML semantics, in particular the definition of the extend relationship. This article is an attempt to clarify the semantics of the extension mechanism. In particular, we advocate for the application of the open-closed principle, adding modification details in the extending use case, instead of in the base case. A revision of the UML standard would be impractical, but a disciplined reinterpretation of the *extend* and *extension point* concepts could represent a great improvement. Textual and graphical approaches (based in the UML Behavior meta-model) are considered. Using these recommendations, the base use cases can be independently described, while the extending use cases will be self-contained.

Keywords: use case, extend relationship, extension point, UML meta-model.

1 Introduction

Ivar Jacobson proposed use cases and incorporated them into his OOSE development method [13], being recognized as a useful technique to elicit and record user requirements. A use case describes the possible interactions that can occur between an actor and the future system. It also describes the responsibilities of the system under design, without getting into implementation techniques or system internal details.

The inclusion of use cases in the first versions of UML as standard modeling language [20] is probably the main reason of its widespread acceptance. Nowadays use cases are one of the preferred techniques for the representation of user requirements (and the external behavior of an information system considered as a black box). They are essential in the Unified Process, as this development method was evolved from the ideas of Jacobson [12]. One of the major controversies is the UML's explanations of *include* and *extend* relationships. In particular, *extend* concept remains vague, and apparently contradictory. Precise and unambiguous definitions are missing in the numerous UML documents. The original motivation of the *extend* relationship was the aspiration of never touching the requirements of a previous version of a system. But the presence of the *extension point* concept in UML violates the original purpose. In UML you must anticipate in the base use case those places at which extensions are permitted (i.e, the extension points). Therefore, UML's explanations for *extend* relationship are still subject to debate. Some conferences have been devoted to these and other conflicting aspects [7].

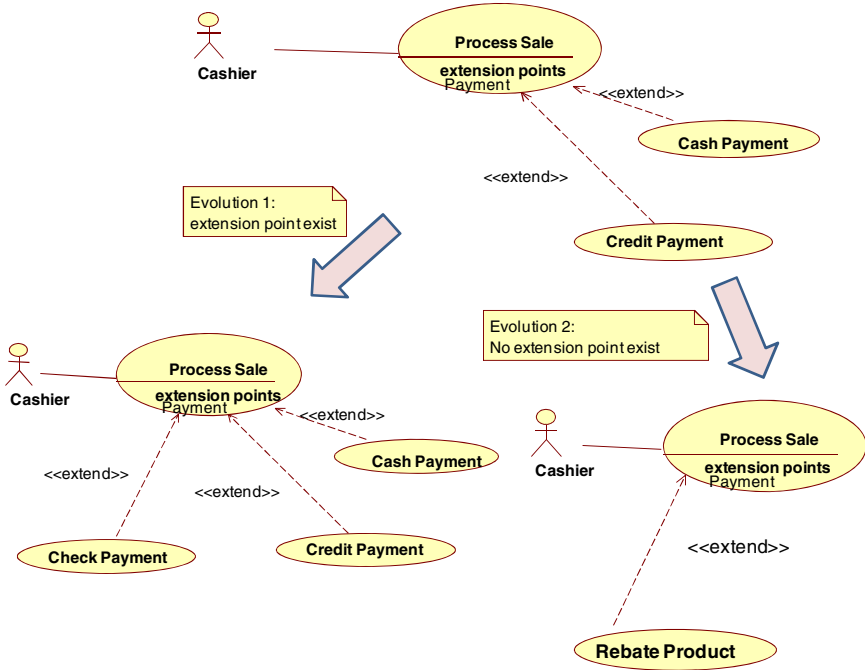


Fig. 1. UML extension (only possible if an extension point exists) and unanticipated extension (not possible in UML)

Figure 1, inspired in the example of *Point-Of-Sale* used by Larman [15], shows the differences between the original intention and the UML concept of *extension*. While the original extension could be considered as an enhancement of existing requirements (leaving untouched the original version), as depicted in evolution 2 of Figure 1, UML requires a previously defined extension point to incorporate a new extension. In the example of the base case of Figure 1, a new payment method (“*Check payment*”) is possible in UML but not the evolution 2 of Figure 1. In this situation, the system evolution requires a new use case that allows the application of discounts to certain products, as introduced by the cashier. If “*Process Sale*” does not have an adequate extension point, UML forbids the direct extension. The original intention is clearly more useful and we suggest recovering it in the UML context. As the alteration of UML meta-model is not an easy mission and we want to use conventional CASE tools, we propose to use some of the existing UML mechanisms to incorporate the original semantics of *extend*.

The rest of the paper is as follows: The next section briefly summarizes the UML vision of the *extend* relationships. Section 3 discusses the problems with the *extend* relationship, under the prism of the open-closed principle, and proposes a semantic reinterpretation of the *extension point* concept. Section 4 uses the UML Activity Package elements to visually represent an unanticipated extension. Section 5 presents related work and section 6 concludes the paper and proposes additional work.

2 The Extend Relationship in the UML Documentation

A use case describes an interaction between one or more actors and the system as a sequence of messages. Thus, a use case diagram has two types of nodes: actors and use cases, connected by association relationships. The original proposal of Jacobson also included two kinds of relationships between use cases: The *uses* and *extends* relationships, both indicated with generalization arrows. This syntax was initially preserved in primitive UML versions [20] but, beginning with the refined 1.3 version, a new set of relationships was proposed and this definition has essentially been kept, with minor changes, until the actual UML 2.1.2 version. From UML 1.3, relationships between use cases can be expressed in three different ways: generalization, *include*, and *extend* relationships. An *extend* relationship defines those instances of a use case that may be augmented with some additional behavior defined in an extending use case. While, the semantics of *include* relationship has always been reasonably clear, the *extend* relationship has generated a lot of controversy. Several modifications have been added to the different versions of UML. Attempts at removing these difficulties have been proposed in these documents. From here until the end of the article, we base the discussion on the official UML documentation, version 2.1.2 [19].

In the UML 2.1.2 meta-model, *Actor* and *UseCase* are both *BehavedClassifier*, which itself is a descendent of *Classifier*. As UML documentation states, the *extend* relationship specifies how and when the behavior defined in the extending use case can be inserted into the behavior defined in the extended use case (at one *extension point*). Two important aspects are: a) this relationship is intended to be used when some additional behavior can be added to the behavior defined in another use case; b) the extended use case *must be independent* of the extending use case.

Analyzing the meta-model (see the white meta-classes of Figure 3), the *extensionLocation* association end references the *extension points* of the extended use case where the fragments of the extending use case are to be inserted. An *extensionPoint* is an owned feature of a use case that identifies a point in the behavior of a use case where it can be extended by another use case. The *extend condition* is an optional *Constraint* that references the condition that must hold for the extension to take place. The notation for conditions has been changed in UML 2: the condition and the referenced *extension points* are included in a *Note* attached to the *extend* relationship.

Semantically, the concept of an “*extension location*” is left underspecified in UML because use cases “are specified in various idiosyncratic formats”. UML documentation refers to the typical textual use case description to explain the concept: “The use case text allows the original behavioral description to be extended by merging in supplementary behavioral fragment descriptions at the appropriate insertion points”. Thus, an extending use case consists of behavior fragments that are to be inserted into the appropriate spots of the extended use case. An extension location, therefore, is a specification of all the various (extension) points in a use case where supplementary behavioral increments can be merged.

The next sections are devoted to analyzing this relationship and the connected *extension point* concept, including its necessity. Then, in section 4, we make use of the UML *Behavior* meta-model package to visualize the notion of use case extension.

3 The Extend Relationship and the Open-Closed Principle

In this section, we try to answer a preliminary question: Is the presence of the *extension point* concept in the use case models really indispensable? From the point of view of the semantics of the dependence relationship, the mere presence of an *extension point* in the base use case is confusing. Removing (or perhaps reinterpreting) the *extension point* concept could be a way of avoiding many problems.

The first intention of a dependence relationship is to establish a directed relationship between an independent element (the base or extended use case) and a dependent element (the extending use case). Therefore, if the base use case must have no information a priori about its possible modification, the obligation of predetermining an *extension point* is contradictory.

Meyer, as long ago as 1988 coined the open-closed principle [16], that states:

- *"software entities (classes, modules, functions, etc.) should be open for extension, but closed for modification"*

This definition applied originally to the object-oriented programming languages. In this context, an entity can allow its behavior to be extended without altering its source code. *Open for extension* means that the behavior of the entity can be extended (we can make the module behave in different ways as the requirements change, or to meet the needs of new requirements). *Closed for modification* means that the source code of such an entity is not modifiable. The well known solution to this dilemma is the use of inheritance in object-oriented languages. This idea applies directly to the specialization relationship among classes in object-oriented designs and can also be adopted in requirements artifacts. In a wider vision, a broader definition can be stated. If we substitute the concept of entity by generic software artifacts, we can affirm:

- *"software artifacts (requirements, designs, implementations, test cases, etc.) should be open for extension, but closed for modification"*

In the use case context, *open for extension* means that the behavior of the use case can be extended (we can make a use case behave in different ways as the requirements change, or to meet the needs of new requirements). *Closed for modification* means that the details of this use case (the textual or graphical description of the behavior) are not modifiable. We are ready to map the open-closed principle from object-oriented programming languages to use cases. We will try to apply the same approach in two parallel situations:

- A base class is extended by redefinition of an operation in a specialized class (the simple addition of a new operation is a less problematic situation)
- A base use case is extended by an extending use case

Figure 2 presents two situations that share the idea of extension: In a typical Sales application, a previously existent (cash) payment is extended by a new type of payment. In structural models we will need probably one or several new classes. We will need also to redefine the **pay()** operation, adding details of the credit card, authorization codes, etc. In the use case diagrams we solve the situation with a *extend* relationship and a new use case that incorporates the new steps (including the introduction of the credit card number, the connection with and external authorization service, etc.).

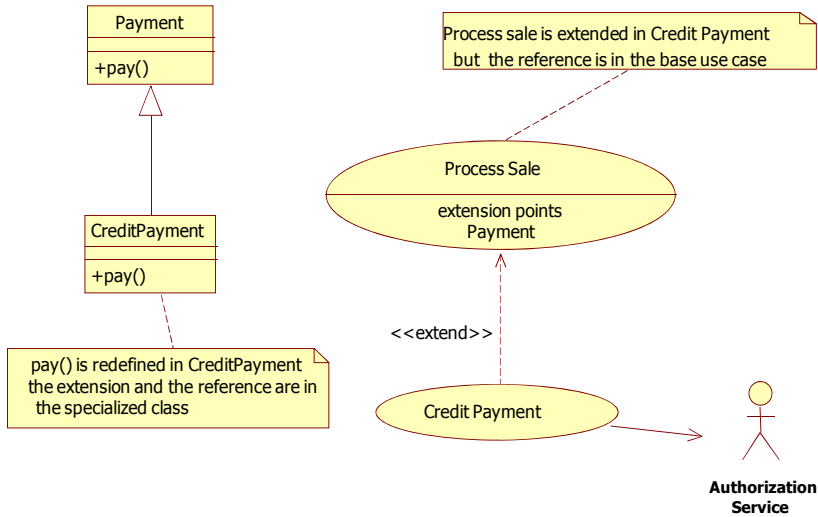


Fig. 2. The *extend* and *specialization* mechanisms in UML 2.1.2 [19]

If we consider Figure 2, we can appreciate that all the extension details of a class in a structural design diagram are defined in the extending class: The original *Payment* class is unaware of the redefinition of the *pay* method in the new class. The Eiffel version is absolutely clear (Java or C# versions are similar but not so explicit):

```

Class Payment
  feature
    pay(...) is do...end
  end

```

```

Class CreditPayment
  inherit Payment
  redefine pay end
  feature
    pay(...) is do...end
  end

```

Nothing in the original **Payment** class make reference to a hypothetical future modification and the class is closed. The two basic elements of the extension are the specialization relationship itself (and this is associated with the source element, i.e. the new class *inherit* the old) and the reference to the operation that need modification (and this is indicated by the *redefine* clause in Eiffel, or simply by duplicating the original name of the operation in UML diagrams).

The parallel version in the use case diagram is different: the details of the extension are in the new use case but, and here is the difference, an extension point must be previously defined in the original use case (in disagreement with the enunciated open-closed principle).

The open closed principle is feasible in object-oriented programming languages because they provide the adequate mechanism: the extension point (really a reference to the redefined method) and the extension itself (the new version) are both in the specialized class. The base class is unaware of the extension. Nevertheless, in use cases the extension is described in the new use case but the point of insertion is

defined in the base class. The situation is loosely similar to the old C++ implementation of inheritance using *virtual functions* (the “*clairvoyance principle*” as opposite to the open-closed principle).

In the context of use cases, the situations we want to solve are, for example: a use case can evolve during the development of several versions of a software system; the requirements can change; new constraints or business rules can appear, etc. The essence of these situations is that the evolution usually occurs “*in an unexpected way*”. While the user requirements are being elicited, we have a possible solution with plain use cases: add an alternative sequence of steps to the set of exceptions of the use case, referring to a step of the main scenario.

The generalization of the idea is exactly the extension concept, useful when a) the use case is already completely developed through a collaboration that involves analysis or design models, or b) the complexity of the steps that must be added recommends separating this piece of behavior in a new use case. In both cases, as in the plain solution, we must be able to indicate where the new sequence must be inserted (after the original step *n*) and where the original scenario must continue (at the original step *m*). This can be as complex as needed, as in the idea of *extension points* with several fragment insertions.

The concept of scenario, as sequence of steps, is not directly present in the UML meta-model Use Case Package, probably in order to allow different particular implementations of the *Behavior* meta-class (visual or textual, formal, structured or informal). However, independently of the concrete format, the concept of *sequence of steps* could be present in this Use Case Package.

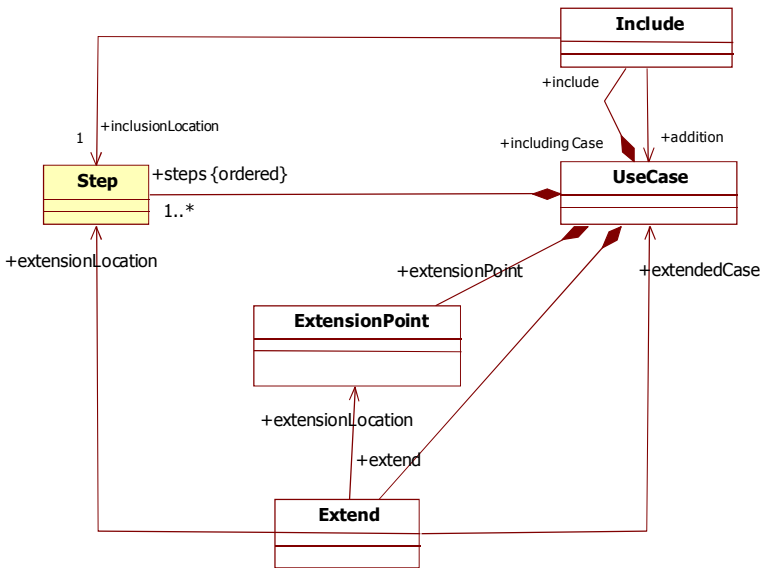


Fig. 3. The *Step* concept added to the UML Use Case Package meta-model

Consider provisionally the minimal variation of the meta-model shown in Figure 3: Only a new meta-class is added, representing the concept of *Step* and a *UseCase* owns an ordered set of steps. (A more complete version, considering all the possible alternative sequences, would include the intermediate *Scenario* concept as a set of ordered steps, owned by a *UseCase*.) This simple addition would allow defining the *extension* directly in function of the steps of the base use case, instead of a predefined extension point. As collateral effect (beneficial in this case), the exact position where the *Include* relationship is located can be made visible.

As we do not foresee immediate changes in the UML meta-model, we suggest an apparently incorrect solution to deal with this problem. Taking into account that the *Step* and *ExtensionPoint* definitions in Figure 3 are very similar from the viewpoint of the *Extend* meta-class, we consider that a use case has a set of steps (or sequences of inseparable steps) called *extension points*. If we think this way, all the steps of a use case are extensible. This interpretation implies that the use cases are completely open to future extensions (in the same way an unaffected class can be extended by a new one using inheritance in object oriented languages).

Really, this discussion about the meta-model is only conceptual. The details are in the textual step-based description of the use cases. In fact, this proposal assumes that the *extension point* concept is not used in the diagrams. In its place, we must indicate in the textual documentation of the extending use case:

- a) The base use case that is extended.
- b) The step where the extended use case is modified, using the same conventions of the alternative/exception fragments of the monolithic use cases; in other words, the precise step number must be referred.
- c) The “rejoin point” in the extended use case in order to continue with the normal sequence of steps. The last step of the extension must indicate it.

The adoption of this approach means that all the possible situations must be documented in the textual information of the extending use case. The extended use case remains unchanged and unaware of the extensions. All the exceptions related with the new extension must be treated (and solved) in the new use case.

Summarizing the idea, in many cases (in particular in agile developments), it is preferable not to use *extension points* with the original UML semantics. Or, changing the point of view, all the steps of a use case can be considered as *extension points*. This version smoothes the learning curve of the technique by beginners (in fact we use this approach with our undergraduate students, avoiding many confusing discussions in the requirements gathering sessions).

In this Section we have dealt with the textual specification of the use case details. The next Section examines the alternative (and less informal) graphical representation of the extension notion using the behavioral concepts of the UML meta-model.

4 The Extend Relationship and the Use Case Behavior Specification

As visible in the meta-model of UML, a *BehavioredClassifier* (and hence a *UseCase*) has an associated *Behavior* that can have a set of *activities*, *actions*, *messages*, *states*,

etc. The sequence of steps of the use case textual representation can be seen as a way of describing this behavior and many authors advocate for this perspective. This Section explores the possibilities that UML 2 have opened to specify the use case details in a graphical representation. The use of an *Interaction* as the representation of use case details is a familiar strategy. Larman [15] distinguishes the system sequence diagrams to specify the external behavior (requirements level) from the complete interaction that is the way the classes' new operations are discovered (the idea of collaboration as realization of the use cases).

The interaction diagrams have an evident problem: it is difficult to represent simultaneously all the alternative paths (in spite of the combined fragments possibilities). A better option is to use activities as the behavior specification of the use case. In the UML meta-model fragment of Figure 4 it can be realized that:

- a) *Activity* is a subtype of *Behavior*.
- b) *Activity* owns *ActivityNodes* (control or executable nodes), *ActivityEdges*, and *ActivityGroups*.
- c) *ActivityNode* or *Action* can be used as representation of a step of the use case scenarios. As UML states: "An action represents a single step within an activity, that is, one that is not further decomposed within the activity". Other specialized control nodes, such as *InitialNode* or *DecisionNode*, can be used to organize the alternative paths.
- d) *ActivityPartition* (shown as a *swimlane* or a stereotype) can represent the actors that interact with the system (and the system itself). The *swimlane* graphical representation allows associating easily each *Action* with an actor or the system.

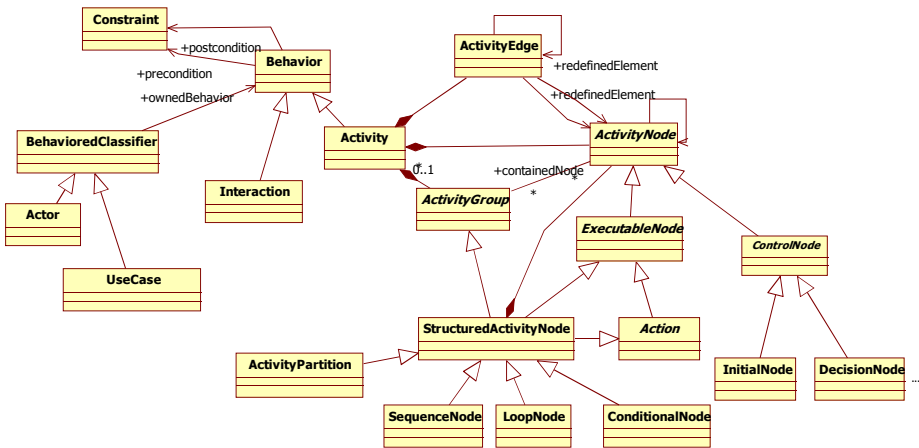


Fig. 4. Meta-model fragment of UML Activities and Use Case Packages

A fragment of the Process Sale use case (cash payment, following the example of Larman [15]) is shown in Figure 5. Only a successful scenario is represented and no extension points are provided.

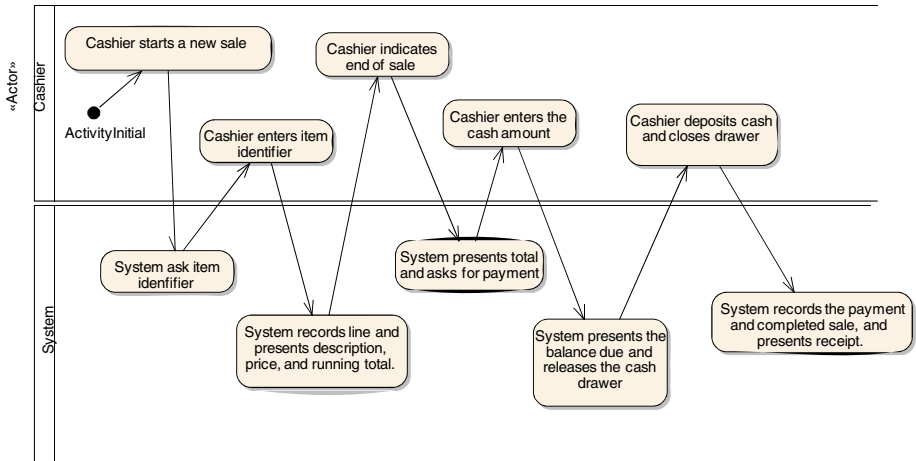


Fig. 5. A simplified fragment (successful scenario) of the Process Sale use case of Figure 1

To handle the extension points, all the potential of the Activity Packages of Figure 4 must be used. Note that an *ActivityNode* can be redefined. This opens the possibility of replacing a node with a *StructuredActivityNode*, indicating that the original node is substituted by a group of nodes. The semantics of this element refers to a basic sequence of nodes (*SequenceNode* specialization) or a complex construction (*ConditionalNode* and *LoopNode*). The key is that *StructuredActivityNode* is simultaneously a specialization of *ActivityNode*, *ActivityGroup* (pag. 309 of [19]) and *Action* (pag. 310 of [19]). At the same time, *StructuredActivityNode* contains a set of *ActivityNodes* (pag. 309 of [19]), where each node can be itself a *StructuredActivityNode*, configuring a typical composite pattern.

The strategy for representing the extension mechanism can be seen in Figure 6: A stereotyped node represents the extension point in the base activity and each extending use case requires an activity that redefines the extension point node (`<<extend>>` stereotyped dependency). This interpretation is consequent with the UML semantics: a pre-defined “hook” allows inserting the new behavior at the extension point.

As in the previous section discussion, this approach is only valid if we know *a priori* the details of the future extension. A set of alternatives are imaginable to conform to the open-closed principle. The original use case can be described with no mention to future extensions and the activity diagram will be similar to Figure 5. If a later evolution of the system requires an alternative payment form, two respectful variants of the previous solution are sketched in Figure 7. The first solution uses the generalization relationship between use cases, adding the extension point to the specialized use case. This variation does not change the original use case and divides the new information between the two new use cases: the intermediate *Advanced Process Sale* and the extending *Credit Payment*. However, the UML generalization semantics implies lost of control over the changes in the new version. The second variant uses the *package merge* mechanism of UML 2: The new version package contains a use case named as the original but with an added extension point. The package merge

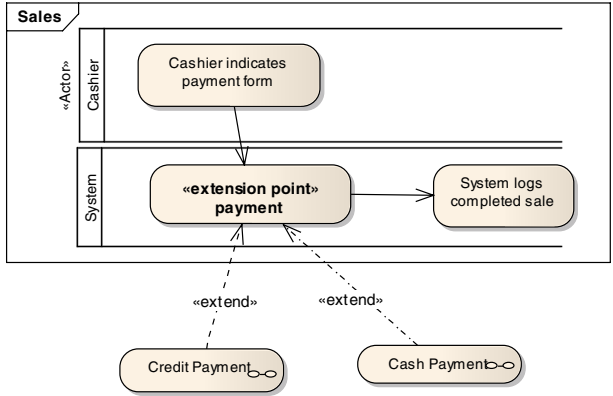


Fig. 6. A variant of the Process Sale use case with an extension point and two extensions

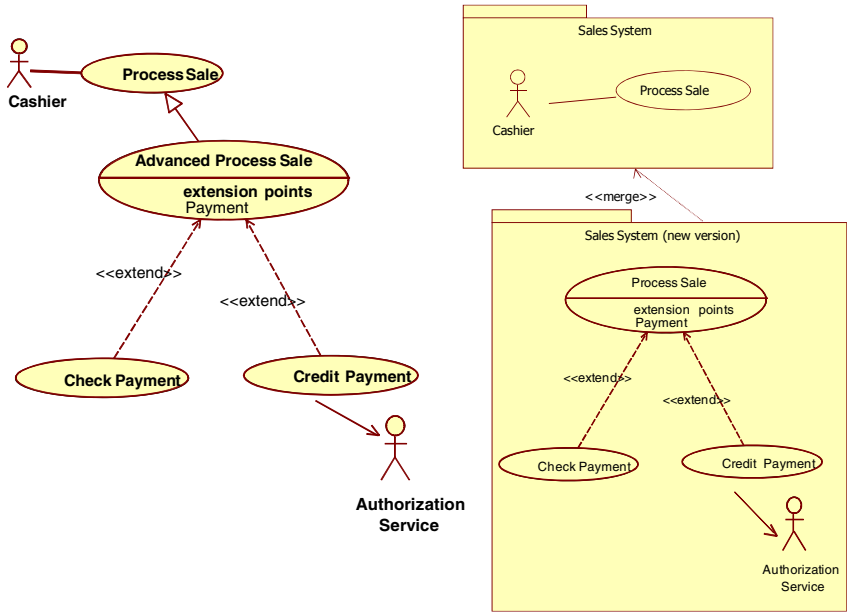


Fig. 7. The Use Case Process Sale and two reinterpretations of the Process Payment extension as an *a posteriori* addition

operation yields a full extensible use case. This is a valuable solution for the development of product families, as new features can be incorporated gradually.

Although these solutions are semantically correct and complies with the open-closed principle, the overload is manifest. A slightly modification of the scheme of Figure 6 can exploit the node redefinition mechanism of the UML meta-model. Figure 8 shows how an activity node (not an extension point) can be replaced by a

structured activity (<<redefine>> stereotyped dependency). The simplicity of the solution is appealing but it is only applicable to situations where all the sequence of elements can be neatly inserted in the position of the original node. Therefore, several problems remain unsolved: How to insert a new sequence when there is not a clear-cut node to be substituted? And, what if there are one or more “rejoin points” different of the extension point? Note that, if it is difficult to anticipate an extension point, it is yet more difficult to foresee the several possible “rejoin points” as the extension can include several success and fail scenarios

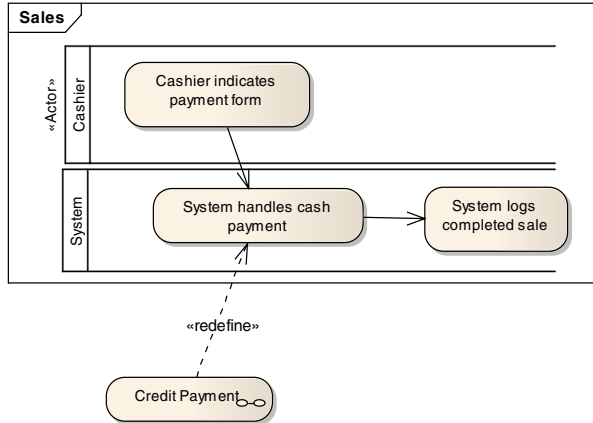


Fig. 8. Activity node replaced by a structured activity

The solution emerges if we recognize that an extension is really an alternative path (group of elements or *ActivityGroup*) that complements another existing path (another group). Therefore, the extend relationship is better formulated between the original unchanged activity and the extension activity. Inside the new activity, the connecting points must be referenced. The concrete graphical representation is an open issue, and varies with the concrete facilities of CASE tools. Taking into account these considerations, the use cases behavior specification could be depicted as shown in Figure 9, where an activity (“Process Sale”) is extended by another activity (“Rebate Product”), in parallel with the use case diagram. The inner region delimits the extension and the nodes outside this region are references to the original activity nodes that connect the old and new paths. The use of the associated Note indicates the extend condition (as in the standard) and the list of parameter values that serve to reference the connection points. This parameterization allows the reuse of the extending use cases with different base cases. The main advantage is the exclusion of the extension details from the original use case. In this sense, the final conclusion is that the UML meta-model provides the extensibility elements we need to apply the open-close principle in use case evolution.

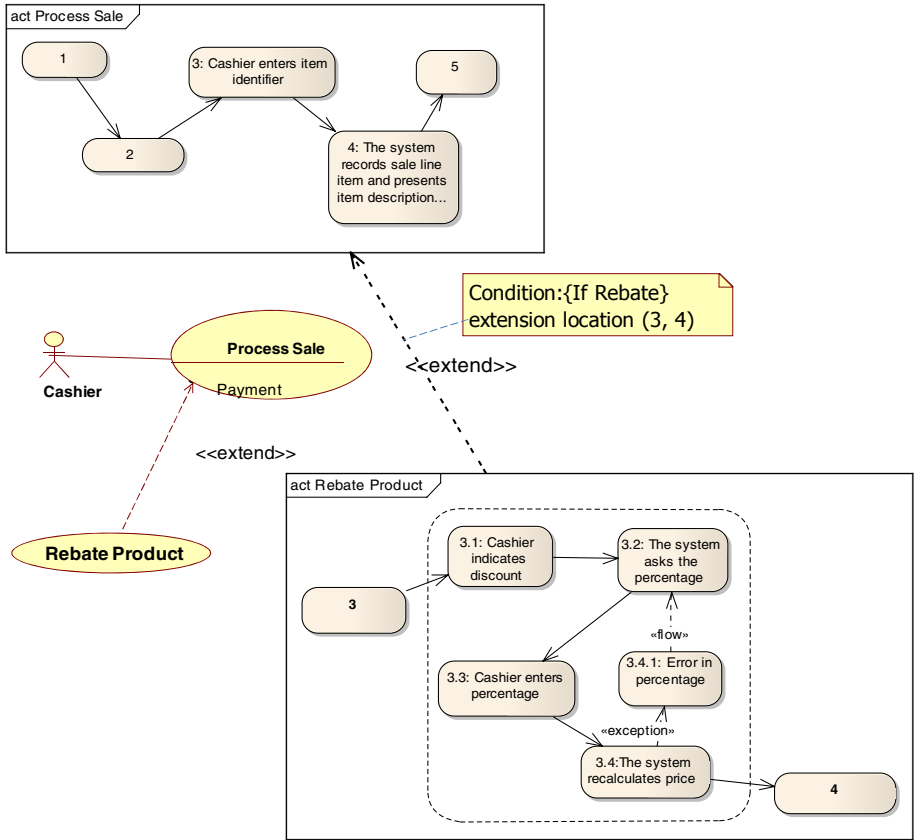


Fig. 9. Not anticipated use case extension with inclusion of the connection points

5 Related Work

Many suggestions for modification of the UML meta-model have been proposed, including the use of ontologies instead [9]. Some authors, such as Berard [1] or Simons [24] have been detractors of use cases, though Simons (in a work with van der Berg [27]) proposed control flow semantics for use case scenarios. Conversely, there are many works that try to improve or at least clarify them, such as the classical book of Cockburn [4] or the work of Williams *et al.* [29]. Metz *et al.* distinguish several textual variants for clarifying the semantics of extension points and rejoin points [17].

Many authors, such as Cockburn [4], have suggested that the most important characteristics of use cases are the textual details to be discussed with the end users while neglecting the visual representation and semantics proposed by UML. Others, such as Rumbaugh and Jacobson, insist on the graphics aspects [23].

Some additional relationships between use cases have been proposed. Rosenberg [21] uses precedes and invokes constructs to factor out common behavior. Conversely, other authors such as Larman [15] advocate not using the extend relationship or using only when it is undesirable to modify the base use case.

The *BehavioredClassifier* specialization of the use cases has been analyzed in [7]: The *Behavior* meta-class is a specification of how its context classifier (use case) changes over time and the *BehavioredClassifier* is a classifier that can have behavior specifications. In other words, a *BehavioredClassifier* is rather an ordinary classifier that can own behaviors [7]. The conclusion is that the formalization of use cases as classifiers in UML has obscure points: Two contradictory notions of use cases coexist in UML 2: “set of interactions” vs. “set of entities”. The authors propose the meta-model should be changed to make *UseCase* a subtype of *Behavior*, not of *BehavioredClassifier*. Alternatively, they admit that the meta-model may be kept as it is, but it should be recognized that a use case is the specification of a role. Williams *et al.* also analyze the UML 2 meta-model and propose changing *UseCase* to a subclass of *Behavior* [29]. Isoda states that UML 2 has a correction about the relationship between use cases and actors, which effectively means that UML has finally abandoned the idea of “actors call operations of a use case”, but the details of UML 2 in fact still retain those defects [10]. Jacobson believes that integrating use cases and aspect oriented programming (AOP) will improve the way software is developed. The idea is to slice the system and keep the use cases separate all the way down to the code. “In the long term we will get more of extension-based software-extensions from requirements all the way down to code and runtime; and extensions in all software layers, for example, application, middleware, *systemware*, and extensions across all these layers” [11]. Other approaches different from UML have considered the behavior extension. For example, the Taxis semantic model defines transactions as class hierarchies [18].

Sousa *et al.* adapt use-cases in order to explicitly provide the separation of cross-cutting concerns in requirements artifacts. They propose *crosscutting* use-cases to separate crosscutting behavior (for example, security) from the main scenario [26]. In the same line, Somé and Anthonysamy [25] present an approach for modeling use cases with aspect-oriented techniques. Cross-cutting requirements are modeled in addition to functional requirements using a new relationship stereotyped as <<*aspect*>>. The composition is based in Petri nets properties. In both cases, new elements are proposed to complete the UML meta-model or to represent the details of use case composition.

Araujo *et al.* compose aspectual and non-aspectual scenarios instead use cases. Non-aspectual scenarios are modeled as UML sequence diagrams. Aspectual scenarios are modeled as Interaction or State Machine Pattern Specifications [6]. Then, aspectual and non-aspectual scenarios are composed at sequence diagram and state machine levels [1]. Whittle proposes use case charts with the aim of formally defining the semantics of scenarios [28]. These proposals are close to our vision of composing fragments of models but introduce additional complexity and separates from UML standard.

As a general observation about these advances, in spite of the utility of considering early aspects (for example to model repetitive requirements as security or audit steps), we think that extension problems should not be treated inevitably as aspects.

Braganza *et al.*, discuss the semantics of use case relationships and their formalization using activity diagrams in the context of variability specification. They propose an extension to the *extend* relationship that supports the adoption of UML 2 use case diagrams into model driven methods. The proposal results from the 4 Step Rule Set, a model driven method in which use cases are the central model for requirements specification and model transformation [3].

The common conclusion of most of the work done in use case semantics is that the question is not well solved in UML and a redefinition of the concepts is needed. We believe that our contribution can help in this redefinition.

6 Conclusions and Future Work

In this article, the problems of interpretation of the *extend* semantics in use case models are analyzed. An improvement of the *extension* notion is proposed, based in the open-closed principle, by adding the *Step* concept or alternatively using as substitute the *extension point* concept itself.

The second part of the article presents a possible graphical approach, using the Behavioral aspects of the UML meta-model. We have shown that UML provides the extensibility elements we need to apply the open-close principle in use case extension. We think that, without neglecting major future modifications in the UML meta-model, these proposals can help in the process of elicitation and specification of functional requirements, clarifying the intention of the final users.

In conclusion, we believe that the set of semantic reinterpretations proposed in this article can help to solve many of the practical extension problems the requirements engineers face in their daily work. The empirical validation is a work in progress to check the usefulness of the approach. We are defining a set of problems so that diverse groups (from undergraduate students to experts) can use these techniques to compare (via controlled experiments) the comprehensibility and feasibility of the diverse variants of the extension concept.

References

1. Araujo, J., Whittle, J., Kim, D.: Modeling and Composing Scenario-Based Requirements with Aspects. In: Proceedings of the 12th IEEE international Requirements Engineering Conference (2004)
2. Berard, E.V.: Be Careful with Use Cases. Technical report. The Object Agency, Inc. (1995)
3. Braganca, A., Machado, R.J.: Extending UML 2.0 Metamodel for Complementary Usages of the «*extends*» Relationship within Use Case Variability Specification. In: Proceedings of the 10th international on Software Product Line Conference, pp. 123–130. IEEE Computer Society, Washington (2006)
4. Cockburn, A.: Writing Effective Use Cases. Addison-Wesley Professional, Reading (2000)
5. Constantine, L., Lockwood, L.: Software for Use. Addison-Wesley, Reading (1999)
6. France, R., Kim, D., Ghosh, S., Song, E.: A UML Based Pattern Specification Technique. IEEE Transactions on Software Engineering 30(3), 193–206 (2004)
7. Génova, G., Llorens, J., Metz, P., Prieto-Díaz, R., Astudillo, H.: Open Issues in Industrial Use Case Modeling. In: Jardim Nunes, N., Selic, B., Rodrigues da Silva, A., Toval Alvarez, A. (eds.) UML Satellite Activities 2004. LNCS, vol. 3297, pp. 52–61. Springer, Heidelberg (2005)

8. Génova, G., Llorens, J.: The Emperor's New Use Case. *Journal of Object Technology* 4(6), 81–94 (2005); Special Issue: Use Case Modeling at UML 2004
9. Genilloud, G., William, F.: Use Case Concepts from an RM-ODP Perspective. *Journal of Object Technology* 4(6), 95–107 (2005); Special Issue: Use Case Modeling at UML 2004
10. Isoda, S.: A Critique of UML's Definition of the Use-Case Class. In: Stevens, P., Whittle, J., Booch, G. (eds.) *UML 2003*. LNCS, vol. 2863, pp. 280–294. Springer, Heidelberg (2003)
11. Jacobson, I.: Use Cases and Aspects—Working Seamlessly Together. *Journal of Object Technology* (2003), <http://www.jot.fm>
12. Jacobson, I., Booch, G., Rumbaugh, J.: *The Unified Software Development Process*. Addison-Wesley, Reading (1999)
13. Jacobson, I., Christerson, M., Jonsson, P., Overgaard, G.: *Object-Oriented Software Engineering, A Use Case Driven Approach*. Addison-Wesley, Reading (1994)
14. Jacobson, I., Griss, M., Jonsson, P.: *Software Reuse*. In: *Architecture, Process and Organization for Business Success*, ACM Press/Addison Wesley Longman (1997)
15. Larman, C.: *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and the Unified Process*, 3rd edn. Addison Wesley, Reading (2004)
16. Meyer, B.: *Object Oriented Software Construction*. Prentice-Hall, Englewood Cliffs (1988)
17. Metz, P., O'Brien, J., Weber, W.: Specifying Use Case Interaction: Clarifying Extension Points and Rejoin Points. *Journal of Object Technology* 3, 87–102 (2004)
18. Mylopoulos, J., Bernstein, P.A., Wong, H.K.: A language facility for designing database-intensive applications. *ACM Trans. Database Syst.* 5(2), 185–207 (1980)
19. OMG, Unified Modeling Language: Superstructure, version 2.1.2. formal doc. 2007-11-01. 2007 (2007)
20. Rational Software Corporation Unified Modelling Language Version 1.1 (1997)
21. Rosenberg, D., Scott, K.: *Applying Use Case Driven Object Modeling with UML: A Practical Approach*. Addison Wesley, Reading (1999)
22. Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., Lorensen, W.: *Object-Oriented Modeling and Design*. Prentice Hall, Englewood Cliffs (1991)
23. Rumbaugh, J., Jacobson, I., Booch, G.: *The Unified Modeling Language Reference Manual*, 2nd edn. Addison-Wesley Professional, Reading (2004)
24. Simons, A.J.H.: Use Cases Considered Harmful. In: 29th Conf. Tech. Obj-Oriented Prog. Lang. and Sys. (TOOLS-29 Europe), pp. 194–203. IEEE Computer Society, Los Alamitos (1999)
25. Somé, S.S., Anthonysamy, P.: An approach for aspect-oriented use case modeling. In: *Proceedings of the 13th international Workshop on Software Architectures and Mobility*, pp. 27–34 (2008)
26. Sousa, G., Soares, S., Borba, P., Castro, J.: Separation of crosscutting concerns from requirements to design: Adapting the use case driven approach. In: *Proceedings of the Early Aspect Workshop at AOSD*, pp. 93–102 (2004)
27. van den Berg, K.G., Simons, A.J.H.: Control flow semantics of use cases in UML. *Information and Software Technology* 41(10), 651–659 (1999)
28. Whittle, J.: Precise specification of use case scenarios. In: Dwyer, M.B., Lopes, A. (eds.) *FASE 2007*. LNCS, vol. 4422, pp. 170–184. Springer, Heidelberg (2007)
29. Williams, C., Kaplan, M., Klinger, T., Paradkar, A.: Toward Engineered, Useful Use Cases. *Journal of Object Technology* 4(6), 45–57 (2005); Special Issue: Use Case Modeling at UML 2004

Situational Evaluation of Method Fragments: An Evidence-Based Goal-Oriented Approach

Hesam Chiniforooshan Esfahani¹, Eric Yu², and Jordi Cabot³

¹ Department of Computer Science, University of Toronto

² Faculty of Information, University of Toronto

³ INRIA - École des Mines de Nantes

hesam@cs.toronto.edu, yu@ischool.utoronto.ca,

jordi.cabot@inria.fr

Abstract. Despite advances in situational method engineering, many software organizations continue to adopt an ad-hoc mix of method fragments from well-known development methods such as Scrum or XP, based on their perceived suitability to project or organizational needs. With the increasing availability of empirical evidence on the success or failure of various software development methods and practices under different situational conditions, it now becomes feasible to make this evidence base systematically accessible to practitioners so that they can make informed decisions when creating situational methods for their organizations. This paper proposes a framework for evaluating the suitability of candidate method fragments prior to their adoption in software projects. The framework makes use of collected knowledge about how each method fragment can contribute to various project objectives, and what requisite conditions must be met for the fragment to be applicable. Pre-constructed goal models for the selected fragments are retrieved from a repository, merged, customized with situational factors, and then evaluated using a qualitative evaluation procedure adapted from goal-oriented requirements engineering.

Keywords: Software Development Methodology, Situational Method Engineering, Goal-Oriented Modeling, Method Evaluation.

1 Introduction

One of the common concerns of project managers is to ensure that their development processes fit well with the particular needs of their projects. Despite the proposal of elaborate frameworks for building situational methods [1-4], many software companies still follow ad-hoc methods of software development, which are built intuitively by adopting some fragments from different methodologies and tailoring them to their development method [5, 6]. With this approach, the best case scenario is that the company will benefit from all advocated advantages of selected method fragments. Unfortunately, this is not always the case, and there are numerous reports of project failure that were due to the improper choice of development method [7].

For instance, suppose that a software organization wants to incorporate “Pair Programming” [8] as part of its development method. It is known that pair programming usually helps to achieve some objectives such as “reduced defect rate” and “real-time

knowledge transfer”. But, are these objectives fully achievable in all project situations? Could other method fragments, when combined with pair programming, facilitate (or hamper) these objectives? There are increasing empirical studies that have addressed such questions, and investigated the success or the failure of different method fragments in different project situations. For instance, there exist over 100 empirical studies about “Pair Programming” in various situations¹. Considering the large number of existing method fragments and their volume of supporting studies, the need arises for a solution that can systematically make this evidence-based information available to practitioners, and help them during the process of method construction.

This paper proposes a framework for evaluating software development processes. The proposed framework considers a development method as a set of method fragments [2], and helps project managers anticipate whether the expected benefits in adopting the selected fragments will be attainable given the particular circumstances faced by the project. Starting from a set of candidates fragments, the framework leads the user to consider their objectives and requisites. Then, method objectives will be evaluated by reflecting the impacts of situational factors on method requisites, and propagating the achievement status of method requisites to method objectives.

In order to take advantage of existing situational knowledge of different method fragment, the proposed framework provides *evidence-based* repositories, which contain a structured representation of results from empirical studies. The contents of these repositories are gathered through systematic literature review [9]. The evidence base of this framework visualizes the collected knowledge of method fragments in a *goal-oriented* representation. The framework reuses its ready-made goal models to facilitate the process of method evaluation. The proposed framework can be deployed as a complement to existing method engineering frameworks, and due to its simplicity, it can also be used by project managers who are not willing to become involved in the detailed steps of situational method construction. In the rest of this paper, we will explain the framework using an illustrative scenario, introduced in the next section.

2 Motivating Scenario

As an illustrative scenario, consider a project manager facing the following circumstances in her project: (1) Technology used in the project is new to the company; (2) Developers are mainly junior programmers; (3) Team members are distributed, with a project wiki and email-lists being the main communication channels; (4) the project manager serves as the customer representative; and (5) Developers do not have access to the target device on which the product software will be installed, and can only test the software on a limited emulator.

The project manager is attracted to agile methods as they are said to lead to faster time to market, improved communication, more reliable project planning, and more effective teamwork (e.g. efficient self-organizing). Based on what she knows about agile methods and the characteristics of her organization, the manager is planning to adopt the following method fragments from XP [8] and Scrum [10]: (1) Daily Scrum Meeting: Short daily meetings to explain performed activities and to discuss obstacles; (2) Short iterations: Delivering an increment of software every three weeks;

¹ The search was run on libraries of IEEE Explorer, Springer, and Elsevier in November 2009.

(3) Iteration Planning: Selecting high priority requirements at the beginning of each iteration, and breaking them down to smaller tasks; (4) Pair Programming: Assigning two programmers to work together on each software component. However, she is unsettled by many questions, such as:

- Will the selected method fragments work well together or will they conflict? (e.g. should pair programming be used together with iteration planning?)
- Does my team and project environment satisfy the necessary requisites to take advantage of the selected method fragments?
- Will the combined set of fragments produce the desired results and meet project objectives?

3 Evaluation Framework

This section introduces a framework for evaluating a set of candidate method fragments to be a part of organization development method. The framework helps project managers to anticipate the achievement of method objectives, and find answer for questions, such as those mentioned in previous section. Fig. 1 shows the overall structure of the framework. The existing knowledge about method fragments will be kept in an *Evidence Base*, which is composed of two repositories: (1) *Method Fragment Repository* that holds objectives and requisites of method fragments, along with situational evidences; and (2) *Model Fragment Repository* that holds the graphical representations of method fragments.

To evaluate a number of candidate method fragments, their corresponding model fragments will be retrieved from the Model Fragment Repository, and will be merged based on their objectives. Then the integrated models will be customized and initialized with respect to the project situational attributes. During this step, the situational evidences that have been stored in Method Fragment Repository will be reused. At the end, the integrated models will be evaluated, to anticipate the satisfaction degree of method fragments’ objectives. The framework makes its evidence base available to software organizations, and process designers are responsible for performing the evaluation steps.

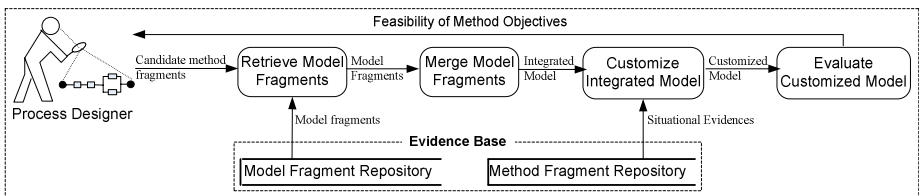


Fig. 1. Evaluation Framework

3.1 Evidence Base

The evidence base of this framework is inspired by the assembly-based SME frameworks [2, 11]. But, unlike current method repositories (e.g. OPF [12]) that store the specification of method fragments, the evidence base of this framework contains the

experience results of empirical studies that have been conducted for different method fragments. This framework follows the definition of *Method Fragment* as explained in [2], which classifies method fragments into *process* and *product* fragments, respectively referring to the elements of development process and structure of products. Another category of method fragments has been introduced in [12], called *producer*, which represent methodology roles played by persons or tools. As mentioned before, the evidence base is composed of two repositories that will be explained in the next two subsections.

3.1.1 Method Fragment Repository

The Method Fragment Repository stores the textual information of method fragments. The information has been collected through systematic literature review of published empirical studies, which have tested the enactment of method fragments in different situations. For each method fragment, this repository holds two datasets: *Objectives Dataset*, and *Requisites Dataset*.

3.1.1.1 Objectives Dataset. For each method fragment, the repository provides an *objectives dataset*, which represents the quality goals that are expected to be achieved by the enactment of the method fragment. These objectives have been extracted from published empirical studies on method fragments (i.e., the dataset does not include quality goals that were just claimed for a method fragment without any supporting empirical evidence). This dataset stores method objectives in two categories: *major* and *minor*. A major objective is defined as a quality goal that can be decomposed into a number of sub-goals, called minor objectives. Perhaps the classification of quality goals could be performed more elaborately; however, for the sake of simplicity this framework considers only these two levels. The objectives dataset also provides situational evidences for the contribution of method fragment to its objectives. Besides, for every contribution relation, the dataset provides the reference

Table 1. A subset of major and minor objectives that “Daily Scrum Meeting” contributes to them, with reference to the investigating empirical studies, and particular situational evidences

| Major Objective | Minor Objective | Contribution Type from Fragment | Study | Situation |
|--|--|---------------------------------|-----------|--|
| Effective Communication | | ++ | [S1] | Default |
| | Improved awareness (of what others are doing) | ++ | [S1] | Default |
| | | - | [S1] | Large projects, as they may need extensive number of meetings |
| | Real-time knowledge transfer | + | [S8] | Default |
| | | - | [S2, S12] | Distributed Development: use of email and wiki pages for comm. |
| | Enhanced Communication with business people / project leader | ++ | [S3, S8] | Existence of multi-level Scrum in case of many scrum teams |
| Better understanding of customer needs | + | [S8] | Default | |

to the study that provided the empirical evidence, and possibly the description of the situation of study. Table 1 shows a portion of the objective dataset for method fragment “Daily Scrum Meeting”.

Typically the contribution of a method fragment to its objectives is positive. However, there might be some situations where a method fragment adversely impacts its objectives. For instance, although “Daily Scrum Meetings” usually makes strong positive contribution to the “Improved Awareness” of a team about the activities of other team members, as studied in [S1], in the case of large projects with multiple development teams, daily meetings can cause confusion by bringing up excessive details, which are not relevant for a large portion of developers. The framework proposes four possible types of contribution relations: *strongly positive* (++), *positive* (+), *negative* (-), and *strongly negative* (--).

3.1.1.2 Requisites Dataset. The other dataset kept for each method fragment is the *requisites dataset*. This dataset contains the conditions that should be met for the successful enactment of a method fragment (e.g., resources to be provided, tasks to be performed, or personnel qualifications to be met). The framework defines the relation of a method fragment to its requisites as *decomposition relation*, since the successful enactment of a method fragment is due to the successful achievement of its requisites. Similar to the objectives dataset, this dataset represents method requisites in two levels of abstraction (major and minor), also sets the contribution relation of minor requisites to major ones. Besides the dataset provides situational evidences for each requisite, by referencing to the studies in which the requisite was satisfied or denied (partially or fully). In most cases it explains the significant situational factors of the referenced empirical studies that affected the fulfillment of method requisites. Fig. 2 (a) shows the metamodel of the method fragment repository.

Table 2. A subset of requisites of “Pair Programming“, Contribution of Minor to Major Requisite; Situational Fulfillment Status [Satisfied(✓), Denied(×), or Partially Denied(✕)]

| Major Req. | Minor Requisite | Contrib. to Major Req. | Situa. Fulfill. Status | Study | Situation |
|-------------------------|-----------------------------------|------------------------|------------------------|-------|---|
| Effective Collaboration | Equal engagement in coding | + | ✓ | [S15] | Pairing programmers with equal expertise |
| | | + | ✕ | [S15] | Pairing programmers with different expertise (weaker programmer became passive) |
| | Joint Decision Making | + | ✕ | [S15] | Similar pairs; the one who had the control of machine usually had a significant advantage w.r.t decision making |
| | Collaboration be viable | + | ✓ | [S17] | Pairs with heterogeneous personality profile |
| | | | ✕ | [S17] | Pairs with homogenous personality profile |
| | Similar working and resting hours | + | × | [S30] | Pairs with different times for starting their job or resting |

Table 2 shows a subset of the requisites of “Pair Programming”, focused on “Effective Collaboration”. For instance, it shows that “Equal engagement (of pairs) in Coding” is a minor requisite that contributes positively to the major requisite “Effective Collaboration”. However, not every situation can satisfy this requisite. For example, the empirical study [S15] has shown that pairing programmers with different levels of expertise can result in passiveness of the weaker programmer, thus partial denial of the requisite. The objectives dataset also takes a goal-oriented approach in representing method requisites in order to facilitate their modeling and evaluation in later stages of the framework.

3.1.2 Model Fragment Repository

The Model Fragment Repository of this framework contains the pre-constructed graphical representations of method fragments. For each method fragment, this repository contains a number of models, called *model fragments*, each visualizing a subset of its objectives and requisites (which were textually stored in the method fragment repository). In fact, each model fragment is a piece of a comprehensive model that could represent the whole knowledge of a method fragment. The main reason for breaking down the visualization of each method fragment was to avoid the potential complexities that could have emerged if the whole knowledge of a method fragment was represented in a single model.

The suitable modeling language for representing model fragments should have enough constructs to model method objectives and requisites, and also contribution and decomposition relations. We found the i^* modeling language [13] as an appropriate choice, especially by considering its prior use for modeling the intentional aspects of software processes [14]. For developing model fragments, the following set of i^* constructs are used: *Task*, representing method fragments; *Softgoal*, representing method objectives and requisites; *Contribution Link*, representing contribution relations; and *Decomposition Link*, representing decomposition relations. We defer the representation of actors to future work.

Fig. 2 (b) shows an example model fragment, developed for the method fragment: “Iteration Planning” and the major objective: “Effective Project Planning”. This model fragment shows that conducting iteration planning sessions would help to enhance the Effective project planning by positively contributing to the (1) realistic prioritization of tasks; (2) clear definition of iteration goals; and (3) accurate project scheduling. The meaning of contribution links used in model fragments corresponds to their counterparts in method fragment repository: Some+ and Help contribution links represent strong (++) and normal (+) positive contributions; Some- and Hurt contribution links represent strong (--) and normal (-) negative contributions.

Since a model fragment usually represents only one of the major objectives of a method fragment, it contains a limited set of fragments’ requisites – those that are somehow related to the major objective. For instance, in Fig. 2 (b) if the objective of model fragment was “Effective Communication” (rather than “Effective Project Planning”), then requisites that were related to the capability of team members in task estimation and task breakdown, should have been discarded.

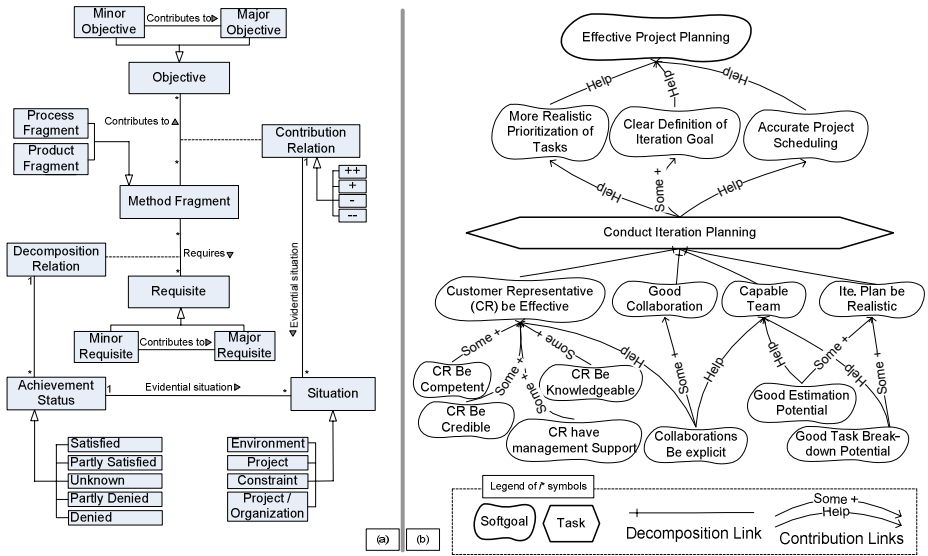


Fig. 2. (a) Metamodel of method fragment repository; (b) An instance of model fragment repository, representing the method fragment: “Iteration Planning” and the objective: “Effective Project Planning”

3.2 Situational Evaluation

So far we have described the evidence base of the framework. In the following subsections we describe the steps to be taken to evaluate a set of method fragments being considered for a given organization/project.

3.2.1 Model Merging

This framework evaluates a set of candidate method fragments by anticipating the extent to which their objectives will be satisfied. Some objectives of a method fragment may be in common with those of other method fragments. Thus, for correct anticipation of method objectives we have to consider the contribution of all candidate method fragments to those objectives. Model merging is intended to develop integrated views of model fragments, which highlight their contribution to the common set of objectives. In order to develop integrated models, process designers should first specify a set of major objectives that they want to be evaluated. Then, retrieve the relevant model fragments from the Model Fragment Repository, and for every intended major objective, merge the related model fragments by following these steps:

- 1) Identify its minor objectives that are contributed to by candidate method fragments
- 2) Set the contribution relations from minor objectives to the major ones
- 3) Set contributions of method fragments to the minor objectives

While merging method fragments based on their objectives, we should also consider the potential contributions that may exist between the requisites of method fragments. Such contribution relations (if detected) should be represented in integrated models.

The contribution relation of minor objectives to major ones (with respect to the method fragment) is proposed in the model fragment repository.

In our motivating scenario, project manager was interested to anticipate the satisfaction (or denial) degree of three major objectives: “Effective Communication”, “Effective Project Planning”, and “Efficient Self-Organizing”. Fig. 3 shows the result of merging three of our model fragments based on the common objective: “Effective Communication”.

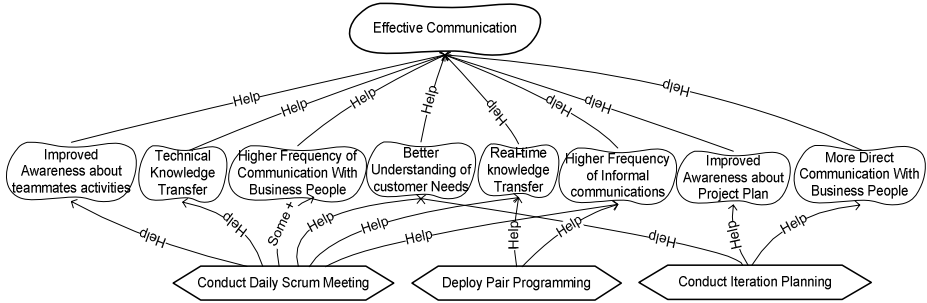


Fig. 3. Merging model fragments based on the common objective: “Effective Communication” (Requisites are not shown in this model)

Integrated models serve as intermediate artifacts in this framework. They can help process designers identify (1) method objectives that receive contradictory contributions via different method fragments; (2) method objectives that are strongly contributed, typically by different method fragments; and (3) method objectives that are weakly contributed, typically by only one method fragment. For instance, Fig. 3 shows that “Better Understanding of Customer Needs” is a method objective that receives contributions from two fragments “Daily Meeting” and “Iteration Planning”, thus we would expect its successful achievement according to the generic knowledge from the evidence base. However, will the specific situation of this project affect this assessment? To answer this question we need to first customize the integrated model according to the project situational factors and then complete its evaluation.

3.2.2 Model Customization

This phase is intended to customize the integrated goal models, in order to reflect the impacts of situational factors on them. The customization process takes place in two complementary stages: *Model Refinement*, which modifies the structure of the goal model; and *model initialization*, which assigns initial values to the requisites of method fragments. This framework proposes two strategies for model refinement:

- *Changing the type of contribution links* – regarding the specific situation of a software project, the types of contributions that a method fragment makes to its objectives may differ from their default contribution types (stored in the model fragments repository). In such cases, the contribution type should be altered to represent the specific situational attributes of software project. For instance, the default contribution of method fragment “Conduct Daily Meetings” (as defined in Scrum [10]) to its objective “Improved Awareness (of what others are doing)” is

“Help”. However, in an organization where the size or the number of development teams is too large, this contribution should be altered to “Hurt” (explained in the reference [S1] of the Appendix).

- o *Adding new softgoals to the requisites* – situational factors that impact the fulfillment of method requisites will be represented as new softgoals, contributing to the existing method requisites. Different categories of situational factors have been proposed in [2, 15, 16]. These can be represented as softgoals. For instance, “Established Standards” is a situational factor [2] that can be represented as softgoal “Standard X be followed”.

Different project situations imply different initial values for the achievement status of method requisites. This framework identifies two types of initialization for method requisites:

- o *Evidential* – initializations that are supported by solid evidence(s). For instance, in distributed projects where development teams have substantial time zone differences, the achievement status of method requisite “Meetings be face-to-face” will be initialized with “Denied” value.
- o *Assumptive* – Initializations that are based on assumptions with no significant evidence. For example, in situation that most of team members have just met each other, the achievement status of method requisite “Collaboration be Viable” cannot be predicted for sure, yet it can be assumed to become at least “Partially Satisfied”.

The empirical evidences, which have been populated in the repository of method fragments, can be quite helpful during model refinement and initialization, provided that a similar situation has been expressed.

Table 3. Significant situational factors, their category based on [2] (Project/Organization, Project, Constraint, Environment), and their impact on customizing the goal model

| Situational Factor | Category | Model Customization |
|--|----------------|---|
| Developers have little experience of software development | Project / Org. | Initialize the softgoal “Experienced Programmers” to “Denied” |
| The technology is new to the company | Project / Org. | Initialize softgoal “High Tech Knowledge” to “Partially Denied” |
| Developers are not always working at the same time/place | Project / Org. | Initialize the softgoal “Team members be Co-located” to “Partially Denied” |
| Face-to-face meeting with Scrum Masters are accessible only at certain times | Constraint | Add softgoal “Scrum Master be Available”, initialize to “Partially Denied” |
| Wiki pages are used for implementing daily meetings | Project | Add softgoal “Wiki Pages be Deployed” / Change the value of contrition links that connects “Daily Meeting” to “Higher Frequency of Com. With Business People”, and “Real-Time Knowledge Transfer” to “Hurt” |
| Use of emulator instead of actual devices | Environment | Add softgoal “Resources be Adequate” with initial value of “Partially Denied” |

Table 3 describes a number of significant situational factors of the illustrating scenario, their category (based on [2]), and their impact on model refinement. For initialization of model, both evidential and assumptive initializations have been used. For instance, since developers are junior programmers, we have enough evidences to initialize the “Experienced Programmers” to “Denied”. For requisites that we do not have solid evidences, we use assumptive initializations. For example, we assume the initial values of “Joint Decision Making” and “Collaboration be Viable” to be “Satisfied”. Fig. 4 shows the evidential initializations with green labels, and assumptive ones with red labels.

3.2.3 Model Evaluation

This phase iteratively propagates the initial (achievement) value of method requisites into the higher level elements of the integrated model. The proposed value propagation algorithm mainly follows the i^* forward evaluation algorithm [17] [18]. Value propagation will be performed with respect to the current value of contributing element and the value of contribution link, based on the propagation rules defined in Table 4. Since a contributed element might receive various values in this procedure, its “value bag” will be resolved based on the following value resolution rules [17]:

- Use the single label if the value bag has only one label
- Use the full label if:
 - The value bag has multiple labels of the same value
 - The value bag has labels with the same polarity with at least one full label
 - The previous human judgment produced a full label and new contribution is of the same value
- Use the minimum label if the bag has been filled through decomposition links. (Satisfied > Partially Satisfied > Conflict > Unknown > Partially Denied > Denied)
- Otherwise, use human judgment

Giorgini et al. in [19] introduced the concepts of *symmetric* and *asymmetric* contributions. When an element of a goal model symmetrically contributes to a softgoal, both its satisfaction and denial will be propagated to the softgoal. But in asymmetric contribution, the propagation will be performed either when the contributor is satisfied, or denied. The i^* forward evaluation algorithm considers all of the contribution relations as symmetric. However, this may not be appropriate for negative contributions (Hurt, Some-). For instance, when a method fragment (e.g. “Pair Programming”) makes a Hurt contribution to a softgoal (e.g. “More LOC per programmer per hour”), the denial of method fragment would not necessarily result in partial satisfaction of softgoal, while its satisfaction would result in partial denial of softgoal. Here, we modified the i^* forward evaluation algorithm by considering positive contributions (Help, Some+) as symmetric, and negative contributions (Hurt, Some-) as asymmetric, applicable only when the contributor is (Partially) satisfied.

Table 4. Propagation Rules for Contribution Links (adopted from [17])

| Original Label | | Contribution Link Type | | | | |
|----------------|---------------------|------------------------|------|--------|--------|---------|
| Label | Name | Help | Hurt | Some + | Some - | Unknown |
| ✓ | Satisfied | ✓ | ✗ | ✓ | ✗ | ? |
| ✓ | Partially Satisfied | ✓ | ✗ | ✓ | ✗ | ? |
| ? | Unknown | ? | ? | ? | ? | ? |
| ✗ | Partially Denied | ✗ | ? | ✗ | ? | ? |
| ✗ | Denied | ✗ | ? | ✗ | ? | ? |

Scenario: As an example of automatic resolution (not requiring human judgment), consider the objective “Better Understanding of Customer Needs”. The method fragment “Conduct Daily Meeting” has a “Help” contribution to this softgoal, and had been evaluated to “Partially Satisfied”. Therefore, based on the propagation rules of Table 4, a “Partially Satisfied” label will be added to the value bag of this softgoal. A similar label will be added to this bag, through the contribution of method fragment “Conduct Iteration Planning”. Therefore, this softgoal will be automatically evaluated to “Partially Satisfied”.

Manual resolution is needed where the automatic approach cannot be applied. In such cases, human judgment determines the value of the element, considering its value bag and status of its contributing elements. For example, consider the requisite softgoal “Be Productive” (a requisite of “Pair Programming”). This softgoal had received two “Partially Denied” and one “Partially Satisfied” labels through its contributing elements. We evaluated this softgoal to “Partially Denied”, based on the perception that the contributions of adequate resources and programming experience to the productivity of programmers are more significant than the similarity of their personality. Fig. 4 shows the result of evaluating an integrated model, developed for the “Effective Communication”.

4 Validity of Framework

Although the motivating scenario introduced in Section 2 was fictitious, we had experienced a similar scenario in a classroom setting. A software development method was created by combining the above-mentioned method fragments, and used for the major project in a second-year undergraduate computer science course, with 40 4-person teams. The project was to build an application for Blackberry devices. Most team members had little programming experience. Furthermore, the Scrum masters, who were course teaching assistants (TAs), had limited availability (their office hours), and were not so familiar with mobile programming.

As an initial test of framework validity, we compared the evaluation results generated by framework with what was observed in the classroom. For this comparison, we considered 18 minor objectives that were contributing to the three major objectives.

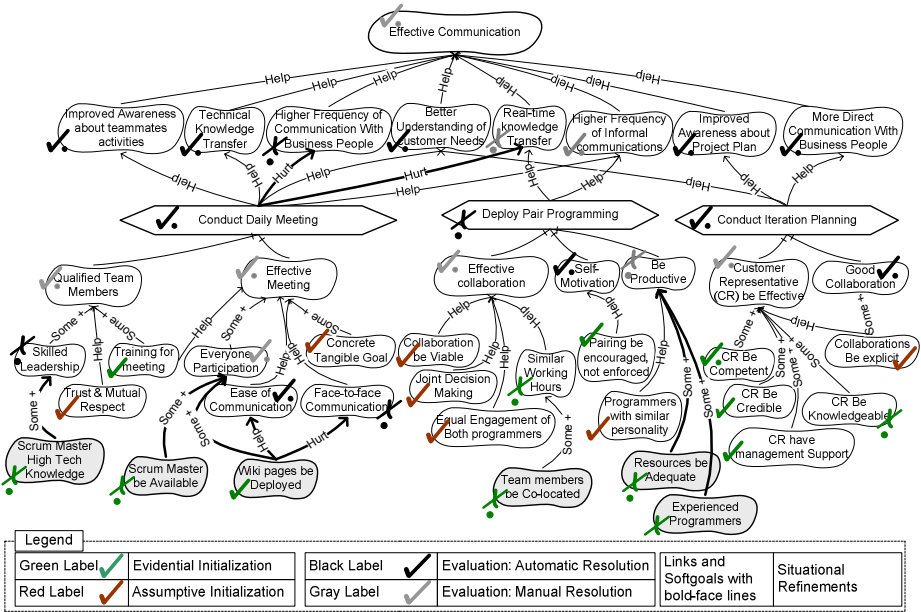


Fig. 4. Modification, Initialization, and Evaluation of an integrated model

The comparison showed that for 12 softgoals (66%), the framework-generated values were the same as the observed ones. For 4 of the softgoals (22%), the observed status of method objectives were better than what the framework suggested. In most cases the framework-generated values were “Partially Denied”, whereas the observed values were “Partially Satisfied”. For 2 of the softgoals (12%), the observed values were worse than what the framework anticipated. Thus overall, the framework produced results that were more conservative than actually observed.

5 Discussion and Future Work

This paper proposed a framework for anticipating the attainability of the objectives of a development method in a particular project situation. The framework aims to help project managers in deciding on a set of suitable method fragments. Methods for selecting method fragments have been proposed, e.g. Situation-Scenario-Success (S^3) [2] and Assembly Process Model (APM) [3], but they do not focus on the evaluation of objectives. A method fragment evaluation technique was proposed in [20], which also deals with capturing the evaluation experiences in method fragments repository. The proposed framework adopts a goal-oriented approach for analysis, modeling, and evaluation of method fragments. Similar approaches have been acknowledged in [19, 21, 22].

The reliance of this framework on the results of empirical studies has both positive and negative sides. On the positive side, it can help project managers by providing an evidence base, supported by the experiences of many practitioners and researchers on deploying different method fragments. On the negative side, reliance on the extensive body of empirical evidence can be subject to misinterpretation, inadequate or unreliable evidential data, and conflicting scenarios (e.g., same situation, different results). These negative effects can be moderated by relying mostly on empirical studies that had been peer reviewed and published in highly regarded conferences or journals and, if possible, whose results have been replicated.

One of the important factors that we considered in the design of this framework is that it should not impose excessive overhead to software organization. The provision of a model fragment repository is an initiative towards this end. Although this claim needs to be verified in our further industrial case studies, we expect that the pre-constructed model fragments facilitate the use of framework. So far, we have populated the framework repositories with an initial collection of 15 method fragments. More are being added in an ongoing project. As future work, we are developing automated tools to support the process of model merging, and will take advantage of the OpenOME² tool for the evaluation of goal models.

An underlying premise for this framework is to focus on method fragments rather than on the prescription of an entire process. This approach is in line with the idea of agility while doing software process improvement (SPI) [23]. Unlike traditional SPI frameworks that focus on improving the maturity of software processes, a goal-oriented agile SPI framework would focus on improving the effectiveness of software processes with respect to stated objectives. In future work, we intend to use the method fragment evaluation framework as a part of an agile SPI framework, in which the local improvements (improving method fragments) provide the basis for global optimization (improved overall process).

Limitations of the framework include reliance on human judgment during qualitative evaluation, and when there is no adequate evidential data. For instance, during the model refinement and initialization phase, the correct identification of significant situational factors is crucial. This issue has been widely studied in situational method engineering [2]. In this framework we have tried to propose a simple approach for modeling project situation factors. Another limitation of this framework is the value propagation and resolution phase, in which the manual resolution of value bags is subject to personal judgments. Of course the experience of model evaluator is important here, but we hope to reduce the risks of such subjective decisions by enriching the evidence base. However, the evidence base is unlikely to cover all possible situations as there might be always some short-cuts to achieve goals, or some hidden factors that impede certain objectives of method fragments, regardless of their stated requisites. Nevertheless, the proposed framework can potentially help process managers make better-informed decisions based on the growing body of empirical evidence.

² An Eclipse-based tool for modeling and evaluation of *i** goal models, available on-line at: <https://se.cs.toronto.edu/trac/ome>

Appendix: List of Referenced Studies

| Code | Approach | Reference |
|------|-----------------------|---|
| S1 | web-based survey | Begel, A., & Nagappan, N. (2007). Usage and Perceptions of Agile Software Development in an Industrial Context: An Exploratory Study. <i>First International Symposium on Empirical Software Engineering and Measurement, 2007 (ESEM 2007)</i> . , 255-264. |
| S2 | Observatory | Sutherland, J., Viktorov, A., Blount, J., & Puntikov, N. (2007). Distributed Scrum: Agile Project Management with Outsourced Development Teams. <i>40th Annual Hawaii International Conference on System Sciences, 274a</i> . |
| S3 | Experience Report | Given, P. (2006) Scrum in a Fixed-Date Environment. accessed via: http://www.scrumalliance.org/articles/29-scrum-in-a-fixeddate-environment , in May, 2009 |
| S8 | Experience Report | Moore, R., Reff, K., Graham, J., & Hackerson, B. (2007). Scrum at a Fortune 500 Manufacturing Company. <i>Agile</i> , 175 - 180. |
| S12 | Experience Report | Berczuk, S. (2007a). Back to Basics: The Role of Agile Principles in Success with an Distributed Scrum Team. <i>Agile</i> , 382-388. |
| S15 | Ethnographic study | Chong, J., & Hurlbutt, T. (2007). The Social Dynamics of Pair Programming. <i>Proceedings of the 29th international conference on Software Engineering, 354-363, IEEE Computer Society</i> . |
| S17 | Controlled Experiment | Sfetsos, P., Stamelos, I., Angelis, L., & Deligiannis, I. (2009). An experimental investigation of personality types impact on pair effectiveness in pair programming. In <i>Empirical Software Engineering, 14(2)</i> , 187-226. |
| S30 | Experience Report | O'Donnell, M. J., & Richardson, I. (2008). Problems Encountered When Implementing Agile Methods in a Very Small Company. (Ed.) In: <i>Software Process Improvement</i> (pp. 13-24). Springer |

References

1. Brinkkemper, S.: Method engineering: engineering of information systems development methods and tools. *Information and Software Technology* 38(4), 275–280 (1996)
2. Harmsen, A.F.: *Situational Method Engineering*, Utrecht, Moret Ernst & Young Management Consultants (1997)
3. Ralyté, J., Rolland, C.: An Assembly Process Model for Method Engineering. In: Dittrich, K.R., Geppert, A., Norrie, M.C. (eds.) *CAiSE 2001*. LNCS, vol. 2068, pp. 267–283. Springer, Heidelberg (2001)
4. Saeki, M.: CAME: The first step to automated method engineering. In: *Workshop on Process Engineering for Object-Oriented and Component-Based Development*, Anaheim, CA (2003)
5. Henderson-Sellers, B.: Method engineering for OO systems development. *Communications of the ACM* 46(10), 73–78 (2003)
6. Bajec, M., Vavpotic, D., Krisper, M.: Practice-driven approach for creating project-specific software development methods. *Information and Software Technology* 49(4), 345–365 (2007)

7. Linda, W., Mark, K.: Software project risks and their effect on outcomes. *Communications of ACM* 47(4), 68–73 (2004)
8. Beck, K., Beedle, M., Bennekum, A.V., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R.C., Mellor, S., Schwaber, K., Sutherland, J., Thomas, D.: *Manifesto for Agile Software Development* (2001)
9. Kitchenham, B.A., Dyba, T., Jorgensen, M.: Evidence-Based Software Engineering. In: *Proceedings of the 26th International Conference on Software Engineering*. IEEE Computer Society, Los Alamitos (2004)
10. Schwaber, K., Beedle, M.: *Agile Software Development with Scrum*, p. 158. Prentice Hall PTR, Englewood Cliffs (2001)
11. Ralyté, J., Deneckère, R., Rolland, C.: Towards a Generic Model for Situational Method Engineering. In: *Advanced Information Systems Engineering*, p. 1029 (2003)
12. Firesmith, D.: *Open Process Framework (OPF)*, accessible via: date accessed (November 2009)
13. Yu, E.: Towards modelling and reasoning support for early-phase requirements engineering. In: *Proceedings of the Third IEEE International Symposium on Requirements Engineering (received Most Influential Paper After 10 Years Award at RE 2007)* (1997)
14. Yu, E., Mylopoulos, J.: Understanding “why” in software process modelling, analysis and design. In: *Proceedings of the 16th international conference on Software engineering*, Sorrento, Italy. IEEE Computer Society Press, Los Alamitos (1994)
15. Slooten, K.V., Brinkkemper, S.: A Method Engineering Approach to Information Systems Development. In: *Proceedings of the IFIP WG8.1 Working Conference on Information System Development Process*. North-Holland Publishing Co., Amsterdam (1993)
16. Coulin, C., Zowghi, D., Sahraoui, A.-E.-K.: A situational method engineering approach to requirements elicitation workshops in the software development process. *Software Process Improvement and Practice* 11(5), 451–464 (2006)
17. Horkoff, J., Yu, E.: Using the i* Evaluation Procedure for Model Analysis and Quality Improvement presentation. In: *Second International Workshop on i* / Tropos*. University College London, London (2005)
18. Horkoff, J., Yu, E.: A Qualitative, Interactive Evaluation Procedure for Goal- and Agent-Oriented Models. In: *Proceedings of CEUR Workshop in CAiSE 2009* (2009)
19. Gonzalez-Perez, C., Giorgini, P., Henderson-Sellers, B.: Method Construction by Goal Analysis. In: *Proceedings of Int. Conf. on Information System Development*. Springer, US (2007)
20. Ralyté, J., Jeusfeld, M.A., Backlund, P., Kühn, H., Arni-Bloch, N.: A Knowledge-based Approach to Manage Information Systems Interoperability. *Information Systems, Special issue on Advances in Data and Service Integration* 33(7-8), 754–784 (2008)
21. Ågerfalk, P.J., Fitzgerald, B.: Exploring the concept of method rationale: A conceptual tool for method tailoring. In: *Advanced Topics in Database Research*, pp. 63–78 (2006)
22. Rossi, M., Ramesh, B., Lyytinen, K., Tolvanen, J.: Managing Evolutionary Method Engineering by Method Rationale. *Rationale Journal of the Association for Information Systems* 5(9), 356–391 (2004)
23. Aaen, I., Borjesson, A., Mathiassen, L.: SPI agility: How to navigate improvement projects. *Software Process: Improvement and Practice* 12(3), 267–281 (2007)

Incorporating Modules into the *i** Framework*

Xavier Franch

Universitat Politècnica de Catalunya (UPC)
c/Jordi Girona, 1-3, E-08034 Barcelona, Spain
franch@lsi.upc.edu

Abstract. When building large-scale goal-oriented models using the *i** framework, the problem of scalability arises. One of the most important causes for this problem is the lack of modularity constructs in the language: just the concept of actor boundary allows grouping related model elements. In this paper, we present an approach that incorporates modules into the *i** framework with the purpose of ameliorating the scalability problem. We explore the different types of modules that may be conceived in the framework, define them in terms of an *i** metamodel, and introduce different model operators that support their application.

Keywords: goal-oriented models, *i**, modularization, modules, scalability.

1 Introduction

The *i** framework [1] is currently one of the most widespread goal- and agent-oriented modelling and reasoning frameworks. It has been applied for modelling organizations, business processes, system requirements, software architectures, among others.

Several challenges have been identified with the goal of overcoming different reported problems (see e.g. [2]). Among them, one of the most important issues is to make *i** models more manageable and scalable by defining modularity constructs. This paper presents a proposal for converting *i** into a modular language. This is a basic notion for any language expected to create big models as *i** is, but it has been not yet proposed for *i** except for some proposals of new constructs in the language. Instead, we do not propose to extend the language, but to add modularity facilities to the metamodel of *i** in a loosely coupled way.

The rest of the paper is structured as follows. Section 2 provides the background on case studies and empirical evaluation as well as related work. Section 3 presents an *i** metamodel to be used for reference the rest of the paper. Section 4 proposes the different types of modules for the *i** language. Section 5 defines two module operations, combination and application. Section 6 includes some discussion about the presented work. Finally, Section 7 states the conclusions and future work.

Basic knowledge of *i** is assumed in the paper, see [1] and the *i** wiki (<http://istar.rwth-aachen.de>) for a thorough presentation.

* This work has been partially supported by the Spanish project TIN2007-64753.

2 Background and Related Work

2.1 Use of i^* in Industrial Projects

Some industrial experiences on the use of i^* have been documented. In [3], a report is presented about three air traffic management projects in which i^* was applied to model requirements for new socio-technical systems. Among other remarks, the problem of managing large SR models is highlighted. This large size cannot be lowered due to the absence of structuring mechanisms.

In [4] an experience is reported about how to support the continuous alignment of corrective software maintenance processes with the strategic goals of a Software Design Maintenance Organization at Ericsson Marconi Spa. The authors used a model slicing technique to break the model into pieces. However, this partition was done by hand since the authors did not bring the notion of module into the i^* framework, with the inherent drawbacks of proceeding this way.

In [5], the authors report the use of i^* for architecting hybrid systems in two industrial experiences at the Etapatelecom Ecuadorian company. One of the cornerstones of the proposed method is the reconciliation of the individual models that the different stakeholders build. The models have not been encapsulated into modules due to the absence of this capability in the i^* framework, making this process more difficult to implement. Also reusability is highlighted as a key concept, but at the time being it has been supported simply by manual management of the models.

As a summary of these cases, the existence of modularity constructs in the i^* framework's language could help to ameliorate some of the problems mentioned.

2.2 Empirical Evaluation of i^*

At the time being, the only in-depth empirical evaluation of the i^* framework reported in the community we are aware of is [2]. The authors propose a feature-based evaluation scenario and they assess i^* with respect to these features in the light of three industrial projects. The results of the analysis is that i^* supports well the expressiveness and domain applicability features, provides some insufficient support to the refinement, repeatability, complexity management and traceability features, and does not support at all modularity, reusability and scalability features. It is also stated that reusability and scalability have a causal relationship with modularity, which means that providing some solution to the latter feature impacts on the former. We may conclude then that providing a solution to the modularity problem can be a topmost productive effort in terms of improving the evaluation of the i^* framework.

2.3 Existing Approaches to Model Modularity

Although we have stated that the i^* framework does not include modularity constructs, there are some lines of research addressing this issue. The two most remarkable contributions at this respect are the incorporation of aspects and services into i^* .

Concerning the first point, a line of research [6][7] proposes the use of aspects for modelling cross-cutting concerns. Although it is true that separation of concerns may help to structure the i^* models, the proposal still does not include modules to support

the basic concept of stepwise refinement. Also, the addition of aspects into i^* results in a framework that is more complex and may eventually require a steeper learning curve. Therefore we do not consider this proposal as a general solution for the modularization problem.

In [8], the concept of service is incorporated into the i^* framework. This type of modularity unit is closer to the concepts managed in the domain (i.e., business services) and from this point of view fits better than aspects to the natural stepwise refinement process. However it is true that this particular proposal introduces a lot of complexity to the framework, with the fundamental concepts of “service” and “process”, and also with the configuration of services inside SR boundaries using a variability-like model with mandatory and optional features combined in several ways.

In this work, we have preferred to search for other solutions that do not require the addition of new constructs into the i^* framework and that are basic enough to be bound to different concepts in different methods.

3 The i^* Metamodel

The i^* community has defined several dialects of i^* that add new constructs for particular purposes (e.g., trust constructs, temporal constructs, ...), remove some that are not of primary interest for their purposes (e.g., types of actors) or modify some conditions of use (e.g., which type of intentional element is a valid end for a means-end relationship). In [9][10] we provide a thorough analysis of these variations. Also, in [11] we may find a survey of variations used by the community in several proposals. Variations occur both in Strategic Dependency (SD) and Strategic Rationale (SR) diagrams. This diversity makes advisable to identify which constructs we do consider.

Following our previous work [9][12], we propose in this section an i^* metamodel that is used as reference in the rest of the paper. The metamodel is built under the principles of generality (i.e., trying to host as most as possible the existing variations), extensibility (for incorporating future extensions) and suitability for modularity (being this the goal of the paper). As a result, we remark here the most important innovations with respect to the metamodel we have proposed so far:

- We have added an abstract *Link* class that holds binary relationships among *Nodes* (being *Node* the general concept of i^* model element). This *Link* class generalises the concepts of link among actors (e.g., is-A relationships), among SR elements (e.g., means-end link) and among dependums (and thus dependencies). These three particularizations are considered abstract links themselves specialized into the available types of links in a uniform way. As a modelling decision and for clarity purposes, we provide the conditions that each particular type of link may impose (e.g., *Covers* is a many-to-many association from position *Actors* to role *Actors*) as OCL constraints instead of graphically (although we show the resulting dependencies to make evident this relationship).
- For illustration of dependum links, and considering the goal of this paper, we have added a class for the *Support* dependum link as introduced in [13]. A dependum $d1$ supports another dependum $d2$ when $d1$ has been introduced in a later development stage than $d2$ in a way that $d1$ provides details about the form that $d2$ has.

The resulting metamodel is shown in Fig. 1. We also show some representative OCL constraints, especially to illustrate how the different levels of abstraction in the class diagram have also their counterpart in the OCL constraints (see Table 1). In the following sections, we will define new classes and associations corresponding to the modularity constructs that will be linked with the required elements of this metamodel.

4 Types of Modules

In this section we present the two types of modules we envisage for encapsulating meaningful i^* model pieces: SR modules and SD modules. Both types are subclasses of a more general class that declares the common attributes of interest, at least the name of the module and other required information not relevant for this paper (e.g., metadata as author, date, etc.). In addition, also the whole model can be encapsulated in a module, in which case the metamodel of Fig. 1 describes the allowed contents.

4.1 SR Modules

SR modules are the most obvious type of module because the elaboration of SR models relies upon the application of several kinds of refinement operators.

According to the metamodel presented in Section 3, the two usual kinds of refinement operators are decomposition of tasks and identification of means for an end. Also, softgoal contributions need to be considered. However, aligning with the general guidelines of our approach, we define first the general concept of SR module and then show how to customize it to the cases above, leaving open the door for incorporating further types of modules if new decomposition operators are proposed.

Fig. 2 shows the connexion of SR modules to the i^* metamodel and Table 2 lists some additional integrity constraints expressed in OCL, which need to be considered as additions to the ones already defined in the metamodel. In its more basic form, an SR module is composed of SR elements and links among them. Upon this basic structural form, we have added as few additional constraints as possible to allow defining in the future different types of SR modules:

- Multiplicities show that the module shall contain at least two SR elements. Also, an OCL constraint (not shown) requires at least one link among them.
- At least one of the SR elements shall be a root (see the definition of *root* at Table 2). We have considered that constraining to one single root could be unnecessarily restrictive (as illustrated below).
- From the root elements, all other intentional elements shall be reachable (see *All_SR_Elements_Reachable_From_Roots* in Table 2). That is, no unconnected partitions are allowed since we consider that they would represent different conceptual units that would require encapsulation in different modules.
- We remark that we do not impose any restriction on the decomposition depth. This means that the decomposition complexity is left up to the modeller's decision.

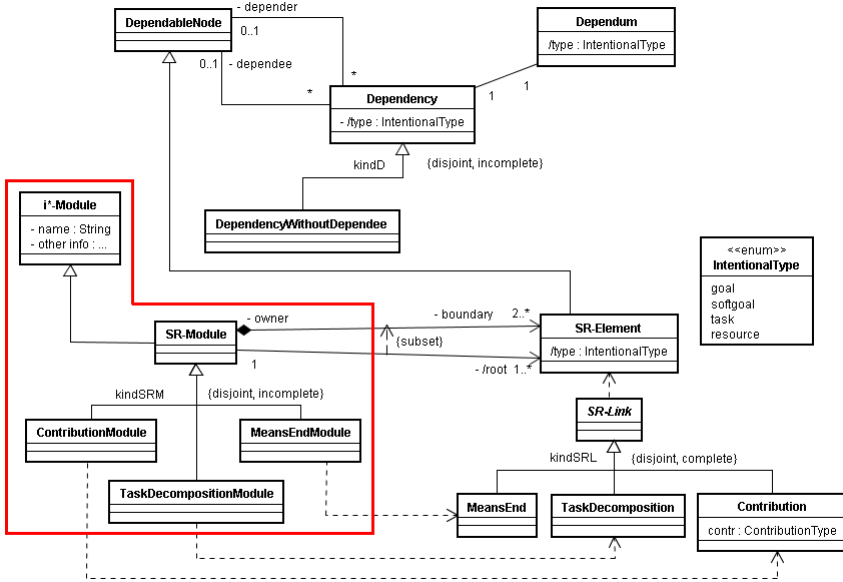


Fig. 2. Integrating SR modules with the *i** metamodel

Table 2. Constraints over SR modules

| General constraints on SR Modules | |
|--|---|
| context SR-Module | inv All_SR_Elements_Reachable_From_Roots: let descendants(x: Set(SR-Element)): Set(SR-Element) = x->union(descendants(x.from)) in descendants(self.root)->includesAll(self->boundary) |
| context SR-Module | inv All_Dependencies_Are_Without_Dependees: self.boundary->forall(x x.dependency[depender] ->forall(d d.ocIsTypeOf(DependencyWithoutDependee)) |
| Particular constraints on particular types of SR Modules | |
| context TaskDecomposition-Module | inv Valid_Task-Decomposition_Module: self.root->select(x x.type = task)->size() = 1 and self.root->reject(x x.type = task)->forall(x x.type = softgoal) and self.root->select(x x.type = task).link[to] ->forall(l l.ocIsTypeOf(TaskDecomposition)) and self.root->select(x x.type = softgoal).link[to] ->forall(l l.ocIsTypeOf(Contribution)) |
| context MeansEnd-Module | inv Valid_Means-End_Module: ...similar to the one above |
| context Contribution-Module | inv Valid_Contribution_Module: self.root->size() = 1 and any(self.root).type = softgoal and self.root.link[to]->forall(l l.ocIsTypeOf(Contribution)) |

This basic form could be enough in those cases where the intentional elements may fulfil the required goals by themselves. But most often, they will require the collaboration of other actors and this will be represented, as usual, by dependencies. The fundamental point here is just to show the dependers and dependums of those dependencies, not the dependees (see *All_Dependencies_Are_Without_Dependees*). As a

result, the SR module does not include any assumption about what intentional element will collaborate with these dependers. The connection of the dependencies defined in the SR module and the surrounding actors will be established as part of the module operations (see Section 5). The structure itself of the enlarged metamodel ensures that dependers are always SR elements (i.e., dependencies at the level of actor are not allowed when decomposing at the SR level).

SR modules in general may contain any arbitrary decomposition of elements. From this general form, we define three different types of SR modules that appear as specializations of the SR module class in the class hierarchy (we remark that the partition is incomplete). Their particular constraints are shown in Table 2.

- Task-decomposition SR modules. The intentional element of interest is a task decomposed into subelements. These subelements may be further decomposed. The intentional elements that appear in this multi-level decomposition may contribute to softgoals (that are also roots in the diagram), and these contributions may also be included in the module.
- Means-end SR modules. The intentional element of interest is a goal whose means are tasks. As happened above, tasks may be further decomposed and all the intentional elements may contribute to softgoals.
- Contribution SR modules. They identify intentional elements that contribute to one softgoal. In this case, we consider methodologically convenient to allow just one root, namely the softgoal of interest. Also just intentional elements that directly contribute to the root softgoal are included.

New types of modules could be eventually defined by adding specializations with the corresponding integrity constraints.

Fig. 3 shows examples of these modules. At the left we have a contribution module with no stemming dependencies, whilst on the right a means-end module states the need of collaboration with some undefined actor and a contribution to softgoal.

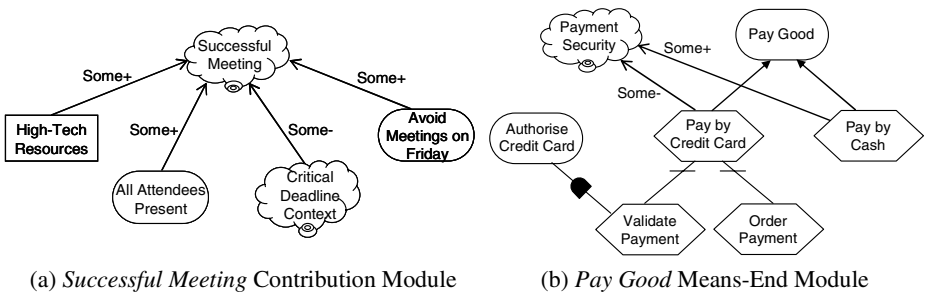


Fig. 3. Examples of SR modules

4.2 SD Modules

This type of module is conceived to contain subsets of actors and dependencies among them. In their general form, SD modules encapsulate actors and dependencies without any restriction. From a methodological point of view, it is interesting to

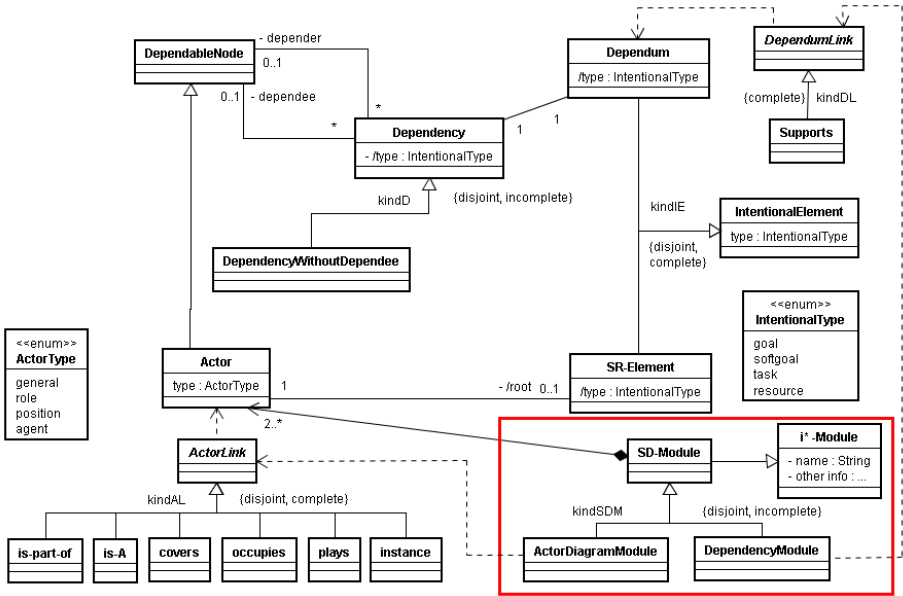


Fig. 4. Integrating SD modules with the *i** metamodel

designate some particular types of SD modules that seem convenient to define, namely actor diagram modules and dependency modules.

Fig. 4 shows the integration of SD modules into the *i** metamodel and Table 3 shows some representative constraints. Some similarities may be established with SR modules: (1) a minimum of two actors are required (since recursive dependencies are not allowed by the metamodel), as well as at least one link or dependency among them (by an OCL constraint not shown); (2) all the actors should be interconnected somehow, otherwise they would represent different abstractions and should be encapsulated in different modules; (3) there is no restriction about the number of actors, links or dependencies, it’s up to the modeller to decide the appropriate size of the module; (4) dependencies are established among actors, not intentional elements; (5) there may be some dependencies from actors that have not dependees inside the module, mixed with dependencies whose both ends are actors belonging to the module (see Fig. 5, (a)). Furthermore, for methodological reasons (see [13]) we allow actors to include a primary objective in the form of an intentional element inside their boundaries. This way, the SD diagram may declare the overall intention of its enclosed actors. Even in this case, the dependencies are between actors. To reinforce that these goals are roots, SR links are not allowed in SD diagrams. OCL constraints take care of these conditions.

The two specializations of the general concept of SD module are defined as:

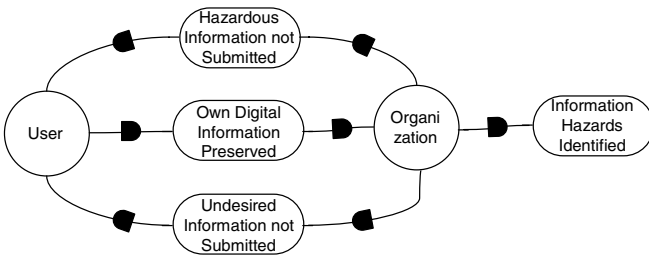
- Actor diagrams SD modules. The module just contains actors and links among them, i.e. no dependencies are included, see [14]. This module recognizes the rich variability of actor types and their relationships by creating networks of roles,

positions and agents with specialization and aggregation information. See Fig. 5 (b) for an example.

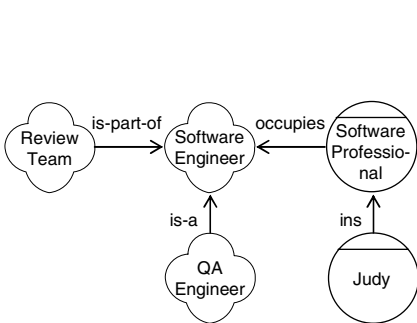
- Dependency SD modules. The module just contains a dependency of interest between two actors and then some supporting dependencies (making use of the *support* dependum link), that may be decomposed at their turn. Therefore, we can refine dependencies in a similar way than SR elements. In Fig. 5 (c) we show an example that reflects the refinement process as mutual needs of both actors.

Table 3. Constraints over SD modules

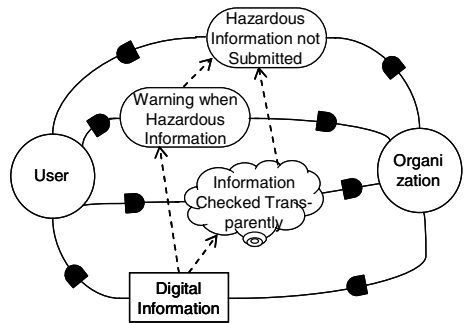
| General constraints on SD Modules | |
|--|--|
| context | SD-Module inv All_Actors_Are_Connected: self.actor->forall(x,y existsPath(x, y)) -- auxiliary function not included |
| context | SD-Module inv There_Are_Not_Dependencies_Without_Dependor: not self.actor.dependency[dependee].oclIsTypeOf(DependencyWithoutDepender) |
| context | SD-Module inv No_SR-Links_Allowed: self.actor.root->forall(SR-Link[from]->isEmpty() and SR-Link[to]->isEmpty) |
| Particular constraints on particular types of SD Modules | |
| context | ActorDiagramModule inv No_Dependencies_Allowed: self.actor.dependency[depender]->isEmpty() and self.actor.dependency[dependee]->isEmpty() |
| context | DependencyModule inv Just_Two_Actors: self.actor->size() = 2 |



(a) General SD Module



(b) Actor Diagram Module (excerpt)



(c) Dependency Module

Fig. 5. Examples of SD modules

5 Module Composition

Once module types and their valid contents have been defined, it is necessary to formulate the operations needed to manage them. Basically we need to cover two particular needs: merging two modules into one, and including a module into a model.

5.1 Model Combination

Fig. 6 defines an abstract module combination operation at the level of the *i*-Module* superclass. This operation fixes common preconditions and postconditions with some protected auxiliary functions (refinement not included), to be enriched in the subclasses. It is worth noting along the section that, since modules and models are defined as composition of elements, node comparison is done not by oid but by identifier.

```

combine(a1: i*-Module, a2: i*-Module, name: String, other info...)
  /* there is not a module with the name of the new one */
pre not i*-Module.allInstances().label->includes(name)
  /* common nodes are of the same type */
pre compatibleNodes(allNodes(a1), allNodes(a2))
  /* the new module has been created */
post oclIsNew(s) and
  s.label = name and s.otherInfo = other info... and oclIsTypeOf(i*-Module)
  /* nodes in the new module are the union of those from starting modules */
  and sameNodes(allNodes(s), allNodes(modelUnion(a1, a2)))
  /* nodes in the new module are compatible to those from the starting mods. */
  and compatibleNodes(allNodes(s), allNodes(modelUnion(a1, a2)))

```

Fig. 6. The combination operation in the superclass

A question arises to know which type of module results from the combination of two modules. In some cases the question is straightforward, e.g. the case of combination of Actor Diagram modules, which results in another module of the same type if preconditions hold. In other cases the answer depends on the contents of the module, e.g. when combining two Task Decompositions modules A and B, if A's task root appears as a leaf inside B, then the operation yields another Task Decomposition module, otherwise the result does not comply with the constraints on this type and needs to be considered as an instance of the more general concept of SR module.

```

combine(a1: ActorDiagramModule, a2: ActorDiagramModule,
  name: String, other info...)
  /* there is at least one actor in common */
pre a1.actor.label->intersection(a2.actor.label)->isNotEmpty()
  /* the result is also an Actor Diagram Module */
post oclIsNew(s) and s.label = name and s.oclIsTypeOf(ActorDiagramModule)

```

Fig. 7. Module combination operation: the Actor Diagram Module case

Fig. 7 illustrates the refinement for the case of Actor Diagram modules combination. A new precondition demanding at least one common actor is requested to ensure the invariant of this type of module. Also, the type of this kind of combination is detailed. The rest of conditions are fulfilled by inheriting the superclass definition

(which is more general than needed, e.g. Actor Diagram modules do not have SR links, but this is not a problem). This particular example of combination refinement is quite straightforward since by definition this type of module does not have dependencies, see 5.2 for this case.

5.2 Module Application

Application of a module over an *i** submodel or element relies on the same principles and in fact, several auxiliary functions appearing in the OCL definition will be shared.

```

apply(m: Model, a: i*-Module, depMtc: Set(dpdm: Dependum, x: DependableNode))
/* common nodes are of the same type */
pre compatibleNodes(allNodes(m), allNodes(a)) -- nodes in m not in a are not
/* the dependency matching is correct */ -- considered
pre depMtc->forAll(
  allNodes(a)->includes(dpdm) and
  dpdm.dependency.isOclTypeOf(DependencyWithoutDependee) and
  allNodes(m)->includes(x) and not allNodes(m).label->includes(dpdm.label)
  and compatibleLinkEndPoints(dpdm.dependency.depender, x))
/* the nodes in the module are included in the model */
post hasNodes(m, allNodes(a))
/* the nodes keep being compatible after the application */
post compatibleNodes(allNodes(m), allNodes(a))
/* the matching has been applied in the model */
post depMtc->forAll(
  allNodes(m).label->includes(dpdm.label)) and
  allNodes(m)->select(label = dpdm.label).
  dependency.depender.label = dpdm.dependency.depender.label and
  allNodes(m)->select(label = dpdm.label).dependency.dependee = x))
    
```

Fig. 8. Applying an *i** module to an *i** model

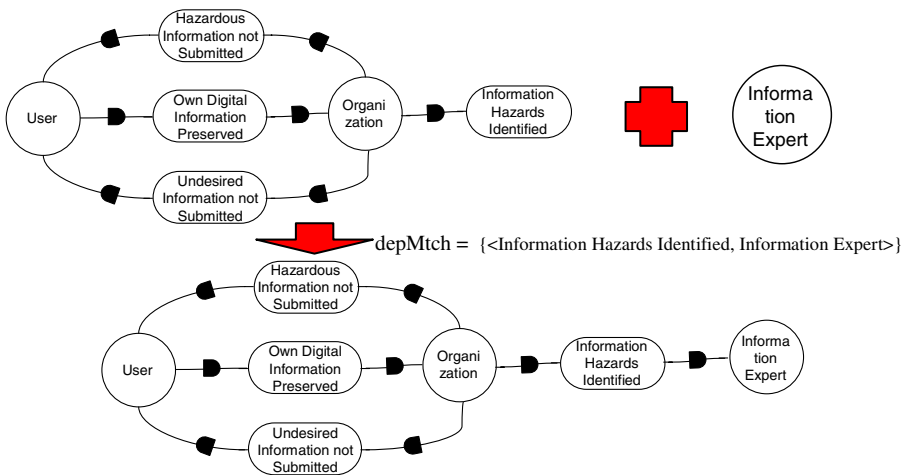


Fig. 9. Example of application of an SD module (left, from Fig. 5) to a model excerpt (an actor)

Fig. 8 shows the general application function. We remark here the connection of dependencies stemming out of the module. The header of the function includes a matching from dependums of the module to dependable nodes of the model. Correctness of this matching requires the dependum to correspond to a dependency without dependee, and compatibility of the already defined depender and the proposed dependable node that acts as dependee in the resulting node. Note that the matching may be partial, meaning that after its application some dependencies may remain without dependee (of course, this means that the model is still incomplete).

Fig. 9 provides a very basic example applying the module in Fig. 5(a) to a model that includes an actor, being in this case the matching: *Information Hazards Identified* → *Information Expert*.

In Fig. 10 we refine the general application function to the particular case of applying a Task Decomposition module to an *i** model. In this case it is necessary to add a precondition to check that the task decomposed in the module is not yet decomposed in the model.

```

apply(m: Model, a: TaskDecompositionModule)
let theTask: SR-Element = a.root->select(type = task) in -- the root task of a
/* the task is in the model and it is not decomposed (it is a leaf) */
pre m.boundary.label->includes(theTask.label) and
m.boundary->select(label = theTask.label).to->isEmpty()

```

Fig. 10. Module application operation: the Task Decomposition Module case

6 Discussion and Further Issues

In Sections 4 and 5 we have presented the types of modules and two basic module management operations. We have presented in detail the structure of the modules and the specification of operations. But there are some additional issues that we have not tackled due to lack of space that we enumerate below.

Relationships among modules. In the proposal, relationships between modules are implicit: a module is related with another if some component of the former is bound somehow with some component of the latter. This kind of relationship is quite basic and could be improved by defining semantic ones. We envisage two types of such relationships. On the one hand, adding rationale to the implicit relationship mentioned above. On the other hand, refinement-related relationships, in which it may be established, for instance, that a module refines another, or that it is a different version, etc.

In addition to this, structural relationships among modules can be convenient, e.g. composition or nesting. The UML metamodel may be a source of inspiration with the objective of having a uniform treatment with respect to this widespread modelling notation. Nevertheless, trade-offs need to be assessed (see future work at Section 7).

Concept-driven modularization. The proposal presented here is not much linked to the problem domain, since the criteria used to identify the modules is not established explicitly. In the specializations of SR modules, it may be argued that the criterion is the intentional element to be decomposed. This also happens in Dependency modules.

But in the general case, the criterion is missing. A simple solution to this problem is to add the *Property* class from the metamodel to the modules metamodel. For instance, going back to Section 2, one property could be Business Service, and then each different business service in [8] could be represented by a module. The same applies to the Etapatelecom case, where the property could simply be Stakeholder and then each stakeholder may have her own model encapsulated in a module.

Model matching and model integration. In this work, operations defined in Section 5 are binding elements by name. But more sophisticated forms may be defined. The most immediate extension is to provide a mapping of names. In fact, this extension is almost mandatory if we think about reusability (see below). But also we may think of more sophisticated integrations. For instance, one possibility is to consider i^* model merging as proposed in [15]. We remark that the consideration of these proposals impacts on the definition of the module combination and application operations but not in the proposal of modules presented in this paper.

Reusability. A natural consequence on having a modularization structure available is thinking about reusability. Currently, reusability is just a copy-and-paste process due to this lack of modularity. Having modules available makes it easier to organize repositories of modules. Several consequences can be listed.

First, new types of modules could be considered, for instance, a specialization of SR module called Actor module that contains all the rationale of an actor, ready to be reused. In this reusability context, the most general type of module, corresponding to a whole model (as mentioned at the beginning of Section 4), would surely play a prominent role.

Second, it may be convenient to add incoming dependencies to modules. These incoming dependencies would synthesise the intentionality provided by the (sub)model encapsulated in the module. Therefore, it would not be necessary to analyse any single intentional element inside that (sub)model to reuse the actors.

Last, as mentioned above, in the reusability context, binding by name as proposed here is clearly insufficient and at least a name mapping is required.

Views. The proposal of this paper is oriented to provide support to the modeller whilst developing the models, and facilitate their latter understandability and maintainability. Another possibility could have been to define views over i^* models that could eventually be stored in modules if required. Views are a powerful mechanism to extract information from models and in fact it is natural to think in this to make modular existing models. This is the idea followed in [6][7] to create aspects from existing i^* models. But still the problem to build the model remains, and scalability is still an issue. Therefore, we see views not as a different alternative but complementary to our proposal.

Tool-support. At the bottom line, modules are a model management mechanism and not a fundamental ontological construct in i^* . Because of their operational nature, tool support is fundamental to make them usable. In fact, a good tool support for this proposal should hide the metamodel details presented here and provide functionalities that represent practical needs for the modeller whilst constructing her model.

7 Conclusions and Future Work

In this paper we have proposed some modularity constructs for structuring i^* models. These constructs have taken the form of building blocks, i.e. modules, together with a model combination operator. We have determined the i^* metamodel, and then defined those modules as new classes that refer those metamodel elements. Finally, we have defined some module operations and outlined some further issues.

In the rest of the section, we assess the modularity proposal using the features defined in [2].

Features considerably improved

- *Modularity*. According to [2], this feature is not currently supported by i^* . The main rationale was “[...] i^* doesn’t have mechanisms for using building modules [...]”. We have tackled this issue in the paper. In [2] the emphasis was on defining building modules for business processes. In our approach, we have adopted a more neutral view, presenting the building modules more related to the structure of the models than to the ontology of the domain. As a consequence, modules may be eventually bound to whatever concept may be considered of primary interest.
- *Refinement*. [2] states the need of “[...] incrementally add more detail [...] until we reach concrete models of business processes and their actor dependencies”. Having building modules impacts positively in this stepwise refinement of i^* models, since modules can be used to encapsulate elements that are at the same level of abstraction.
- *Complexity Management*. This feature is defined in [2] as “the capability of the modelling method to provide a hierarchical structure for its models, constructs and concepts”. In this proposal, since an element that appears in a module may, at its turn, be decomposed itself, there is an implicit hierarchy of models (i.e., those that are enclosed in the modules) and thus of constructs and concepts (as part of the models).
- *Reusability*. As stated in [2], “this feature is causally related to modularity”, thus it can be said that the existing modules are providing the basic foundations for reusability of models and model elements.
- *Scalability*. Also there is a causal relationship to modularity, therefore we may argue that the existence of building modules is expected to improve the scalability of the i^* framework.

Features slightly improved

- *Expressiveness*. Although not a fundamental issue in this work, it must be remarked that a couple of characteristics of the metamodel enhance expressiveness. First, the capability of linking model nodes to any external concept represented in the *Property* class. *Property* instances may represent ontological concepts (e.g., the concept of business process) or instances of these concepts (e.g., a particular business process). Second, the high-level abstraction classes *Node*, *Intentional Element* and especially *Link* support extension of the language from the syntactic point of view (the most fundamental perspective in this modularity-related work).
- *Traceability*. The *Support* dependum link increments the degree of traceability in i^* models, although of course this is a quite limited contribution. More fundamental traceability mechanisms are still missing.

Features not affected. *Repeatability* and *Domain Applicability* are not related to this work.

Our future work moves along three main directions. First, to enrich the modularity constructs especially by supporting module nesting and a language of module relationships. This second feature may help to record semantic relationships among models (a way to support traceability), e.g. a model for a socio-technical system derived from a pure social model. Second, to settle an experimentation program oriented to gain insights about the advantages and possible obstacles of the proposal, whilst obtaining quantitative feedback about production time, learning curve, etc. Assessment of some decisions taken (e.g., to force all elements in a module to form a connected graph) will be stem from this program. Third, to implement the framework both in *i** edition tools (our HiME, <http://www.essi.upc.edu/~llopez/hime/>, but also try to incorporate it in OME, jUCMNav, REDEPEND, etc.) and in the iStarML interchange format [16]. This is a critical point, since most of the concepts presented here at the modeling level need to be naturally generated by adequate tool support, as transparently as possible for the final user.

References

1. Yu, E.: Modelling Strategic Relationships for Process Reengineering. PhD Dissertation, University of Toronto (1995)
2. Estrada, H., Martínez, A., Pastor, O., Mylopoulos, J.: An Empirical Evaluation of the *i** Framework in a Model-Based Software Generation Environment. In: Dubois, E., Pohl, K. (eds.) CAiSE 2006. LNCS, vol. 4001, pp. 513–527. Springer, Heidelberg (2006)
3. Maiden, N.A.M., Jones, S., Ncube, C., Lockerbie, J.: Using *i** in Requirements Projects: Some Experiences and Lessons Learned. In: Yu, E., Giorgini, P., Maiden, N.A.M., Mylopoulos, J. (eds.) Social Modeling for Requirements Engineering. The MIT Press, Cambridge (2010)
4. Annosi, M.C., De Pascale, A., Gross, D., Yu, E.: Analyzing Software Process Alignment with Organizational Business Strategies using an Agent- and Goal-oriented Analysis Technique - an Experience Report. In: Procs. 3rd *i** International Workshop, CEUR Workshop Proceedings, vol. 322 (2008)
5. Carvallo, J.P., Franch, X.: On the use of *i** for Architecting Hybrid Systems: A Method and an Evaluation Report. In: Procs. 2nd PoEM International Conference. LNBIP, vol. 39 (2009)
6. Alencar, F., Castro, J., Moreira, A., Araújo, J., Silva, C., Ramos, R., Mylopoulos, J.: Integration of Aspects with *i** Models. In: Kolp, M., Henderson-Sellers, B., Mouratidis, H., Garcia, A., Ghose, A.K., Bresciani, P. (eds.) AOIS 2006. LNCS (LNAI), vol. 4898, pp. 183–201. Springer, Heidelberg (2008)
7. Alencar, F., Castro, J., Lucena, M., Santos, E., Silva, C., Araújo, J., Moreira, A.: Towards Modular *i** Models. In: Procs. 25th SAC International Conference – RE Track. ACM, New York (2010)
8. Estrada, H.: A Service-oriented Approach for the *i** Framework. PhD Dissertation, Universidad Politécnica de Valencia (2008)
9. Ayala, C.P., Cares, C., Carvallo, J.P., Grau, G., Haya, M., Salazar, G., Franch, X., Mayol, E., Quer, C.: A Comparative Analysis of *i**-Based Goal-Oriented Modeling Languages. In: Procs. 17th SEKE International Conference, KSI (2005)

10. Cares, C., Franch, X., Mayol, E., Quer, C.: A Reference Model for *i**. In: Yu, E., Giorgini, P., Maiden, N.A.M., Mylopoulos, J. (eds.) *Social Modeling for Requirements Engineering*. The MIT Press, Cambridge (2010)
11. Horkoff, J., Golnaz, E., Abdulhadi, S., Yu, E.: Reflective Analysis of the Syntax and Semantics of the *i** Framework. In: Song, I.-Y., Piattini, M., Chen, Y.-P.P., Hartmann, S., Grandi, F., Trujillo, J., Opdahl, A.L., Ferri, F., Grifoni, P., Caschera, M.C., Rolland, C., Woo, C., Salinesi, C., Zimányi, E., Claramunt, C., Frasinca, F., Houben, G.-J., Thiran, P. (eds.) *ER Workshops 2008*. LNCS, vol. 5232, pp. 249–260. Springer, Heidelberg (2008)
12. Franch, X., Grau, G.: Towards a Catalogue of Patterns for Defining Metrics over *i** Models. In: Bellahsene, Z., Léonard, M. (eds.) *CAiSE 2008*. LNCS, vol. 5074, pp. 197–212. Springer, Heidelberg (2008)
13. Franch, X., Grau, G., Mayol, E., Quer, C., Ayala, P., Cares, C., Haya, M., Navarrete, F., Botella, P.: Systematic Construction of *i** Strategic Dependency Models for Socio-technical Systems. *IJSEKE* 17(1) (2007)
14. Leite, J., Werneck, V., de Pádua Albuquerque Oliveira, A., Cappelli, C., Cerqueira, A.L., de Souza Cunha, H., González-Baixauli, B.: Understanding the Strategic Actor Diagram: an Exercise of Meta Modeling. In: *Procs. 10th WER International Workshop* (2007)
15. Sabetzadeh, M., Easterbrook, S.: View Merging in the Presence of Incompleteness and Inconsistency. *Requirements Engineering Journal* 11(3) (2006)
16. Cares, C., Franch, X., Perini, A., Susi, A.: Towards interoperability of *i** models using iStarML. *Computer Standards & Interfaces* (2010), <http://dx.doi.org/10.1016/j.csi.2010.03.005>

Ahab's Leg: Exploring the Issues of Communicating Semi-formal Requirements to the Final Users

Chiara Leonardi, Luca Sabatucci, Angelo Susi, and Massimo Zancanaro

Fondazione Bruno Kessler – IRST CIT
Via Sommarive, 18 I-38050 Povo, Trento
{cleonardi, sabatucci, susi, zancana}@fbk.eu

Abstract. In this paper, we present our experience in using narrative scenarios as a tool to communicate and validate semi-formal requirements with the stakeholders in a large software project. The process of translating the semi-formal language of Tropos into the narrative form of scenarios is introduced and some unintended implications of this process are discussed. In particular, we define the notion of *Ahab's leg* to describe the necessity to introduce new constraints or features in a description when moving to a different representational language. Starting from the lessons learned with this specific case study, we derive some general implications concerning the issue of requirement translation for validation tasks and we propose some methodological guidelines to address the Ahab's leg dilemma.

Keywords: Goal-Oriented Requirements Engineering, User-Centred Design, Persona, Scenario.

1 Introduction

The focus group with the stakeholders was proceeding in a satisfactory way when suddenly in discussing a scenario one of the nurses commented negatively about the use of sensors located on the doors. She complained that the doors in their facilities are wider than the one depicted in the scenario and that for security reasons they should never be closed. A very lively discussion began about the possibility of locking the doors in a nursing home while the facilitator tried to focus the attention of the group to the functional requirement to be validated: that is, the need to trigger an alarm if a guest leaves a common room.

Situations like this happen very often when a design team wants to validate late requirements with stakeholders: although narrative scenarios are powerful tools to represent and communicate requirements to non-technical people, it may be the case that stakeholders focus their attention on non-central aspects. This paper discusses our experience in using narration as a tool to communicate and validate semi-formal requirements with the stakeholders of a large software project. We encountered the kind of problems above in different forms when we had to decide how to instantiate the formal concepts in a narrative form and which details have to be added to make the story more engaging for the stakeholders. In this paper, we present the process of

translating the semi-formal language of Tropos into the narrative form of scenarios and some unintended implications of this process.

The importance of narration for mediating and discussing requirements with stakeholders has already been discussed in requirements engineering (RE) [1,4,10,13,14,15]. While several studies addressed the issue of integrating scenarios in the requirements elicitation phase, few works specifically considered the challenge of using scenarios—and in particular scenarios represented in a narrative form—to communicate and validate requirements with stakeholders. Efficient communication and iterative validation of requirements with stakeholders is a key challenge for RE. The issue of adequately communicating and negotiating requirements with stakeholders and software engineers still challenges approaches based on formal representation of requirements.

In the context of a large software project, we used narrative visual scenarios to efficiently communicate requirements collected in the field and to provide all participants—both software engineers and professionals—with adequate information to envision innovative and useful services. The semi-formal methodology Tropos was used to filter information, maintain traceability and provide tools for requirements conflict analysis. This paper addresses the advantages and shortcomings of the complementary use of semi-formal descriptions and narrative informal descriptions for the purposes of requirements validation focusing on the challenges posed by the translation between the two. Starting from the lessons learned within a specific case study, we derive some general implications concerning the issue of requirement translation for validation tasks.

In Section 2 we introduce the conceptualization of the Ahab's Leg dilemma, that is, the necessity to add more constraints or features to a description when moving to a different representational language. Section 3 briefly summarizes the methodology and the techniques used for requirement elicitation and management in our project and the issue we encountered. Section 4 introduces a post-analysis of the scenarios used and a proposal for a methodological framework to limit and manage the impact of Ahab's legs in a validation process.

2 The Ahab's Leg Dilemma

In the famous novel *Moby Dick*, the main character, Captain Ahab, has a peg-leg. The author, Herman Melville, told us that it is made from a whale jaw but nothing is said whether it is the left leg or the right one. In 1956, John Houston directed a film adaptation of the book starring Gregory Peck as Captain Ahab. He and the screenwriter, the novelist Ray Bradbury, were forced, because of the constraint of the visual media, to decide that the left leg was the whale bone peg. Although the peg-leg is a fundamental part of the story (you cannot imagine any adaptation of the book for which Ahab does not have a peg-leg), knowing which one has no bearing on it. Yet, when the peg-leg is instantiated, this decision may generate a lot of consequences, some of them might be harmless and some might not¹.

¹ The Ahab's leg example was introduced by Eco [7] in discussing the problem of translation.

The Ahab's leg dilemma consists of the necessity to add more details to the original storyline, because of the different characteristics of the target media (visual vs. textual, in the case of *Moby Dick*) or because of the use of a different communication style, dramatization vs. neutral description, that requires the story to be engaging (as in our example above). Ahab's legs are often unavoidable and they do not necessarily represent a problem unless they bring the viewers (or the stakeholders) to draw unwanted inferences that can contradict other aspects of the story or, as in our example above, divert their attention from the important aspects of the story.

In the case of scenarios derived from requirements, Ahab's legs may be introduced because abstract requirements, summarized as short and clear sentences, are translated into full-fledged narrations. In this process, usually information must be added in order to raise the dramatic tension to the story (the importance of engagement in scenarios is well known in literature [8]) and to instantiate requirements in a concrete spatial-temporal context.

It is worth noting that not every problem encountered when validating scenarios during group discussions can be classified as Ahab's legs. Problems may, for example, be related to the group dynamics rising in focus groups that possibly drift the topic of discussion. Actually, focus groups, different from other methods, require greater attention and the role of moderator is crucial in keeping the group discussion on track [12]. In other cases, when stakeholders complain about a specific aspect of a scenario, it may be the case that the corresponding requirement is wrong or not well understood by the designers. Indeed, spotting these problems in the requirements is precisely the purpose of scenarios as we used them.

The problems raised by Ahab's legs do not correspond to any part of an actual requirement and therefore any discussion about them is a useless waste of time. It is worth noting that Ahab's legs do not necessarily induce shortcomings in validating requirements. In many cases stakeholders are able to avoid discussions deemed irrelevant, especially, as noted above, if the focus groups are effectively moderated by a professional facilitator.

The Ahab's legs can thus be recognized as translation challenges. The possibility of such problems is also recognized by authors in the field of RE (see for example [14]). Interesting suggestions are given by Marasco [11] who underlines the shortcomings but at the same time the necessity to create different views of requirements, highlighting the importance of bridging the gap between text-based and visual requirements representation to improve the quality of requirements in terms of completeness and validity. Still, no systematic analysis has been done to understand and provide concrete solutions to help designers and analysts cope with different views of requirements, in particular between semi-formal representation and narration.

3 Methodology and Techniques

ACube is a large research project funded by the local government of the Autonomous Province of Trento in Italy with the aim of designing a highly technological smart environment to be deployed in nursing homes as a support to medical and assistance staff. An activity of paramount importance was the analysis of the system requirements for what concern cost containment and quality improvement of services in specialized centers for people with severe motor or cognitive impairments. From a

technical point of view, the project envisages a network of sensors distributed in the environment or embedded in users' clothes. This technology should allow monitoring the nursing home guests unobtrusively, that is, without influencing their usual daily life activities. Through advanced automatic reasoning algorithms, the data acquired through the sensor network will be used to promptly recognize emergency situations and to prevent possible dangers or threats for the guests themselves.

The ACube project consortium has a multidisciplinary nature, involving software engineers, sociologists and analysts, and is characterized by the presence of professionals representing end users directly engaged in design activities. A User Centered Design (UCD) approach was implemented to manage the multidisciplinary effort of balancing stakeholders' needs and technical constraints. The integration of UCD methods with the goal-oriented requirements engineering methodology Tropos was meant to assure the validity, completeness and traceability of requirements.

In the following we briefly discuss the two methodologies employed in our study and how they were jointly used during the project.

3.1 Tropos

The Tropos methodology [3,17] relies on a set of concepts, such as actors, goals, plans, resources, and dependencies to formally represent the knowledge about a domain and the system requirements. An actor represents an entity that has strategic goals and intentionality within the system or the organizational setting. An actor is used to model both human stakeholders and software and hardware systems. Goals represent states of affairs an actor wants to achieve. Executing a plan can be a means to realize a goal. Actors may depend on other actors to attain some goals or resources or for having plans executed. Tropos models are visualized through actor and goal diagrams. The former are graphs whose nodes represent actors and arcs are strategic dependencies between pairs of actors. A goal diagram represents an individual actor perspective in terms of its main goals, and their decomposition into sub-goals. Furthermore, plans and resources that provide means for goal achievement are depicted through means-end relationships.

Tropos distinguishes five phases in the software development process: Early Requirements, where the organizational domain is described, Late Requirements, where the system-to-be is introduced in the organization, System Architecture Design, System Design and System Implementation. In the project, we applied the first two phases of the methodology to describe the nursing homes organizational setting and stakeholders' needs and to investigate the technical requirements for the ACube system.

3.2 Personas and Visual Scenarios

Usually, in the practice of requirements engineering, scenarios have been intended mainly as abstract descriptions of systems functionalities. In this project, we took a slightly different stance by employing narrative scenarios and *personas* in the way they are used within the field of Interaction Design (ID) [6]. Narrative scenarios are stories characterized by their brevity and simplicity which represent people acting in a specific context and supported by technologies. Scenarios make concrete the behavior of a service as experienced by specific, though fictional, users. They help design

teams in negotiating a shared representation of the domain and hence a more effective elicitation of requirements. In ID, scenarios are proposed to be used in several phases of the design, from early requirement elicitation to design validation. Actually, as recently stressed by Katasonov et al. [9], a major problem in requirement quality control is the achievement of a satisfactory level of understanding on the requirements by stakeholders especially when they lack technical expertise and do not share the same (formal and abstract) language of analysts and engineers. Due to the assumption that validating requirements is more an issue of efficiently communicating and iteratively negotiating knowledge than a linear process of checking a given corpus of data, in our study we designed visual scenarios as communication tools to allow technical and non-technical partners to symmetrically contribute to requirements validation and refinement. We adopted the specific scenarios approach as developed by Carroll and Rosson [4] and subsequently enriched by Cooper with the notion of *personas* [5], Personas are rich descriptions of archetype users meant to draw attention on users’ goals and motivations. Introducing *personas* in scenarios-based approach provides an anchor against self-referentiality in design and makes scenarios more concrete. Personas (an example of personas is given later in Section 3.3) are created starting from data gathered from actual users interviewed and observed through contextual inquiries. They are usually evaluated with respect to their believability for the stakeholders before creating the actual scenarios. It is worth noting the difference between Tropos actors and personas. While actors are abstract entities and they are not sufficiently concrete to provide understanding of empathy with users, personas are expected to engage the empathy that helps the designers, stakeholders and software engineers to make decisions on both the cognitive and emotional sides.

Table 1. Main characteristic of TROPOS and ID methods

| TROPOS | PERSONAS/SCENARIOS |
|---|---|
| <ul style="list-style-type: none"> ▪ Exhaustive picture of the domain. ▪ Abstract representation of the domain. ▪ Static and invariant picture of the domain. ▪ Do not provide specific tools for finer prioritizing requirements than the reasoning on alternatives and contributions. ▪ Neutral representation that do not engender an emotional response. ▪ Do not provide information about the physical context. ▪ Provides a general representation of invariant dependencies among actors. ▪ Support traceability. | <ul style="list-style-type: none"> ▪ Selection of specific situations that are described in a narrative form (coverage problem). ▪ Concrete representation of the domain. ▪ Dynamic representation involving the spatio-temporal dimension. ▪ Stories provide a support for prioritizing requirements. ▪ “Dramatic” representation that engenders empathy. ▪ Provide details about the physical context in which people act. ▪ Provide details about how interactions occur in a given specific situation. ▪ Do not support traceability. |

3.3 Joint Process: Concurrent Use of Tropos and ID

In ACube, we made joint use of the semi-formal Tropos language and ID methods throughout the whole project lifecycle, in particular from early requirements identification to final validation with stakeholders.

The two approaches explore different dimensions of the domain and of the design space of the system using complementary approaches, tools and languages (see Table 1). In this context, one of the most relevant problems is the perspective and fouds of the two research approaches: ID methods strongly emphasize users and the contexts in which they behave while Tropos focuses on roles and goals, promoting a more abstract level of description that is typical in software engineering techniques. In order to effectively integrate the different methods we developed a framework to allow for the alignment of the approaches, their languages and to facilitate a shared representation of data. Table 2 illustrates the four phases that characterized the process: Early exploration, Problem identification, Envisioning, and Validation. For each phase of the process the table shows the contributions of the two methodologies and artifacts that allowed to maintain an alignment between the two representations.

Table 2. The four phases that characterized the process and artifacts used for communication/integration issues

| Phases/ methods | 1. Early exploration | 2. Problem identification | 3. Envisioning | 4. Validation |
|--|--|---------------------------|--|---|
| UCD methods | Contextual inquiries | Definition of personas | Participative workshops to develop envisioning scenarios | Scenarios discussion with the stakeholders |
| Communication/integration tools | <i>Descriptive table (narrative)</i> | <i>Personas</i> | <i>Narrative technological scenarios and storyboards</i> | <i>Narrative description and storyboards</i> |
| | <i>Actor-Action-Resource-Goal Analysis</i> | <i>Critical Aspects</i> | <i>Positive-negative contributions</i> | <i>Functional and Non-functional Requirements</i> |
| Tropos methods | Domain knowledge, Early actor modeling | Early Requirements Phase | Late Requirement Phase | Requirements Refinement |

Table 3. Some of the personas used in ACUBE Project

| | |
|--|---|
| <p>Sabrina Age: 40</p> <p>Health operator, she assists hosts in daily activities. Her activities: (i) monitor dangerous events, analysis of the kind of event, (ii) raise up an alarm via phone or direct contact with operators, (iii) monitor patient conditions.</p> <p>She likes the human side of her work but she complains to spend too much time in bureaucratic matters. Problems: hard to follow the hosts in absence of an adequate number of health workers.</p> | <p>Piera Age: 90</p> <p>She's lived in a Nursing Home for 6 years. She is not self-sufficient because of health and motion disabilities. She suffers from low level depressing that causes her frequent anxiety and agitation. She wishes for more human relationships with caregivers, nurses and relatives.</p> |
| | <p>Maria Age: 85</p> <p>She's lived in a Nursing Home for 1 year. She suffers from Alzheimer's at middle level with lack of memory and confusion. She tried to leave the institute to go back home once, thus caregivers look at her movements with a special attention.</p> |

1. **Early exploration.** The process started with the investigation of the domain to understand the organizational setting in representative sites and to derive possible needs and a set of possible services that the system may provide to users. In the ACube project, contextual interviews [2] were performed in 4 different nursing homes and involved about 40 health professionals including health workers, nurses, medical staff and managers. Information retrieved via contextual inquiry are represented via Tropos actor-goal early requirement models.

2. **Problem setting.** The analysis of critical aspects has been developed to highlight the main problems that professionals of nursing homes experience in their job. We defined a set of *personas* (see Table 3) and generated narrative descriptions of these personas deploying their daily working activities in a specific context, using resources to reach their goals. The aim was to represent criticalities that may be addressed through a technological intervention. Here, Tropos models support the representation of critical aspects via the characteristics of actors and goals.

3. **Envisioning.** A participative workshop has been organized to analyze how technology could positively intervene in activities thus supporting the achievement of goals and resolving critical issues identified in the problem identification phase. It is worth noting, that in this phase our main goal was investigating high level requirements and not producing design ideas. About 10 participants attended the workshop including the ID team and representatives of stakeholders and technologists. The heterogeneity of the group was meant to guarantee the generation of creative but feasible ideas, to provide concrete solutions to problems identified by nursing home professionals as well as to provide solutions that could meet engineers' expectations and their research interests. Outcomes were pursued at multiple levels: to expand the designer's perspective and to see the problems from different points of view, to figure out how their ideas can work in a real context, to identify design criticalities and open issues, to generate requirements of the system-to-be. The workshop ended with the definition of 5 different macro-services the ACube system might provide.

As a consequence of the envisioning focus group, and the introduction of the system into the organization, the Tropos process moved from the early requirement phase to the late requirement phase. Figure 1 shows an excerpt of the Tropos model, describing a small part of the goals and the activities of the SeniorOSS actor (a caregiver in the nursing home). In particular the actor Senior OSS has the goal to [*prevent dangerous behavior in patients*] that can be AND decomposed in [*monitor patients in her visual area*] and [*coordinate interventions in the nursing home area*].

This latter goal is delegated to the ACube System actor via the goal delegations [*identify a guest dismissing the group*] and [*receive alerts of relevant events*]. These goals are two requirements to be satisfied by the system that must instantiate them (means-ends relationships) via the plans [*monitor patients*] and [*send alarms*] respectively.

4. **Validation.** Two focus groups were organized with stakeholders and the technical staff for validating the list of requirements produced in the previous phase. Due the importance of this step for the topic of the paper, this is part is discussed in detail in the following subsection.

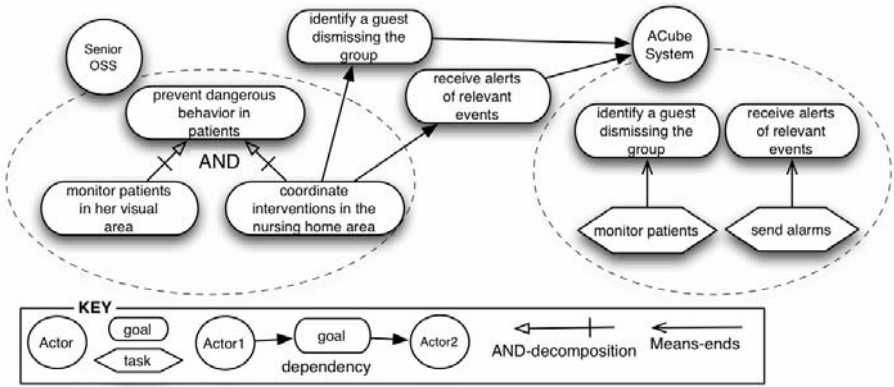


Fig. 1. An excerpt of the Tropos model for the nursing home

3.4 The Validation Phase

After the preparation of the Tropos late requirement diagrams, and the corresponding list of requirements, we started the validation phase. Simple visual scenarios were designed to make the list of requirements more understandable by partners. To generate scenarios we imagined how the system could support personas to cope with problematic situations during their daily work. Macro-services emerged in the envisioning phase have been instantiated into concrete – but non exhaustive – representations of the system functionalities. Eventually, 5 visual scenarios were generated, each addressing a problematic situation identified in nursing homes accompanied by one of the possible technological solutions.

A first focus group was held with the representatives of the 10 research groups involved in ACube project, 27 people attended the meeting. The second focus group was organized with the stakeholders, 3 managers of nursing homes previously involved in the early exploration phase attended the meeting. The goal of these meetings was to assess of the validity, acceptability and feasibility of requirements and to envision alternatives not considered in the scenarios.

The structure of the two meetings was the following: first, a general presentation was given to introduce the goals of the meeting and to discuss general results collected in the fieldwork. Then, for each of the five scenarios generated we introduced: the context of the scenario – organization context and personas acting in those contexts, the rationale for the scenario, which is the criticality we wanted to address with that specific scenario. Subsequently, the scenarios were represented in a visual form through storyboards (see Figure 2). Finally, criticalities the scenarios could rise – in terms of technological feasibility and acceptability for end-users – and the underlying abstract requirements that the scenarios instantiate were presented to trigger the discussion. For each scenario, 20 minutes of discussion followed. A moderator was in charge of driving the discussion on the specific dimensions we wanted to assess.

The workshop with technological partners was focused on technical feasibility and research interest, and on the envisioning of original solutions to the critical situations identified. Acceptability and usefulness were instead the *pivots* of the workshop with

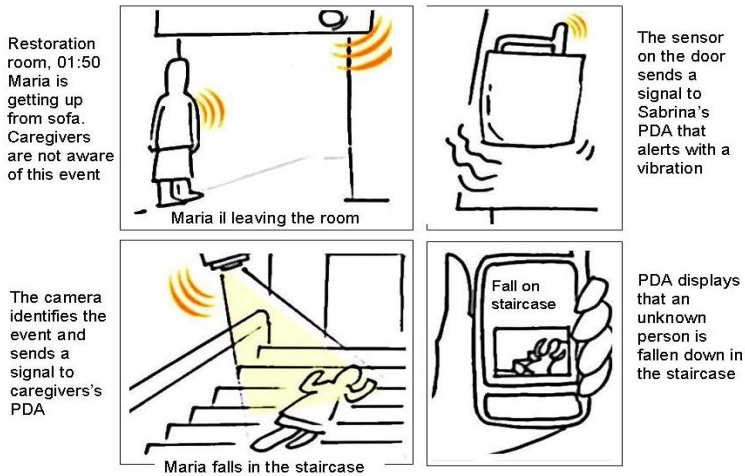


Fig. 2. A scenario extracted from the Tropos model of the domain

end-users representatives. Expected outcomes of the two focus groups were the emergence of design criticalities, the resolution of open problems and the identification of new propositions and ideas, in order to collect additional elements to elaborate an organic description of the technological architecture. The role of the moderator was also to focus participants' attention on specific aspects of the scenarios (those directly related to requirements) and to cut discussions concerning non relevant aspects. The output of this phase was the agreement on certain requirements and the refinement of Tropos late requirement diagrams.

Besides the general positive output and the satisfaction of the partners – principally satisfied of the rich discussion emerged around scenarios – several secondary (collateral) issues emerged. Several times the discussion of participants focused on aspects non-relevant for technological purposes. Beside traditional shortcomings usually found when conducting focus groups [12], we identified other impasses pertaining to the kind of information communicated through visual scenarios, leading participants to lose repeatedly the focus of attention. As discussed above, we defined as Ahab's Legs the translation and communication shortcomings rising from the necessity to translate information from one media to another.

4 Re-thinking the Approach

The issues we experienced during the focus groups with technological partners and with stakeholders were mainly due to Ahab's legs (AL). This section reports the analysis we conducted, after the validation phase, in comparing the knowledge expressed in the requirements documents with the knowledge mediated by the visual scenarios.

4.1 Ahab's Leg Classification

In order to identify the ALs in our 5 scenarios we compared each frame of each scenario with the corresponding requirement. For example, Table 4 illustrates the case of the requirement [*The system identifies when a guest is dismissing the group*]. This requirement has been visually represented in the scenario for the emergency monitoring and prevention of falls. This scenario shows a guest of the institute that wears a sensor that sends a signal. This is captured by a receiver that is placed near the door of the room, where the guest is passing through.

This simple piece of scenario includes four ALs: 1) the definition of the time and place in which the scene is set: a wide common room with one door; 2) the couple wearing sensor/receiver is an AL cause it is not yet defined by design how to track guest movements; 3) the receiver placed on the door is another AL, because it refers to a decision to set the device in a precise position of the environment; and 4) the use of arcs going out from the sensor to the receiver is a graphical means for communicating the presence of an interaction in a symbolic way.

With this procedure, we collected 34 ALs from the 28 frames of the 5 scenarios. An analysis based on similarities of occurrences results in 6 different categories of AL. Three of these categories relate to the cause of the difference in translation: (i) the need to introduce a design feature to visualize a part of the system, (ii) the need to make concrete an abstract representation and (iii) the use of symbols that may be misleading. The other three categories related to the impact that the AL has on the target language: (i) the resource used, (ii) the context in which the scene takes place and (iii) the working practice that is represented. These categories are not exclusive and an AL can be classified as belonging to one or more categories.

An AL is generated by an *early design* when the scenario is constrained to show some design solutions that have not been explicitly chosen but that have been represented in order to elicit underlying problems and suggest concrete solutions. For instance the use of a wearable sensor that sends signals does not come out from the requirements list, but the design team decided to make concrete the presence of the system in such a way. Nevertheless, the presented technical solution is only a possible alternative among others; other solutions could be designed, for instance, the use of cameras spread in the environment was going to be evaluated too. Starting from abstract requirements (such as “The system alerts caregivers of relevant events”), one of the possible design solutions has been visualized (“a PDA in the caregiver’s pocket vibrates to signal the event”) to communicate to stakeholders one of the possible design solution that could meet that requirement.

An AL can be generated by the *level of details* due to the specific media used for representing the scene. The typical example is the one that inspired the name of Ahab’s leg: the cinematographic version of the novel requires a decision about which is the peg–leg. An example from the project concerns the decision to set the scene in a specific kind of environment, a common room, close to a door.

This kind of AL can be minimized by carefully translating abstract information to a concrete scene. While a certain contextualization is necessary to create a credible story, the designer should however pay attention to convey only the necessary information, and to maintain abstract those contents that could generate a discussion on non relevant details. In our case it was important to communicate the information that

the scene was taking place in a common room but we decided to keep all the other information on the environment implicit in order to let stakeholder focus only on the relevant event, such as common room with several patients and few health professional available.

An AL is generated by the *use of symbols*, typical of comics, that communicates something abstract in a scene, as well as an interaction, a mood, the act of thinking and so on. The twofold risk is to use a symbol whose meaning is not commonly recognized by people, or to communicate in a concrete way something that would be better to maintain abstract. An example of AL in this category is the use of arcs for representing a wireless communication or signal connecting the sensor with the receiver. This graphical expedient is used to show an interaction between two devices, but the risk is stakeholder focus on the direction of the communication (who is the transmitter, who is the receiver). This problem pertains to a more general issue well known within the semiotic research area, that is, inter-semiotic translation, occurring every time a linguistic sign is translated by means of non-linguistic signs (visual or audio texts).

Table 4. Requirements, scenarios and Ahab's legs in our example

| Requirement | Scenario Frame | Ahab's Leg |
|---|--|---|
| 1. The system identifies when a guest is dismissing the group | <i>3.00 pm, common room.</i> The scene shows a guest of the institute that <i>wears a sensor that sends a signal.</i> This is captured by a <i>receiver that is placed near the door</i> of the room, where the guest is passing through | AL1: time and place AL2: resources - wearable sensor and receiver AL3: receiver on the door |
| 2. The system alerts caregivers of relevant events | The scene shows (and describes) a <i>PDA in the caregiver's pocket that vibrates</i> | AL4: resource - PDA AL5: caregivers have a device in their pocket AL6: vibration for alerting the caregiver |

Among **impact** we identify: (i) resource, (ii) context and (iii) working practices. An AL may influence a **resource** (typically a technological device) that the system will introduce in the environment, or already existing in the domain. An example is the introduction of a sensor in the environment that tracks guest movements. An AL may influence the **context** represented in a scenario, adding details about the time (for instance by specifying when the scene is set: 'at 3:00 pm'), about the space (for instance by specifying where the scene is set: 'in the common room'), about a condition or an event that is occurring (for instance 'the guest is moving through the door') or a quantity (for instance specifying how many guest and caregivers are present). An AL may also add details about a **working practice** or a methodology that caregivers will adopt as a consequence of the system-to-be. The scene provides details (for instance about decisions that are taken or activities that are executed) just because the dramatization of the story needs a plot in which personas act for solving emergencies.

This classification has been developed starting from a specific case and it has not yet been completely investigated. Therefore, we cannot at this stage claim that it is of general purpose. However this classification is coherent with the categories identified by Eco [7]. Table 5 reports a subset of the ALs we identified in this project.

This analysis has been used to provide a rationale to each AL found in the 5 scenarios and to decide whether it could have been removed or not. The scenario-authoring activity should be iterated by considering the evidence of each AL and considering the relevance of the corresponding detail in the scene. If the detail can be removed without losing important data that designers want to communicate, the scenario should be redefined. For example, in the case of the AL3 [*the receiver placed on the door*], this detail could have easily been removed reducing the risk to focus stakeholders’ attention to technological details that were not yet discussed.

Table 5. Classification of the Ahab’s Legs in our example

| | Early Design | Level of Details | Use of Symbols | Resource | Context | Working Practice |
|---|--------------|------------------|----------------|----------|---------|------------------|
| AL1: time and place | | √ | | | √ | |
| AL2: wearable sensor and receiver | √ | | √ | √ | | |
| AL3: receiver on the door | | √ | | | √ | |
| AL4: resource - PDA | √ | | √ | | | |
| AL5: caregivers have a device in their pocket | | √ | | | | √ |
| AL6: vibration for alerting the caregiver | √ | | √ | | | |

The categorization of the AL dilemma brought us to propose some guidelines to systematically approach the elimination of some irrelevant details from scenarios. In the cases in which removing an AL is not possible or too complicated, it is very important to frame the scenario (for example with an introductory description) in such a way that the discussion from the AL is averted as much as possible.

4.2 Toward a Methodology for Translating Requirements into Scenarios

When we prepared our narrative scenarios we were aware of the possible communication problems that may occur during the validation phase, thus we spent a lot of effort in preparing the meeting, to direct the conversation in the desired direction. Despite this preliminary work, we have been unable to avoid having stakeholders sometimes concentrate on some secondary aspects of the narration (for example the discussion about the doors of the room in the introduction). The post-analysis conducted on the scenario before the validation experience revealed a bigger number of ALs than we recognized at the beginning. This suggested to introduce a scenario-refinement activity in our analysis process in order to consider whether the use of each AL was really beneficial to the scenario (because it supported a greater level of engagement or made a requirement clearly visible, for example) or was just a distracting narrative element.

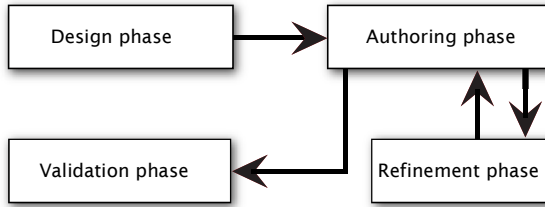


Fig. 3. A methodology for translating requirements into scenarios

The methodology for moving from requirements to their validation with stakeholder can be summarized as (see Figure 3): a *design phase* in which requirements are draft, a *scenario authoring phase* in which these requirements are represented in a narrative (and often visual) format. Finally, the *validation meeting* in which scenarios are presented and discussed with stakeholders. We propose to introduce a *refinement phase* after the *scenario authoring* and before the validation meeting. For each scenario an *AL detection activity* is executed, that may lead to re-elaborate the scenario representation in order to reduce the number of irrelevant details. The steps can be summarized as follows. (i) Sort requirements by the scenario in which they are represented; (ii) divide the scenario in frames, each is a self-explicatory part of the story; (iii) compare the requirement with the frame and record each detail that is added for enriching the story as an AL; (iv) use the classification schema for evaluating the importance of the presence of the AL to the aim of telling the story; (v) proceed to the elimination of the surplus details from the story or to their modification. It is worth noting that scenarios and not requirements are subject to changes during the refinement phase.

As we already mentioned removing ALs from a narrative description is by no means the only way to manage ALs. For example, when a fieldwork analysis was conducted before the requirement phase, the illustrative material (photos, sketches, interviews, etc.) may be used to add details to the narrative scenarios that result “natural” and therefore less distractive for the stakeholders. Finally, the remaining ALs may be the properly framed by the facilitator to reduce the risk to discuss them for too long.

5 Conclusion

In this paper, we discussed some issues that emerged in a large research project when we tried to validate our requirements with the stakeholders by using narrative scenarios instead of the textual version of the requirements themselves. This practice is sometimes used in RE, nevertheless several authors acknowledged the problem of running into misunderstanding when minor details are added to the narration to enrich it and make the presentation concrete and easy to understand. We identified this translation challenge as the Ahab’s leg problem.

Although only a few ALs creates problems during validation with stakeholders, several more were identified in a post-analysis of the scenarios generated. Six non-exclusive categories of ALs have been identified by the analysis. Consequently we

propose a methodology to manage the insurgence of ALs that consists in a scenario-refinement phase in which every AL is checked and, when possible, is eliminated or modified in order to be understandable by stakeholder without conveying undesired meanings. It is worth noting that to some extent ALs may also be considered beneficial. In a ID perspective, showing a hypothetical vision of the system-to-be is a way to open up the discussion on design in order to redefine the problem with stakeholders. In this respect, ALs may help to foster the discussion on some not-central but still very relevant dimensions of the problem. This approach is recommended by a modern approach to ID [19]. Yet, even in these cases, ALs should still to be carefully managed by clearly communicating the goals and motivations behind their introduction.

Finally, the problem of generalization is still an open question on our work. By comparing them with categories identified in semiotics studies, we assume that they are general enough to be applied in other contexts but more data is needed for assessing the classification. Another aspect that still needs attention is how to relate the AL categories identified in the post analysis to systematic procedures for controlling (limiting or encouraging) the occurrence of ALs in the scenario specification.

Acknowledgments

The research was funded by the Autonomous Province of Trento, project ACube (*Grandi Progetti 2006*).

References

1. Aoyama, M.: Persona-and-Scenario Based Requirements Engineering for Software Embedded in Digital Consumer Products. In: Proc. of 13th IEEE International Requirements Engineering Conference (RE 2005), pp. 85–94 (2005)
2. Beyer, H., Holtzblatt, K.: Contextual Design: Defining Customer-Centered Systems. Morgan Kaufmann, San Francisco (1998)
3. Bresciani, P., Giorgini, P., Giunchiglia, F., Mylopoulos, J., Perini, A.: Tropos: An Agent-Oriented Software Development Methodology. *Autonomous Agents and Multi-Agent Systems* 8(3), 203–236 (2004)
4. Carroll, J.M., Rosson, M.B.: Getting around the task-artifact cycle: How to make claims and design by scenario. *ACM Transaction on Information Systems* 10, 181–212 (1992)
5. Cooper, A.: *The Inmates are Running the Asylum*. SAMS Publishing, USA (1991)
6. Cooper, A., Reimann, R., Cronin, D.: *About Face 3: The Essential of Interaction Design*. Wiley Publishing, Chichester (2007)
7. Eco, U.: *Kant and the platypus: essays on language and cognition*. Harvest Books (2000)
8. Grudin, J., Pruitt, J.: Personas, participatory design and product development: an infrastructure for engagement. In: *Proceedings of Participatory Design Conference 2002* (June 2002)
9. Katasonov, A., Sakkinen, M.: Requirements Quality Control: a Unifying Framework. *Requirements Engineering Journal* 11(1) (2006)
10. Liu, L., Yu, E.: Designing Information Systems in Social Context: A Goal and Scenario Modelling Approach *Information Systems* 29(2), 187–203 (April 2004)

11. Marasco, J.: The requirements translation challenge, http://articles.techrepublic.com.com/5100-10878_11-6128696.html
12. Morgan, D.: Focus groups as qualitative research. Sage Publications, Thousand Oaks (1997)
13. Pohl, K., Haumer, P.: Modelling Contextual Information about *Scenarios*. In: Proceedings of the Third International Workshop on Requirements Engineering: Foundations of Software Quality REFSQ 1997 (1997)
14. Potts, C., Takahashi, K., Antòn, A.: Inquiry-Based Requirements Analysis. IEEE Software archive 11(2), 21–32 (1994)
15. Rolland, C., Achour, C.B., Cauvet, C., Ralyté, J., Sutcliffe, A., Maiden, N., Jarke, M., Haumer, P., Pohl, K., Dubois, E., Heymans, P.: A proposal for a scenario classification framework. *Requir. Eng.* 3(1), 23–47 (1998)
16. Seyff, N., Maiden, N., Karlsen, K., Lockerbie, J., Grünbacher, P., Graf, F., Ncube, C.: Exploring how to use scenarios to discover requirements. *Requir. Eng.* 14(2), 91–111 (2009)
17. Susi, A., Perini, A., Giorgini, P., Mylopoulos, J.: The Tropos Metamodel and its Use. *Informatica* 29(4), 401–408 (2005)
18. Uchitel, S., Chatley, R., Kramer, J., Magee, J.: Goal and scenario validation: a fluent combination. *Requir. Eng.* 11, 123–137 (2006)
19. Wolf, T.V., Rode, J.A., Sussman, J., Kellogg, W.A.: Dispelling “design” as the black art of CHI. In: Proceedings of CHI 2006 (2006)

The Brave New World of Design Requirements: Four Key Principles

Matthias Jarke¹, Pericles Loucopoulos², Kalle Lyytinen³,
John Mylopoulos⁴, and William Robinson⁵

¹ RWTH Aachen University, Germany

² University of Loughborough, U.K.

³ Case Western Reserve University, USA

⁴ University of Toronto, Canada

⁵ Georgia State University, USA

Abstract. Despite its undoubted success, Requirements Engineering (RE) needs a better alignment between its research focus and its grounding in practical needs as these needs have changed significantly recently. We explore changes in the environment, targets, and the process of requirements engineering (RE) that influence the nature of fundamental RE questions. Based on these explorations we propose four key principles that underlie current requirements processes: (1) intertwining of requirements with implementation and organizational contexts, (2) dynamic evolution of requirements, (3) architectures as a critical stabilizing force, and (4) high levels of design complexity. We make recommendations to refocus RE research agenda as to meet new challenges based on the review and analysis of these four key themes. We note several managerial and practical implications.

1 Introduction

The genesis of Requirements Engineering (RE) research around 30 years ago was motivated by practitioner who noticed the urgent need for disciplined RE in large software projects [1, 2]. Much of RE research since then has focused on artifacts that help capture, share, represent, analyze, negotiate, and prioritize requirements as a basis for design decisions and interventions (for recent reviews see e.g. [21, 22, 23]). This is evidenced by the volume and impact of a plethora of requirements topics papers published in top the level software engineering and computing conferences and journals (for a survey see [4, 23]). Due to its practical origins it is not surprising that some of its findings, like the use of business and system modeling (ERD, use cases), risk driven methodologies, structured requirements documents, and simple requirements tracing, have found their way into practice[3].

Yet, the environment in which RE is practiced has changed dramatically. Partly, this is due to increases in computing speed, lowering of computing cost, and advances in functionality, which has made software common in all walks-of-life. Partly, this is due to the changes in technological, task, and organizational environments where

software is either produced or deployed. The field's focus and scope has shifted from engineering of individual systems and components towards the generation and adaptation of software intensive ecosystems. Accordingly, a term *design requirements* rather than *software requirements* is needed as an inclusive term to denote all common sets of requirements issues within these ecosystems that need to be addressed at the crossroads of business development, software engineering, and industrial design. This shift has created a strong need to re-think and re-align RE practices to meet the new challenges. Both academia and industry need to understand more deeply issues that underlie current RE and address associated challenges. We posit that answers cannot come just from doing more of the same—i.e., traditional RE research focusing on notations and tools alone. The research scope of RE has to become more interdisciplinary, and it needs to carefully evaluate some of its critical assumptions. This essay aims to identify some of these challenges based on a detailed field and content analysis of extensive expert discussions and feedback on current RE practices [4]. Based on these explorations we propose four principles that underlie future requirements processes and influence their successful resolution. Finally, new research challenges and practical implications are identified.

2 The Changing Nature of Requirements

Current RE landscape is marked by new challenges and opportunities [3]. Its environment, target technologies, processes, and fundamental problems have undergone a tectonic shift. The environment of RE now involves elements that were not there 20 or 30 years ago [5]. First, the economics of RE has changed. Large systems like ERP systems need more rigorous ROI, but at the same time horizons for ROI have reduced to 18-20 months thanks to massive reuse and COTS deployment. Second, there is practically no green-field software development; RE acts more like the Roman god of gates—Janus, with one face, looking at new business and technological challenges and opportunities and another face gazing at existing (technological, organizational, social and political) environments and resource sets. Third, the scaling towards software intensive ecosystems results in exceedingly complex and non-linear dynamic dependencies between system components and their natural, technical and social environment—“green IT” being just one of the latest buzzword that characterize this trend. Fourth, speed and agility, time to market, low-cost iterative, or even end-user development have become critical factors leading to search for new design trade-offs between efficiency, openness, and flexibility. This has also increased outsourcing and off-shoring, which requires disciplined evolution and management of explicit specifications as a basis for delegation and framing design problems. Fifth, RE now cuts across industrial design (e.g., pervasive applications), media design (e.g., e-commerce and media applications), interaction design (e.g., new modalities of interaction in mobile computing, telematics etc.), and business process design (e.g., open business platforms), and regulatory and juridical issues (e.g. management and control of multiple licenses in software platforms). Overall, design requirements need to capture and coordinate increasingly diverging and dynamic needs of users and other stakeholders during the evolution of a product, a service, or a platform.

Table 1. Summary of Critical Design Requirements Issues (adopted from [4])

| | Critical Requirements Issues | Brief description |
|---------------------|-----------------------------------|--|
| Target platform | Business process focus | Requirements focus simultaneously on the business process, and requirements for technological artifact driven by that business process. |
| | Systems transparency | Requirements involve the demand for a seamless user experience across applications. |
| | Integration focus | Requirements focus on integrating applications rather than development of new ones (i.e., less green-field development). |
| | Packaged software | Purchase of commercial off-the-shelf (COTS) software and components rather than internal development. This has led to market driven vendor-led requirements and knowledge brokering. |
| Development process | Distributed requirements | In addition to increasingly diverse stakeholders, requirements processes are distributed across multiple organizations, groups and social worlds globally. |
| | Centrality of architecture | Architectural considerations and associated evolutionary paths take a central role and drive business, product and application requirements. |
| | Layers of requirements | Requirements need to be iteratively developed across multiple levels of abstraction, design focus, or temporal horizon. |
| | Interdependent Complexity | While some forms of design complexity have been reduced (loosely coupled components), the overall interaction complexity of the design ecology has risen enormously. |
| | Fluidity of design | Requirements process must accommodate the need for continued evolution of the artifact and the solution after initial implementation. |

What are the critical issues that emerge during RE in this brave new world? Table 1 presents nine critical issues that were solicited in a field study in Fortune 500 companies [4]. These are divided into the changing nature of the object of RE (target platform), and the process of RE (development process). Overall, these issues resonate well with the debate Simon engages in his design classic, *The Sciences of the Artificial* [6]. On the one hand, software designs resemble increasingly continuous and dynamic searches for satisficing solutions—not an optimized and fixed solution at one time point conforming to a fixed set of requirements. On the other hand, they go beyond Simon’s original model in that they emphasize sense-making in shifting and complex environments [7] and associated problem framing over focused problem solving in a bounded context. To wit, these changes in the environment and the object and process of RE are changing the three classic RE problems as follows:

First, the *design requirements problem* already pointed out in [1, 2] can be stated as follows: *What is the emergent behavior and dynamics of the software artifact and its environment in their evolutionary trajectory?* Now users, designers and other stakeholders need to ask: will the system continue to satisfy emergent goals, and what

those goals could be expected to be during the artifact's life-time; in contrast to the older problem: What are the goals of the system and what is expected to do

Second, the *specification problem* can be stated as follows: *How can designers anticipate and represent the emergent behaviors of the system and its components and how does the resulting system behavior conform and relate to the emerging environments and the notations used to represent and predict it?* Accordingly, now designers need to ask how they can represent, communicate and analyzed increasingly complex and dynamics systems and their emergent requirements in contrast to the older problem: how to represent the system components, their relationships and behaviors and guarantee that they meet functional and non-functional requirements?

Third, the *predictability problem of designs* can be stated as follows: *How does the artifact and its behavior change the environment as to make our predictions of the system behaviors faithful?* In other words, now designers need to attend more to the dynamic composition of the system and environment and do they together differ from the environment alone, and can he/she accordingly predict faithfully the impact of the system on the environment, and vice versa? This is a different problem from those faced earlier where the system was assumed to *not* affect the environment, or the environment the system, but in some rare cases [14].

3 Four Requirements Principles

Past RE research has been informed by a few key principles such as those of: separation of what, why and how [1], information hiding (see e.g. [2], and the principle of abstraction (see e.g. [22]). These principles reduced design complexity by localizing design decisions. They also helped reduce their interference and analyze and predict system behaviors and structure from specific viewpoints [21, 22].

But, what are the key principles that will underlie successful design and RE in the brave new world we face? What principles will help us address the design requirements problem, the specification problem, and the predictability problem in the new context? We propose next four principles that were gleaned from our analysis of expert opinions and related discussions, and a review of the literature dealing with design dynamism and complexity. These four principles are:

Intertwine Requirements and Contexts: The necessity to *intertwine* design and requirements with design and implementation across multiple dimensions.

Rationale: This principle observes the design requirements problem and the new demands for understanding evolutionary trajectories.

Evolve Designs and Ecologies: The necessity to view design and design processes as *evolving* elements in an *ecology*.

Rationale: This principle observes the new design requirements and specification problem and increasing demands to analyze evolutionary principles of a large set of heterogeneous elements comprising the requirements space.

Manage through architectures: *Architectures* have a critical role of as enablers and constraints in the constant creation and shaping of design ecologies.

Rationale: This principle recognizes the specification problem and the shifting focus towards ecologies where increased emphasis must be placed on antecedent factors that affect the organization and evolution of the ecology.

Recognize complexity: The heightened *interaction complexity* of requirements processes demands new ways to approach design problems and manage requirements.

Rationale: This principle recognizes the predictability problems and that the new interaction complexity requires designers to heed on the external relationships of the software and their evolution as reflected in the requirements.

We will next describe each principle in more detail in terms of the content of the principle, related research questions, and emerging research.

3.1 Intertwine Requirements and Contexts

The debate about the role of requirements is as old as the field itself. Whilst a rough consensus has been reached that requirements are a pre-requisite for downstream development, there is a great deal of controversy on how ‘problem’ and ‘solution’ spaces interplay during the evolution. One school of thought regards the influence of implementation on requirements as being harmful [8]. They argue that understanding the system’s context, such as its organizational and social factors and goals can provide a sufficient set of functional and non-functional requirements, which can then be mapped onto appropriate implementation models. This school regards requirements as the “downward” bridge between the ‘subject’ and ‘system’ worlds by assuming that there exists a high degree of stability on business, organizational, and community goals. An opposing view stresses the need for revisiting requirements as implementation progresses and emphasizes the dynamics and intertwining of these activities [9].

The review of existing practice [4] shows that implementation and its requirements specification are now necessarily intertwined. In fact, many requirements emerge from existing solution spaces. Accordingly, the concept needs to be extended to the whole system context. In addition, the salient factors shaping RE seem to be innovation and effective differentiation. The interplay between the two worlds has thus become more intricate, complex, dynamic, and generative. In these innovation-driven settings, requirements become part of both the business solution and the system solution, and they constantly bridge new solutions to organizational and societal problems. The evolving designs need to reduce the distance between a problem and a solution through novel and dynamic thinking, acting, and innovating. In such a design-thinking culture, design requirements become increasingly central and need to be understood as part of a multi-system, socio-technical ecology, which drives organizational innovation. Therefore, software requirements need to be dynamically situated between these spaces as they intertwine organizational and implementation considerations, providing leverage to influence both.

Due to this constant intertwining some systems and ecologies may be reaching a practical *world-model limit*. While prior design efforts could rely on an adequate, stable, world-model as the basis for specifying nearly stable software designs, now, software must be agile—rapidly evolving to meet changing needs. The level of stability of the world-assumptions is less limited in context-aware, customer-focused applications. The unavoidable intertwining between requirements and contexts will make

designers constantly seek correspondence between the models in software and its world context. Only software embedding an adequate, flexible, and evolvable world-model is likely to survive. The idea of evolutionary software and variability selection aims to partially meet this need. But, little attention has been given to the challenge of formulating evolutionary world models that form the basis for the necessarily simplifying, but evolving model assumptions in the software. Software developers must thus monitor and evolve their understandings of the world, and sustain an adequate correspondence between the world and the modeled world.

Overall, RE processes face a new kind of uncertainty that goes beyond traditional RE uncertainty characterized by: (1) requirements identity (knowing requirements), (2) requirements volatility, and (3) requirements complexity [10]. In addition, designers need to devote their efforts: (1) *requirements fidelity uncertainty*, which denotes the uncertainty about the level of intertwining between the world and the software model. Examples of techniques that help mitigate fidelity uncertainty are exception and event-based analysis; software tailoring and user-based development, and case-based learning; and (2) *requirement monitoring uncertainty*, which denotes uncertainty of the level and mode of observation, and analysis necessary to assess the world, the model, the requirements, and their alignment. Examples of monitoring include ethnographic methods, business activity monitoring (BAM), and software instrumentation. These two new levels of uncertainty highlight the need for increased run-time monitoring to maintain the fidelity of the world-model intertwining with requirements.

3.2 Evolve Designs and Ecologies

Meeting stakeholder needs is fundamental to requirements activity. When requirements increasingly intertwine with organizational and implementation concerns, they will constantly and non-linearly evolve as part of the “ecology”. Traditional causes of software evolution have been classified into: (1) the software, (2) the documentation, (3) the properties of the software, and (4) customer-experienced functionality[11]. Evolution has been studied mainly as a software design problem and it has been addressed by improving methodological support (e.g., how can development activities most effectively incorporate evolution?) and its management (e.g., how one can one record and trace software releases or link the code to changing domain knowledge?). Now, the reality of an ever incomplete and evolving design needs to be addressed. We need to ask: what are the principles that guide developers evolve ‘incomplete designs’ so that they remain functionally adequate, but offer flexibility? What are appropriate co-evolutionary design methods to achieve these goals? How does one determine the impact of co-evolutionary design change?

Activities in open source development, such as inter-project merging and the creation of new software artifacts, for example, compound the need for new frameworks to cope with requirements evolution. Another example is agile methods and scenario based modeling which offer a means to better cope with the fast paced evolution of requirements ecologies [12]. Likewise research into co-evolution and co-design [13] has addressed drivers and interaction laws that deal with the intertwining of contexts and requirements. Yet, such studies are in early stages, and agile methods only deal with micro-level evolution of local tasks, but ignore their recursive nature as the

change propagates across the higher levels of architectures and systems. Here we clearly need longitudinal studies of the dynamics of software ecologies and how different causes ranging from technological, user level learning, organizational policies, market based, and regulatory changes intertwine and generate new evolutionary paths.

3.3 Managing through Architectures

Architecture is concerned with blueprints that connect high level organizational, business, or implementation considerations with a long-term evolutionary perspective. Organizations now increasingly conform to business or information architectures that provide stability, scale and change to their data, business rules and decision models. Designers have, for some time relied on implementation architectures while evolving their designs. In its variety of forms an architecture provides the stepping-stone necessary to understand and evolve any system functionality across different domains. Through release planning, requirements play a central role in systems evolution, where architectures provide “nearly” fixed points of reference to moderate, constrain and enable evolution. Such dependence on architecture is inherent in Lehman’s law that “the incremental growth (growth rate trend) of evolutionary software systems is constrained by the need to maintain familiarity” [14]. Architectural dependencies arise also in approaches like IKIWISI (I’ll Know It When I See It), and COTS (Commercial off-the-shelf) based software deployment [15].

Though RE research in the past has paid significant attention to ‘software architectures’, it offers limited insight to the role of architectures in the new RE terrain. Many of the past studies focus on organizing sets of design elements and their components in the context of a single system. Accordingly, they approach architectural design akin to generating a blueprint for a single house. In the context of dynamic software ecologies, such an analogue fails. In the brave new world, managing through architectures is about generating and evaluating multiple and multifaceted plans similar to urban planning. Like urban planning blueprints, architectural models provide the key artifact for coordinating components, functionalities, and their evolution. As in urban planning, the architectures in RE may have alternative and overlaying *variation points* that influence the evolution of the software ecology. As in urban planning, architectures embody specific business models or design visions, and come in different forms in different design contexts. By doing so, they integrate the needs of multiple stakeholder groups with varying roles. Finally, like urban plans they involve the same level of complexity and interdependencies. Therefore, in the new RE, we see an increasing need to understand the variation between types of architectural models needed and how they relate to specific families of systems and their ecologies.

Recent attempts to deal with architectural considerations include the transfer of industrial concepts, such as product lines to the software field [16]. They help manage and cope with continuous and rapid change in software by defining the scale, scope, and direction of its variance and selection. Other architectural models such as business architectures [3] help stakeholders to envision the impact of proposed changes on business by providing contextual information that allows for selecting variation points across multiple stakeholders. Yet, many research challenges remain in taking the advantage of the idea of architecture: How do architectures influence the evolution of requirements and their identification? What is the nature of requirements discovery

and elicitation under varying architectural principles? Is it possible or even desirable to construct a single common ontology of business, information and technology architectures? How to relate different architectural presentations and reason around them? How can architectures help in flexible composition of systems and ecologies?

3.4 Recognize and Mitigate against Design Complexity

Complexity is borne out of the existence of multiple uncertain futures that relate to software and their evolving ecologies [17]. A mix of human, social, political, economical, technological, and organizational factors has a bearing on the level of complexity associated with RE. Overall a new sort of interaction or systemic complexity¹ needs to be reckoned and managed during the RE, where design becomes a problem solving and framing process with inherent uncertainties driven by the partial unknowns. Dealing with such design complexity impacts two areas of RE: (1) the strategic decision-making in generating and selecting requirements, and understanding their impact on the ecology; and (2) selecting tractable design-approaches that make complex system designs possible. The former is the concern of how to relate complexity with stakeholders within their ‘subject’ world, whereas the latter influences behaviors within the ‘design’ world.

In the brave new world of RE the implementation of requirements impacts not only on the technical systems, but also on their organizational and social settings that increases interaction complexity. In addition, the increased variety of requirements that emanate from diverse communities need to be negotiated, evaluated, and selected compounding the complexity. Qualitative and often structural conceptual models like goal or business models, whilst rich and useful in representation and analysis for design are less helpful for stakeholder evaluation and understanding the interaction complexity. Due to the design complexity, it is also difficult for some stakeholders to visualize and understand the system’s behavior. It is tempting to think that “stakeholders understand a description when they don’t really understand it at all”. Therefore, many questions remain poorly understood concerning design complexity: What is the nature of design complexity and increased interaction complexity, and how can we identify, analyze and measure it?

One way of coping with design complexity is through architectural designs and control that allow ‘nearly’ decomposable system designs. This mitigates complexity by ensuring that interactions among components are weak, though not negligible. Thus, designing a nearly decomposable system in the face of uncertain requirements becomes a difficult satisficing problem. Perhaps, not surprisingly, a new look at design methodologies can play the central role here. An emergent design methodology will be an improvement over conventional *a priori* methodologies. Open source systems, following nontraditional methodologies, for example evolve systems faster than traditional life cycle and requirements driven development approaches [18]. The co-design and co-evolution of the system and its stakeholders seems to play here a pivotal role, as does the fact that “open source systems entail internal architectures with orthogonal features, sub-systems, or modules, as well as external system release

¹ This should not be confused with computational complexity as defined by well-known complexity notions like NP hard problems.

Table 2. Four key requirements principles

| Principle | Description | Rationale |
|--|--|--|
| <p>Intertwine Requirements and Contexts</p> | <p>Requirements are interdependent with their social and technical contexts. As boundary objects in the intersection of the technical and social domains, design requirements seek to constantly resolve the gap between problems and solutions. Specification and implementation intertwining is long recognized, but the social context and specification intertwining is growing in importance. <i>RE Problem addressed: design requirements problem.</i></p> | <p>Intertwining between business, organizational, community context and requirements is as important as it is between requirements and software. <i>New RE Issues : Fluidity of designs, , Business process focus; Integration focus, Distribution of requirements</i></p> |
| <p>Evolve Designs and Ecologies</p> | <p>Design ideas and artifacts evolve, from stakeholder preferences to the implementations. Evolution needs to be managed through selectively freezing some aspects while changing other aspects thus allowing increased variation, dynamic selection and diffusion of structures and behaviors. <i>RE Problem addressed: design requirements problem, specification problem.</i></p> | <p>Everything evolves, but at different rates. Design around relatively fixed evolutionary paths that allow for increased but controlled variation and effective selection and diffusion. <i>New RE issues : Fluidity of designs, Layers of requirements, Distribution of requirements, Packaged software</i></p> |
| <p>Manage through architectures</p> | <p>Architecture is the least evolving and most widely referenced anchor of the design, be it business architecture, or implementation architecture. <i>RE Problem addressed: specification problem</i></p> | <p>If well-designed, the architecture (business or software) evolves slowly, and influences and interacts with many requirements. We know poorly however how architectures shape, allow and constrain evolution. <i>New RE issues : Interdependent complexity, business process focus, Centrality of architecture</i></p> |
| <p>Recognize and mitigate against design complexity</p> | <p>The necessity to consider simultaneously a large number of issues and their non-linear interactions during design raises design and requirements complexity beyond what a single designer can understand or visualize. <i>RE Problem addressed: predictability problem.</i></p> | <p>Historically, tools aided a single designer or a small group in design decision making. New design tools need to be extended to monitor and analyze the dynamic design evolution, highlighting its trajectory and helping negotiate at the team and community level. <i>New RE issues : Interdependent Complexity, Fluidity of designs, Business process focus, Layers of requirements</i></p> |

architectures that span multiple deployment platforms”[18]. A better understanding of complexity can be obtained, if system descriptions are tested against concepts familiar to stakeholders, and multiple scenarios are played out by fixing key parameters as proposed by the architecture. Experimenting with different scenarios has proved a powerful means for discovering and refining requirements with heightened complexity [19]. Still, we need to examine: What types of interdependencies influence and affect system change and create higher levels of design complexity? How can architectural models be exploited mitigate against design complexity, and to what extent they are a cause of it?

3.5 Summary of Four Key Design Principles

The four principles are summarized in Table 2 together with the rationale for using each principle, as well as what critical RE issues motivate each. We do not claim that they all apply in all design contexts, or even that some will apply in all contexts. In contrast, when e.g. designs are expected to be fluid, when they integrate with business process, or involve significant distribution of requirements we can expect the principle of intertwining of requirements and contexts to be instrumental. Finally, these are similar to *threshold* indicators as used in claiming that abstraction is important, when the size and the number of dependencies within the software system goes beyond a certain threshold.

4 Implications

Over its thirty-year history, the idea of design requirements has changed from single, static and fixed-point statements of desirable system properties into dynamic and evolving rationales that mediate change between the dynamic business environments and the design and implementation worlds. As Fred Brooks noted in the Dagstuhl workshop: “Design is not about solving fixed problems; it is constant framing of solution spaces”. This evolution has now probably reached a new turning point characterized by unprecedented scale, complexity, and dynamism. This calls for new ways to think about requirements and their role in the design. Like earlier turning points, such as the software crisis in the 1970s, it will demand a resolute and careful intellectual response. The four requirements principles discussed have numerous implications for research of which only a tip of the iceberg has been addressed. There are also multiple implications for RE practice both at the management and at the engineering level. For the sake of brevity, we discuss below three closely related practical strategies being pursued and exposed by the four principles.

Service orientation and task distribution: The life cycle of systems must now be aligned more closely with the business process lifecycle. Service oriented architectures, possibly combined with model-driven code and test generation, are now reasonably well established at the programming level to do the job. The situation is quite different, however, at the level of business services, despite their ongoing standardization. The decomposition of monolithic business process systems into freely configurable business services has turned out to be far more complex task than expected due to the need to make business semantics explicit that were hitherto hidden in the code.

This is, however, not only true for runtime service configurations, but also when outsourcing and especially when off-shoring. The design challenge is how to tackle the domain of business semantics, which is often culture dependent. Intercultural competencies become a must for requirements engineers in such settings, where pilot cases can be a promising means to establish cultural understanding. Legal aspects are also forming an increasingly important aspect, not only in terms of how to protect intellectual property (IP)—what should I *not* offshore, if IP handling is doubtful?—but also in terms of protecting oneself against being sued by customers due to imprecise contractual agreements or.e.g not honoring open source licenses in some parts of the code.

Importance of the Edge: In the increasingly complex environments the distinction between users and developers is vanishing, and the user networks and the developer networks are becoming increasingly fuzzy and intermingled. Many contributors to system designs, and even implementations are no longer located in the kernel, but at the network edge. This situation—characterized as an evolution from *user* to *citizen*—enables greater diversity of system variants and system uses, especially in cross-cultural environments. Web-based social networks have proven to be the infrastructure of choice for such settings, and quick and dirty testing of incremental changes in limited market portions is the necessary requirements testing strategy. Accordingly, design incentives now go beyond monetary ones. *Bricoleurs* at the edge must be harnessed to contribute to usability and functionality beyond the initial enthusiasm. Thus, transparency, accountability, and maintenance of a core vision in ecologies become more important. Requirements traceability within such evolutionary processes often involves runtime monitoring towards the requirements. Finally, new design goals enter the stage. For example, industrial design ideals such as innovativeness or aesthetics of the user experience play often a larger role than pure functionality.

Capability-based platforms: For the past twenty years, the dominant trend design has been business process modeling and optimization that go together within monolithic COTS, or software product lines. As noted, we witness a move towards capability-based evolutionary platforms defined by consumption and production networks. Such networks need to analyze or define their core capabilities, and seek opportunities (often initiated from the edge, as noted above) to exploit for market or process innovations in a speedy and flexible manner. Capability-based platforms define core capabilities that can be competitively delivered with and by the networks. These platforms also hold the networks together. By fixing ‘core’ requirements and the related architecture for the efficient core implementation, the platform designer makes bets on assumptions about the speed of change related to different requirements sets associated with the platform (and the network). For example, platform strategies have been a critical success factor in the automotive industry, but this advantage can turn into a deadly trap, when economic considerations mandate that the company must be split in a manner orthogonal to the original platform and related network. Software platform strategies go thus beyond the idea of product lines as a means to manage efficiently variability in software. They need to consider distinctions between core processes (supported in the platform) and context processes (around it). These distinctions must be based on a careful analysis of market power and network strength, and anticipate technology evolution in the underlying technological standards and architectures.

To sum up, the good news is that the RE has never been more important and its criticality will continue to grow. The bad news is that RE is a different beast now. Accordingly, we need to consider RE in new ways that take into consideration the need for the alignments with business processes, the need to understand core capabilities, the need to ensure legal protection or, the need to create user buy-in, or the need to minimize training costs. In consequence, we need to expand RE research into new directions—including complexity science, industrial design, organization design, and economics- and engage these fields in a honest intellectual exchange why and how design requirements matter in the design of complex software and world.

Acknowledgments

We thank Sean Hansen, Nicholas Berente, Dominik Schmitz, and Anna Glukhova for helping to organize the workshops, and the workshop participants for inspiring discussions and Sol Greenspan for constructive comments. This research was in part funded by National Science Foundation's "Science of Design" initiative, Grant Number: CCF0613606, by DFG project CONTICI and by BMBF project ZAMOMO.

References

1. Ross, D.T., Schoman Jr., K.E.: Structured analysis for requirements definition. *Transactions on Software Engineering* SE-3(1), 6–15 (1977)
2. Frederick, J., Brooks, P.: *The Mythical Man-Month*. Addison Wesley, Reading (1995)
3. Lyytinen, K., et al.: *Design Requirements Engineering: A Ten-Year Perspective*. In: *Design Requirements Workshop*, Cleveland, OH, USA, June 3-6 (2007); Revised and Invited Papers, p. 495. Springer, Heidelberg (2009)
4. Hansen, S., et al.: Principles of Requirements Processes at the Dawn of 21st Century. *Ingenierie des Systèmes d'Information* 13(1), 9–35 (2008)
5. Schuler, D., Namioka, A.: Participatory design. In: *Proc. Lawrence Erlbaum Assoc.* (1993)
6. Simon, H.: *The Sciences of the Artificial*. MIT Press, Cambridge (1996)
7. Schon, D.: *The reflective practitioner: How professionals think in action*. Basic Books, New York (1983)
8. Bowen, J.P., Hinchey, M.G.: *Ten Commandments of Formal Methods* (1995)
9. Swartout, W., Balzer, R.: On the inevitable intertwining of specification and implementation. *CACM* 25(7), 438–440 (1982)
10. Mathiassen, L., et al.: A Contingency Model for Requirements Development. *Journal of the Association for Information Systems* 8(11), 569–597 (2007)
11. Chapin, N., et al.: Types of software evolution and software maintenance. *Journal of Software Maintenance and Evolution Research and Practice* 13(1), 3–30 (2001)
12. Cockburn, A.: *Agile Software Development*. Addison-Wesley, Reading (2002)
13. Berger, C., et al.: Customers as co-designers. *Manufacturing Engineer* 82(4), 42–45 (2003)
14. Lehman, M.M.: Software Evolution. *Encyclopedia of Software Engineering* 2, 1507–1513 (2002)
15. Nuseibeh, B.: Weaving together requirements and architectures. *Computer* 34(3), 115–119 (2001)
16. Pohl, K., et al.: *Software Product Line Engineering: Foundations, Principles and Techniques*. Springer, Heidelberg (2005)

17. Godet, M.: Scenarios and strategic management. Butterworth-Heinemann, Butterworths (1987)
18. Scacchi, W.: Understanding Open Source Software Evolution. In: Software Evolution and Feedback, Theory and Practice. Wiley, New York (2006)
19. Carroll, J.M.: Scenarios and design cognition. In: Proceedings of IEEE Joint International Conference on Requirements Engineering, pp. 3–5 (2002)
20. Glasser, B., Strauss, A.: The development of grounded theory. Alden, Chicago (1967)
21. Zave, P.: Classification of research efforts in requirements engineering. *ACM Comp. Surv.* 29(4), 315–321 (1997)
22. van Lamsweerde, A.: Requirements Engineering in the Year 00: A Research Perspective. In: Proceedings of the 22nd International Conference on Software Engineering, pp. 5–19 (2000)
23. Cheng, B., Atlee, J.: Current and Future Research Directions in Requirements Engineering. In: Lyytinen, K., Loucopoulos, P., Mylopoulos, J., Robinson, W. (eds.) Design Requirements Engineering – A Ten-Year Perspective. LNBIP, vol. 14. Springer, Heidelberg (2009)

Appendix: Data Collection and Analysis

The new requirements engineering challenges reported in this essay were discussed in a series of workshops, designed to accomplish three objectives: (1) engage separate research communities in a dialogue, (2) strengthen design science principles, and (3) open new vistas for the research on design requirements. The first workshop was held in the United States in June 2007, while the second workshop was held in October 2008 in Germany (see [3] for more detailed description). The discussions were recorded in the workshop Wikis (see e.g. <http://weatherhead.case.edu/requirements>; <http://www.dagstuhl.de/Materials/index.en.phtml?08412>). In addition we undertook a field study to understand the perspectives of practitioners on successful requirements practices, anticipated developments, and new challenges in design environments [5].

The ICoP Framework: Identification of Correspondences between Process Models

Matthias Weidlich¹, Remco Dijkman², and Jan Mendling³

¹ Hasso-Plattner-Institute, University of Potsdam, Germany
matthias.weidlich@hpi.uni-potsdam.de

² Eindhoven University of Technology, The Netherlands
r.m.dijkman@tue.nl

³ Humboldt-Universität zu Berlin, Germany
jan.mendling@wiwi.hu-berlin.de

Abstract. Business process models can be compared, for example, to determine their consistency. Any comparison between process models relies on a mapping that identifies which activity in one model corresponds to which activity in another. Tools that generate such mappings are called matchers. This paper presents the ICoP framework, which can be used to develop such matchers. It consists of an architecture and re-usable matcher components. The framework enables the creation of matchers from the re-usable components and, if desired, newly developed components. It focuses on matchers that also detect complex correspondences between groups of activities, where existing matchers focus on 1:1 correspondences. We evaluate the framework by applying it to find matches in process models from practice. We show that the framework can be used to develop matchers in a flexible and adaptable manner and that the resulting matchers can identify a significant number of complex correspondences.

1 Introduction

Organisations compare business process models to identify operational commonalities and differences. Such comparisons are, for example, necessary when organisations merge and need to determine and resolve the differences between their operations, and when an organisation needs to check whether its operations conform to an company-wide or industry-wide standard. The first step when comparing business process models is always to determine which activities in one business process model correspond to which activities in the other. This step is called the *matching* step and can be supported by tools that are called *matchers*. This paper presents a framework that can be used to develop such matchers.

A major challenge for matchers is that business process models often do not use the same level of detail and the same words to describe activities. For example, there is a problem with level of detail if one business process model contains an activity ‘Check Invoice’, whereas the other describes the same activity using a sequence of ‘Verify Customer Data’, ‘Decide on Correctness of Data’, and

‘Approve Invoice’. This problem relates to refinement and meronymy. There is a problem with the words used to describe an activity if there are two labels ‘Check Invoice’ and ‘Verify Bill’ that point to the same activity, although using very different words. These are problems of synonymy and homonymy. This problem of heterogeneous representation and description is of striking relevance in practice [12], because organisations usually do not align the way in which they describe their business processes. It also is the major reason why the comparison of related processes requires an extensive amount of manual preprocessing.

Therefore, this paper presents automated assistance for this ‘preprocessing’ step, by proposing a re-usable framework for identifying correspondences between activities in one process and equivalent activities in a similar process, while taking into account that equivalent activities may be modelled at different levels of granularity, have different labels, and have different control-flow relations to other activities. The framework is specifically tailored to also deal with complex 1:n matches (i.e., each activity can be matched to an arbitrary number of other activities), where existing matchers focus on elementary 1:1 matches (i.e., each activity can be mapped to at most one other activity). The framework consists of an architecture and a set of re-usable matcher components. Matchers can be developed in the framework by composing them from existing components and, if desired, newly developed ones. We present matchers that we implemented within the framework, and we evaluate them using models from practice. Our contribution is a framework with re-usable components to automatically find both 1:1 and 1:n matches between activities from similar business processes.

While this contribution is significant to the process model matching field, it also has implications for schema matching in general [3]. The research area of schema matching covers a broad set of techniques, including structural analysis and natural language processing to automatically identify the elements of one schema (which are activities of a process model in our case) that match to those of a second schema. Unfortunately, there has been a predominant focus on 1:1 matches in the schema matching community, such that *‘1:n and n:m mappings [...] are currently hardly treated at all’* [3]. Although there are notable exceptions, like iMAP [4], these techniques cannot be applied in our context as they have been tailored for data models and partially also use the extension of a database. We further discuss this issue when reviewing related work in section 5.

The paper is structured as follows. Section 2 describes the background of the problem by an example of process models with activity correspondences. Section 3 introduces the ICoP framework for developing matchers. It describes both the framework itself and the techniques that are implemented as components of the framework. Section 4 presents the evaluation of the framework. Section 5 assesses our contribution in the light of related work. Section 6 concludes.

2 Background

The relevance of finding matches in pairs of corresponding process models has been described before [12]. In this section, we aim to illustrate the problem and

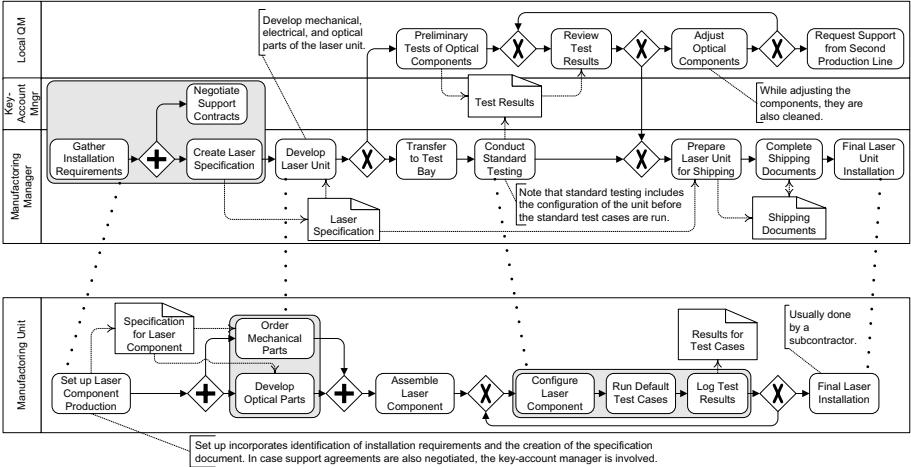


Fig. 1. Two BPMN process models with one 1:1 and three 1:n correspondences

the underlying combinatorial challenges using a pair of example processes. We also introduce terminology that we will use throughout the paper.

Figure 1 shows the operations of a company that produces lasers. The upper model depicts the process in a way that hand-overs between different departments can easily be traced. The lower process focusses more on the operational details as being conducted in the manufacturing unit. Roughly speaking, both processes describe that first the component production is set up, then the laser units are developed, assembled, tested, prepared for shipment, and finally installed at the sight of the customer. There are *matches* of different complexity between the activities of the two models. In the general case, a match refers to a correspondence, which is an element of the powerset of activities of a first model times the powerset of activities of a second model. Thus, a match is denoted by a tuple (A_1, A_2) of two sets of activities. A match (A_1, A_2) is called *elementary match*, if $|A_1| = |A_2| = 1$. An example are the ‘Final Laser Unit Installation’ (FLUI) and the ‘Final Laser Installation’ (FLI) activities. We say that FLUI matches or corresponds to FLI. This match pair can be easily identified due to the syntactic similarity of the labels and the same position at the end of the processes.

Figure 1 also highlights three complex matches by dotted lines. A match (A_1, A_2) is called *complex* if at least one activity set in the pair contains more than one element, i.e., $|A_1| > 1$ or $|A_2| > 1$. The first complex match on the left-hand side shows that ‘Set up Laser Component Production’ of the lower model describes the same activities as the set of ‘Gather Installation Requirements’, ‘Negotiate Support Contract’, and ‘Create Laser Specification’ in the upper model, but at a different level of granularity. Thus, it is a 1:3 match as there is one activity corresponding to three activities in the upper model.

The goal of automatic process model matching is to find all those matches that are meaningful. Usually, there are conditions that a collection of meaningful matches has to obey, for instance, that the matches do not *overlap*. That is, for every pair of matches (A_1, A_2) and (A_3, A_4) in the mapping it holds $A_1 \cap A_3 = \emptyset$ and $A_2 \cap A_4 = \emptyset$. We call this collection of matches a *mapping*. From a structural point of view, a mapping is an element of the powerset of matches.

Identification of complex matches imposes serious challenges. For the case of 1:1 matches, it might be feasible to analyse the whole set of potential matches, which corresponds to the Cartesian product of activities of two process models. In contrast, the possibility of 1:n matches increases the size of the set of potential matches significantly. For two process models with n and m activities, respectively, the number of 1:x matches is given by $n * \binom{m}{x} + m * \binom{n}{x}$. Here, the binomial coefficient $\binom{n}{x}$ defines the number of x-element subsets of an n-element set. For instance, the two process models depicted in Fig. 1 consist of 13 and 8 activities, respectively, which, in turn, yields 104 potential 1:1 matches. However, considering the possibility of 1:2 and 1:3 matches in addition increases the set of potential matches by $13 * \binom{8}{2} + 13 * \binom{8}{3} = 1092$ matches for one direction, and $8 * \binom{13}{2} + 8 * \binom{13}{3} = 2912$ matches for the other direction. One might argue that for the case of process models, the value of x might be bound by some threshold instead of the number of nodes of the respective model. However, our experience shows that this threshold must not be set too low (we encountered up to 1:9 matches), such that the amount of possible combinations still hinders any attempts to analyse the whole set of potential matches. These observations call for adequate search heuristics and an architecture that evaluates efficiently.

3 The ICoP Framework

This section introduces the ICoP framework for automatic derivation of matches between two process models. Section 3.1 discusses the overall architecture. We elaborate on the four types of components, namely searchers, boosters, evaluators, and selectors, in detail and present exemplary realisations in Sections 3.2 to 3.5.

3.1 Architecture

The overall architecture of the ICoP framework stems from the observation that the number of (1:n) matches between two business process models is potentially large and therefore it is not feasible to explore all possible matches exhaustively. Instead, the ICoP framework proposes a multi-step approach, which is illustrated in Fig. 2. Given two process models, *searchers* extract potential matches based on different similarity metrics and heuristics for the selection of activities. The result of the search stage is a multiset of matches due to the possibility of multiple searchers identifying the same potential matches. Each match is assigned a *match score*, which results from the scoring function implemented by the searcher to

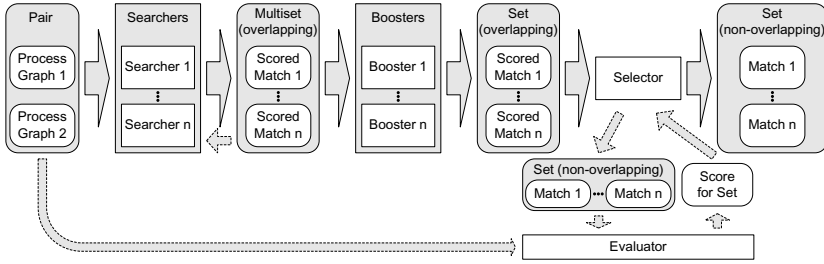


Fig. 2. The architecture of the ICoP framework

select potential matches. Note that a searcher may use the knowledge about potential matches that have been identified by other searchers already.

After completion of the search stage, the scored potential matches are conveyed to *boosters*. These components ‘boost’ the matches that are returned by the searchers using certain heuristics in order to aggregate matches, remove matches, or adapt the score of a match. As part of the boosting stage the *multiset* of potential matches is transformed into a *set* of potential matches by aggregating matches that have been identified by multiple searchers.

Subsequently, a *selector* builds up the actual mapping from the set of potential matches. That is, it selects the best matches from the set of potential matches, in such a way that the constraint that matches are not overlapping is satisfied. The selection of the best matches can be guided by two kinds of scores. On the one hand, the individual match scores of the potential matches can be exploited. On the other hand, an *evaluator* can be utilised, which assigns a single *mapping score* to a mapping. An evaluator may use knowledge about the original process models to compute this score. The selection is an iterative process. In each iteration the selector selects a set of matches, the evaluator computes the score for this set, upon which the selector either decides to modify the set of matches and continue the selection, or to complete the selection. Once the selection procedure completes, the selector produces the final mapping between elements of the process models.

3.2 Match Searchers

Searchers identify potential 1:1 and 1:n matches between two process models along with a score that indicates the quality of the match. Such a match is denoted by (A_1, A_2, s) with s being the match score based on the similarity of the matched activities. Therefore, we will also refer to the similarity score of a match. The similarity of a match can be determined based on various aspects, including the labels or descriptions of the matched activities and structural or behavioural relations between those activities.

We now introduce four searchers that have been implemented in the ICoP framework. Although a similarity score for the activity labels is at the core of all searchers, they incorporate similarity metrics for different aspects of labelling.

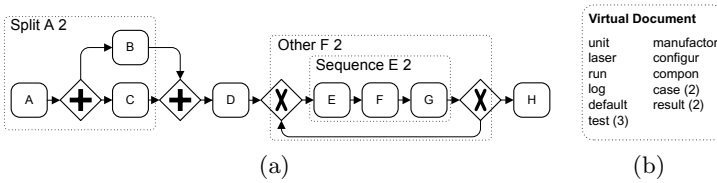


Fig. 3. (a) groups of activities for the lower model of Fig. 1 (b) virtual document for ‘Sequence E 2’

Similar Label Searcher. The purpose of this searcher is to identify straightforward 1:1 matches based on a high syntactic similarity of activity labels. It computes the Cartesian product of activities of two process models and selects all pairs of activities for which the string edit similarity of their labels is above a given threshold. For two strings s_1 and s_2 the string edit similarity is defined as $sim(s_1, s_2) = 1 - \frac{ed(s_1, s_2)}{\max(|s_1|, |s_2|)}$ with $ed(s_1, s_2)$ as the string edit distance, i.e., the minimal number of atomic character operations (insert, delete, update) needed to transform one string into another [5]. As the string edit similarity is a rather strict criterion (different orders of words and linguistic phenomena such as synonymy are neglected), the potential matches are identified with high confidence, such that the initial score for these matches is set to one by the searcher. For the scenario illustrated in Fig. 1 and a threshold of 0.8, for instance, the searcher identifies the match between the activities describing the installation of the laser. Clearly, the runtime complexity of this searcher depends solely on the number of activities of the respective process models.

Distance Doc Searcher. This searcher follows a two step approach in order to identify potential 1:n matches. First, activities of both process models are grouped heuristically. Second, the similarity between such a group of activities in one model and all single activities in the other model is assessed. Assuming that it is more likely that activities that are closer to each other should be in the same group, we use the graph distance to group activities. The graph distance between two activities is the number of edges on the shortest path from one activity to the other. Given a base activity and a distance, we look for four types of groups:

- Sequences, which are determined by a base activity and the activities on a directed path of the given length (distance) from the base activity.
- Splits, which are determined by a base activity and the activities that can be reached from the base activity and that are within the given distance. The base activity can or cannot be considered in such groups, depending on whether the choice leading to the split is or is not modelled explicitly.
- Joins, which are determined by a base activity and the activities from which the base activity can be reached and that are within the given distance. The base activity can or cannot be considered in such groups.

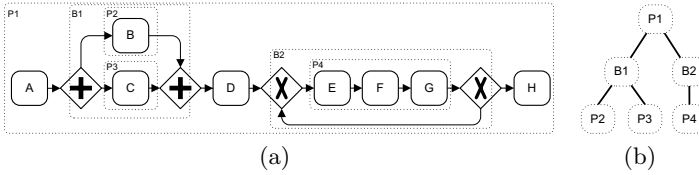


Fig. 4. (a) RPST fragments for the lower model of Fig. 1 (b) the corresponding RPST

- Others, which are the groups that consist of all activities that are within the given distance of a base activity (not considering the direction of edges). In contrast to sequences, these activities are not necessarily on a path.

For the lower model of Fig. 1, Fig. 3(a) shows examples of a group that is a sequence with base ‘E’ and distance 2, a split with base A and distance 2, and an ‘other’ with base F and distance 2. The distance doc searcher identifies all groups of activities in a process model by taking each of the activities as a basis and creating each possible type of group for each possible graph distance value. A maximum distance value can be set as a parameter. The groups are collected in a set to avoid duplication.

Once all groups of activities have been identified, the notion of ‘virtual documents’ is used to score their similarity. Virtual documents were introduced for aligning ontologies [6]. A virtual document of a node consists of the words from all textual information that is related to that node. Given two virtual documents, their similarity can be calculated based on their distance in a vector space, in which the dimensions are the terms that appear in the documents and the values for the dimensions are computed using term frequency [7]. In our setting, a virtual document for an activity consists of the terms that are derived from the activity label and, if this information is available, the labels of the roles that are authorized to perform the activity, the assigned input and output data, and a textual description of the activity. For a group of activities, the virtual document is derived by joining the documents of the respective nodes. Note that the creation of virtual documents includes a normalization of terms, filtering of stop-words, and term stemming [8]. Fig. 3(b) illustrates the terms of the virtual document for the group ‘Sequence E 2’. These terms originate from the activity labels, the label of the associated role, and the data output of one activity, while we also applied stop-word filtering and term stemming. Note that the runtime complexity of this searcher is heavily influenced by one parameter besides size and structure of the models. The maximal graph distance value for grouping activities might increase complexity significantly.

Fragment Doc Searcher. This searcher is similar to the distance doc searcher, except that it relies on the Refined Process Structure Tree (RPST) [9] for grouping activities. The RPST parses a process model into a hierarchy of fragments with a single entry node and a single exit node. Fig. 4(a) depicts these fragments for the lower process model in Fig. 1. The fragments are defined in such a way

that they do not overlap. Consequently, they form a tree-structure, as illustrated in Fig. 4(b) (cf., [9]). We leverage the hierarchy of fragments in order to select the groups of activities that will be considered by the searcher. Starting with the leaf fragments of the RPST, the tree is traversed upwards up to a height that is given as a parameter. All traversed fragments are considered by the searcher that creates two virtual documents for each fragment, one containing all activities of the fragment and one containing all activities except those that are boundary nodes of the fragment. As in case of the Distance Doc Searcher, the similarity of the virtual documents is assessed using a vector space approach. For the example in Fig. 4 and a height threshold of two, the searcher considers the fragments B1, B2, P2, P3, and P4. The runtime complexity of this searcher mainly depends, besides size and structure of the models, on the height up to which the RPST is traversed upwards for identifying groups of activities.

Wrapup Searcher. This searcher resembles the Similar Label Searcher, as it also aims at deriving potential 1:1 matches by analysing the string edit similarity for the labels of a pair of activities. In contrast to the Similar Label Searcher, however, not the whole Cartesian product of activities of two process models is considered. In fact, solely activities that have not been addressed in potential matches retrieved by other searchers are taken into account. Obviously, the Wrapup Searcher has to be run after all other searchers. In addition, the threshold for the similarity of two activity labels is typically set to a lower value than for the Similar Label Searcher.

3.3 Match Boosters

After potential 1:1 and 1:n matches have been identified, the multiset of scored matches is propagated to a set of boosters. We implemented the following four boosters as part of the ICoP framework.

Cardinality Booster. This booster reduces the *multiset* of potential matches to a *set* by aggregating the similarity scores for potential matches that associate the same (sets of) activities to each other. Two matches (A_1, A_2, s_1) and (A_3, A_4, s_2) with $A_1 = A_3$ and $A_2 = A_4$ are replaced by a match (A_1, A_2, s_a) , such that $s_a = s_1 + (1 - s_1) * s_2$ is the first score increased by the second relative to its current value. Note that the operation is symmetric and can iteratively be applied if more than two scores need to be aggregated.

Subsumption Booster. The idea behind this matcher is that a 1:n match (A_1, A_2, s_1) might subsume another match (A_3, A_4, s_2) , such that $A_1 = A_3$ and $A_4 \subset A_2$. As similarity scoring based on vector spaces tends to favour documents consisting of a small number of terms, the subsumed matches will have higher initial similarity scores on average. To countervail this effect, we boost a 1:n match, if it subsumes other matches. If the match (A_1, A_2, s_1) subsumes the match (A_3, A_4, s_2) , its similarity score is increased relative to the current value, such that $s_1 := s_1 + w_s * (1 - s_1) * s_2$ with $w_s \in [0..1]$ as a weighting factor.

Tree Depth Ratio Booster. In contrast to the aforementioned boosters, this booster considers solely a single match and boosts the score of a match, if its activities show a certain structural property that is evaluated based on the RPST. Given a match (A_1, A_2, s_1) , we determine the least common ancestors (LCA) of A_1 and A_2 in the RPST of the respective process model, denoted by $lca(A_1)$ and $lca(A_2)$. Further on, let $maxDepth_1$ and $maxDepth_2$ be the maximal depths of a fragment in the RPSTs of the two process models. Based thereon, we determine two ratios by relating the depth of the LCA to the maximal depth of the tree, i.e., $r_1 = \frac{lca(A_1)}{maxDepth_1}$ and $r_2 = \frac{lca(A_2)}{maxDepth_2}$, with $r_1, r_2 \in [0..1]$. Boosting the match takes places, if the average of the two ratios is above a threshold, which indicates that both LCAs are relatively low in the tree. That, in turn, can be interpreted as a hint for a good quality of the match. In this case, the similarity score of the match is increased according to the average of the two ratios, relative to the current similarity score, i.e., $s_1 := s_1 + (1 - s_1) * w_r * 0.5 * (r_1 + r_2)$ with $w_r \in [0..1]$ as a weighting factor.

Distance Ratio Booster. This booster also considers solely single matches. Here, the structural property that is evaluated relates to the graph distance as mentioned above. For each of the sets of activities A_1 and A_2 of a match (A_1, A_2, s_1) , we determine the maximal distance between two activities of this set. That is, for each activity we compute the distance from or to all other activities and select the minimal distance. Then, the maximal value of all these minimal distances is chosen, denoted by $maxDistance(A_1)$ and $maxDistance(A_2)$, respectively. Furthermore, let $maxDistance_1$ and $maxDistance_2$ be the maximal distances that can be observed between two activities that are connected by a path in the two process models. Again, we define two ratios $r_1 = 1 - \frac{maxDistance(A_1)}{maxDistance_1}$ and $r_2 = 1 - \frac{maxDistance(A_2)}{maxDistance_2}$, with $r_1, r_2 \in [0..1]$. If the average of both ratios is above a threshold, the similarity score of the match is increased according, i.e., $s_1 := s_1 + (1 - s_1) * w_d * 0.5 * (r_1 + r_2)$. Here, $w_d \in [0..1]$ is the weighting factor.

3.4 Mapping Selectors

Once the match similarities have been adapted by the match boosters, a selector extracts a mapping from the set of scored potential matches. As mentioned above, any mapping has to satisfy the constraint of non-overlapping matches. While selectors follow different notions of quality for a mapping, they have in common that the derivation of the optimal mapping (w.r.t. to the chosen quality criterion) is a computationally hard problem. Therefore, our selectors follow a greedy or 1-look-ahead strategy. Experiments in a similar context revealed that results obtained by a greedy matching strategy are close to those obtained with an exhaustive strategy [10]. The ICoP framework consists of the following selectors.

Match Similarity Selector. This selector selects a non-overlapping mapping solely based on the similarity scores assigned to potential matches. The match with the highest score is selected (in case of multiple matches with an equal

score one is chosen arbitrarily) and the set of potential matches is reduced by the matches that are overlapping with the selected match. The mapping is constructed iteratively until the highest score for a potential match is below a given threshold. Besides this greedy strategy, we also implemented a 1-look-ahead strategy, which optimizes the score for the succeeding iteration in case of multiple matches with equal scores.

Mapping Similarity Selector. This selector neglects the scores assigned to potential matches and relies solely on the score for a (partial) mapping as provided by an evaluator. The matches to build the mapping are iteratively selected such that in each step the match leading to the maximal score for the mapping is selected. In case of multiple matches meeting this requirement, one is chosen arbitrarily. The procedure terminates once the mapping score cannot be increased any further. Again, we implemented this greedy strategy and a 1-look-ahead variant that selects the match that leads to the maximal mapping score in the succeeding iteration.

Combined Selector. This selector uses both, match scores and mapping scores as provided by an evaluator. In a first step, the highest match score is determined. All potential matches having assigned this score are then selected for the mapping (in case this is not possible due to overlapping matches, selection is randomised). In a second step, the mapping is iteratively extended with the match that maximises a combined score that is built from its individual match score and the mapping score as computed by the evaluator. Here, both scores are combined in a weighted way (e.g., the mapping score might have a bigger impact than the match score). The procedure terminates if the combined score cannot be increased any further. Again, the second step of the selector has been implemented as a greedy and as a 1-look-ahead strategy.

3.5 Mapping Evaluators

A mapping selector can use a mapping evaluator to score mappings. Given a (partial) mapping, the evaluators return a single score for the quality of the mapping. Of course, different notions of quality can be considered. In the ICoP framework, we implemented the two following evaluators.

Graph Edit Distance Evaluator. This evaluator scores a given mapping based on the graph edit distance of the two original process models that is induced by the mapped activities. For a pair of graphs and a mapping between their nodes, the graph edit distance defines the minimal number of atomic graph operations (substitute node, insert/delete node, (un)grouping nodes, substitute edge, insert/delete edge) needed to transform one graph into another [11]. For these operations, only mapped nodes are considered to be substituted and potentially grouped. For example, to transform the upper process to the lower process in Fig. 1, considering the mapping that is represented in the figure, operations that have to be performed include: substituting ‘Final Laser Unit Installation’ by ‘Final Laser Installation’; grouping ‘Order Mechanical Parts’ and ‘Develop

Mechanical Parts’; and inserting ‘Assemble Laser Component’. Optionally, operations can be weighted instead of just counted (e.g.: a node insertion can count for 0.8 instead of 1). The graph edit distance can be leveraged to define a similarity score according to [12]. This score is computed as 1 minus the fraction of the given mapping and a hypothetical ideal mapping. The ideal mapping matches each node and each edge in one process with a node or an edge in the other process with a quality of 1.0. The similarity score defines the quality of the mapping according to the Graph Edit Distance Evaluator.

Path Relation Evaluator. The evaluator scores a given mapping based on whether the path relations are preserved for the activities of a pair of matches of the mapping. For a pair of matches $M_1 = (A_1, A_2, s_1)$ and $M_2 = (A_3, A_4, s_2)$, we derive the number of preserved path relations $pre(M_1, M_2) = |\{(a_1, a_2, a_3, a_4) \in (A_1 \times A_2 \times A_3 \times A_4) \mid path(a_1, a_3) \Leftrightarrow path(a_2, a_4)\}|$ with $path(x, y)$ being a predicate that denotes the existence of a path from activity x to activity y . Then, the evaluation score for the pair of matches M_1 and M_2 is defined as $s(M_1, M_2) = \frac{pre(M_1, M_2)}{|A_1 \times A_2 \times A_3 \times A_4|}$. Based thereon, the score for the mapping is computed by iterating over the Cartesian product of matches and computing the average of their scores.

4 Evaluation

We evaluate the matchers by comparing the matches that they discover to matches that business process analysts found in a collection of 20 pairs of process models. 3 pairs are taken from a merger in a large electronics manufacturing company. Each of these pairs represents two processes that have to be merged. 17 pairs are taken from municipalities. Each of these pairs represents a standard process [13] and an implementation of this standard process by a municipality. Each process model from the collection has, on average, 31.1 nodes, with a minimum of 9 nodes and a maximum of 81 nodes for a single process model. The average number of arcs pointing into or out of a single node is 1.2 and the average number of words in the label of a single node is 2.8.

For the 20 process model pairs, process analysts determined a total of 520 matched activity pairs. Of these 520 pairs, 221 were part of a complex match. However, the distribution of these complex matches in our model collection shows a high variation. For instance, for 3 out of the 20 model pairs (the model pairs from the merger), more than 90% of the activity pairs relate to complex matches. In turn, 6 model pairs contain solely elementary 1:1 matches.

We evaluate the performance of the matchers in terms of precision and recall. The precision is the fraction of found activity matches that that is correct (i.e., that is also found by process analysts). The recall is the fraction of correct activity matches that is found. The F-Score combines precision and recall in one value. We also compute the Overall score, an alternative metric that has specifically been developed for measuring the quality of schema matches [14]. Note that all metrics are based on activity pairs. Thus, complex matches are split up into activity pairs, e.g., $(\{a, b\}, \{x\})$ yields two activity pairs $(\{a\}, \{x\})$ and $(\{b\}, \{x\})$.

C_h = Activity pairs identified by human observer
 $CC_h \subseteq C_h$ = Activity pairs that are part of a complex match
 $CE_h \subseteq C_h$ = Activity pairs that are elementary matches (CC_h and CE_h partition C_h)

C_m , CC_m , and CE_m are analogously defined as the sets of activity pairs, complex matches, and elementary matches identified by matcher m .

$$\begin{aligned}
 \textit{precision} &= |C_m \cap C_h|/|C_m| & \textit{recall} &= |C_m \cap C_h|/|C_h| \\
 \textit{recall-elementary} &= |CE_m \cap CE_h|/|CE_h| & \textit{recall-complex} &= |CC_m \cap CC_h|/|CC_h| \\
 \textit{F-score} &= 2 \cdot (\textit{precision} \cdot \textit{recall})/(\textit{precision} + \textit{recall}) \\
 \textit{Overall} &= \textit{recall} \cdot (2 - 1/\textit{precision})
 \end{aligned}$$

For our evaluation, we created five matchers within the ICoP framework.

Baseline Matcher. This matcher represents the greedy graph matcher presented in [12] and consists of a Wrapup Searcher, a Graph Edit Distance Evaluator, and a Mapping Similarity Selector. This matcher identifies solely elementary matches and achieves high precision and recall in doing so. Therefore, we use it as a baseline benchmark for our framework, which focuses on improving results with respect to complex matches.

Matcher A. This matcher consists of all searchers (cf., Section 3.2) and a Look Ahead Match Similarity Selector. Thus, it demonstrates the pure performance of our searchers.

Matcher B. This matcher extends Matcher A by incorporating all four match boosters introduced in Section 3.3. Therefore, this matcher shows the impact of the boosters on the matching process.

Matcher C. This matcher consists of all searchers, but in contrast to matcher A, it takes the evaluation of (partial) mappings into account when selecting a mapping. That is, it relies on the Path Relation Evaluator, while a Look Ahead Combined Selector is used to demonstrate its effect.

Matcher D. This matcher consists of all searchers and evaluates partial mappings, but uses another mapping evaluator than matcher C, i.e., a Graph Edit Distance Evaluator.

Note that matchers A and B, as well as C and D, are very similar. The former rely solely on the scores assigned to matches in order to build up the mapping, whereas the latter use a combined approach that also considers the scores derived by evaluating a (partial) mapping.

Fig. 5 depicts the results of applying these five matchers to the set of 20 model pairs, by

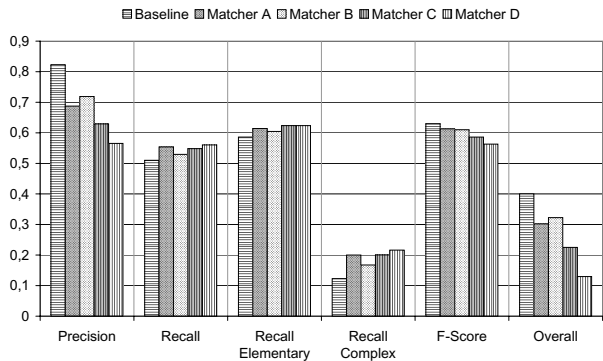


Fig. 5. Metrics derived for the whole model collection

showing the precision, recall, F-Score, and Overall as defined above for one specific configuration of each matcher that maximises the F-score for the whole model collection. In addition to presenting the recall as a whole, it presents the recall for the complex 1:n matches and the elementary 1:1 matches separately. Note that, although the baseline matcher is not meant to detect complex matches, it does return some, due the fact that some activity pairs that it identifies as 1:1 matches are actually part of a complex match.

The results show that the architecture works and, while it has a more adaptable modular setup, produces the same results as were obtained by the more rigid matcher that we developed in prior work [12]. The results also show that the architecture can be used to produce matchers that detect complex 1:n matches and that their recall is better than that of our baseline matcher. Unfortunately, the complex matchers improve recall at the expense of precision, leading to an F-Score that is slightly lower than that of the baseline matcher. The comparison of the results for matchers A and B reveals that the application of match boosters increases the precision, at the cost of decreased recall. Similarly, the Path Relation Evaluator in matcher C leads to a better precision than the Graph Edit Distance Evaluator in matcher D, which, again, is traded for recall to a certain extent.

There is a large difference between the results with respect to the use-case from which they were derived. As indicated 3 model pairs originated from a comparison in a merger, while 17 originated from a comparison of standard processes to their implementations. The results for the merger model pairs are much worse than the results for the standard process comparison pairs. With an F-Score of around 0.3 the results for the merger model pairs are so bad that we conclude that the matchers cannot be used for this use-case. Qualitative analysis of the results shows that the reason for the bad performance of the matchers for this use-case mainly stems from the fact that the activity labels of matched tasks are very different. This leads to the conclusion that, in order for the current matchers to work, there must be some level of similarity between activity labels of similar activities; a pre-condition that is met in the standard process implementation use-case, but not in the merger use-case.

5 Related Work

Our work can be related to two main areas of research, namely process model similarity and database schema matchers that aim at finding complex matches.

Work on process model similarity can be traced back to formal notions of behavioural equivalence (see [15]) and integration of process models (see [16,17,18,19,20]). These works assume that correspondences between process models have been identified, but do not discuss how these correspondences can be found. Recent research in this area provides promising solutions to automatically match activities in pairs of process models, mainly as a vehicle to automatically calculate a similarity value for the models altogether. Structural and semantic information is used in [21], behavioural information in [22], and

graph-edit distance techniques are used as well [10]. All these approaches only identify 1:1 matches between activities, just as many approaches from schema matching do [3,7]. With our ICoP framework we address the need to automatically propose complex matches between process model activities, which has been identified in [1].

One of the few database schema matching approaches that consider complex matches is the iMAP system [4]. iMAP inspired our work, as it introduces the idea of searching the space of potential complex matches using a set of searchers implementing different search heuristics. Subsequently, the similarity of target entities and potential matches is analysed in order to select the final mapping. In contrast to our work, iMAP searchers exploit the value distribution of instance data and also take domain knowledge (e.g., domain constraints) into account. Another fundamental difference is the search result, as iMAP searchers derive (linguistic, numeric, structural) mapping expression. While these expressions are of crucial importance for relating database entities to each other, they might be derived automatically in case of process model correspondences. Given a match between one activity in one process model and multiple activities in a second model, the model structure can be leveraged to decide whether the former corresponds to the conjunction, disjunction, or another logical combination of the latter. Similarly, the approach presented by Xu and Embley [23] relies on the discovery of characteristics for the instance data, the application of domain ontologies that describe expected data values, and further external knowledge (i.e., WordNet relations). Other work that aims at finding complex matches uses correlation mining techniques. The DCM framework [24] proposes to mine web query interfaces in order to identify grouping attributes, i.e., attributes that tend to be co-occurring in web interfaces. This knowledge is exploited to mine negative correlations between groups of attributes, which, in turn, hint at potential complex matches. Note that there are other matchers, e.g., Cupid [25], that retrieve complex matches by just applying a static similarity threshold for the selection of matches. Given a similarity matrix for all model elements, various match combinations for a single element might show similarity values above the threshold, such that complex matches are created. However, such an approach does not hint at strategies that are used to identify complex matches, as it assumes this knowledge to be already encoded in the similarity matrix.

We summarize that the few existing approaches for finding complex matches extensively rely on instance data and external knowledge. The former is not always available for process models, while the latter raises the question of how to utilize external knowledge for a dedicated domain. Nonetheless, the use of instance data and external knowledge are promising directions for future work.

6 Conclusion

This paper presents the ICoP framework, which provides a flexible and adaptable architecture for the implementation of matchers, by splitting up matchers into searcher, booster, evaluator and selector components. Also, it enables the

development of matchers that detect complex 1:n matches, where existing matchers focus on detecting elementary 1:1 matches. Experimental results show that the framework can reproduce results of existing matchers by composing them from separate searcher, booster, evaluator, and selector components. The results also highlight that we are able to identify a significant number of complex 1:n matches. While this demonstrates the potential of our framework for improving matching results, we also explicated that, compared to existing 1:1 matchers, the increase in recall is often traded for a decrease in precision. Finally, the experiments show that a minimal level of label similarity for similar activities is required for the matchers to produce acceptable results. This level is met in case matches are determined between standard processes and their implementations, but not for the use-case in which matches are determined between processes that must be merged.

We aim at addressing this phenomenon in future work. In order to counteract the decrease in precision when considering complex matches, we plan to integrate the usage of external knowledge into the ICoP framework. The usefulness of applying such knowledge in general, and WordNet in particular, has been demonstrated by the aforementioned approaches for matching data schemas [4,23]. It is worth to mention that external knowledge for the domain of business processes is also available in terms of several reference models such as the MIT Process Handbook. Moreover, we aim at extending the framework towards n:m matches. That requires new heuristics to select groups of activities for analysis, as the combinatoric problem is increased even further for this kind of matches.

References

1. Dijkman, R.: A Classification of Differences between Similar Business Processes. In: Proceedings of IEEE EDOC, pp. 37–50 (2007)
2. Dijkman, R.: Diagnosing differences between business process models. In: Dumas, M., Reichert, M., Shan, M.-C. (eds.) BPM 2008. LNCS, vol. 5240, pp. 261–277. Springer, Heidelberg (2008)
3. Rahm, E., Bernstein, P.A.: A survey of approaches to automatic schema matching. VLDB Journal 10(4), 334–350 (2001)
4. Dhamankar, R., Lee, Y., Doana, A., Halevy, A., Domingos, P.: imap: Discovering complex semantic matches between database schemas. In: SIGMOD, pp. 383–394 (2004)
5. Levenshtein, V.I.: Binary codes capable of correcting deletions, insertions, and reversals. Soviet Physics Doklady 10(8), 707–710 (1966)
6. Qu, Y., Hu, W., Cheng, G.: Constructing virtual documents for ontology matching. In: Carr, L., et al. (eds.) WWW, pp. 23–31. ACM, New York (2006)
7. Euzenat, J., Shvaiko, P.: Ontology matching. Springer, Heidelberg (2007)
8. Porter, M.F.: An algorithm for suffix stripping. Program 14(3), 130–137 (1980)
9. Vanhatalo, J., Völzer, H., Koehler, J.: The refined process structure tree. In: Dumas, M., Reichert, M., Shan, M.-C. (eds.) BPM 2008. LNCS, vol. 5240, pp. 100–115. Springer, Heidelberg (2008)
10. Dijkman, R.M., Dumas, M., García-Bañuelos, L.: Graph matching algorithms for business process model similarity search. In: Dayal, U., Eder, J., Koehler, J., Reijers, H.A. (eds.) BPM 2009. LNCS, vol. 5701, pp. 48–63. Springer, Heidelberg (2009)

11. Bunke, H.: On a relation between graph edit distance and maximum common subgraph. *Pattern Recognition Letters* 18(8), 689–694 (1997)
12. Dijkman, R., Dumas, M., García-Bañuelos, L., Käärik, R.: Aligning business process models. In: *Proceedings of IEEE EDOC*, pp. 45–53 (2009)
13. Documentair structuurplan (February 20, 2009), <http://www.model-dsp.nl/>
14. Do, H., Melnik, S., Rahm, E.: Comparison of schema matching evaluations. In: Chaudhri, A.B., Jeckle, M., Rahm, E., Unland, R. (eds.) *NODE-WS 2002*. LNCS, vol. 2593, pp. 221–237. Springer, Heidelberg (2003)
15. van Glabbeek, R.J., Goltz, U.: Refinement of actions and equivalence notions for concurrent systems. *Acta Informatica* 37(4/5), 229–327 (2001)
16. Preuner, G., Conrad, S., Schrefl, M.: View integration of behavior in object-oriented databases. *Data & Knowledge Engineering* 36(2), 153–183 (2001)
17. Basten, T., Aalst, W.: Inheritance of Behavior. *Journal of Logic and Algebraic Programming* 47(2), 47–145 (2001)
18. Grossmann, G., Ren, Y., Schrefl, M., Stumptner, M.: Behavior based integration of composite business processes. In: van der Aalst, W.M.P., Benatallah, B., Casati, F., Curbera, F. (eds.) *BPM 2005*. LNCS, vol. 3649, pp. 186–204. Springer, Heidelberg (2005)
19. Pankratius, V., Stucky, W.: A formal foundation for workflow composition, workflow view definition, and workflow normalization based on petri nets. In: Hartmann, S., Stumptner, M. (eds.) *APCCM*. CRPIT, vol. 43. Austral. Comp. Soc. (2005)
20. Mendling, J., Simon, C.: Business Process Design by View Integration. In: Eder, J., Dustdar, S. (eds.) *BPM Workshops 2006*. LNCS, vol. 4103, pp. 55–64. Springer, Heidelberg (2006)
21. Ehrig, M., Koschmider, A., Oberweis, A.: Measuring similarity between semantic business process models. In: *APCCM*. CRPIT, vol. 67, pp. 71–80. Austral. Comp. Soc. (2007)
22. van Dongen, B.F., Dijkman, R.M., Mendling, J.: Measuring similarity between business process models. In: Bellahsene, Z., Léonard, M. (eds.) *CAiSE 2008*. LNCS, vol. 5074, pp. 450–464. Springer, Heidelberg (2008)
23. Xu, L., Embley, D.W.: Discovering direct and indirect matches for schema elements. In: *DASFAA*, pp. 39–46. IEEE Computer Society, Los Alamitos (2003)
24. He, B., Chang, K.: Automatic complex schema matching across web query interfaces: A correlation mining approach. *ACM Trans. Database Syst.* 31(1), 346–395 (2006)
25. Madhavan, J., Bernstein, P.A., Rahm, E.: Generic schema matching with cupid. In: Apers, P., et al. (eds.) *VLDB*, pp. 49–58 (2001)

Process Compliance Measurement Based on Behavioural Profiles

Matthias Weidlich¹, Artem Polyvyanyy¹, Nimit Desai², and Jan Mendling³

¹ Hasso Plattner Institute at the University of Potsdam, Germany
{Matthias.Weidlich,Artem.Polyvyanyy}@hpi.uni-potsdam.de

² IBM India Research Labs, India
Nimit123@in.ibm.com

³ Humboldt-Universität zu Berlin, Germany
Jan.Mendling@wiwi.hu-berlin.de

Abstract. Process compliance measurement is getting increasing attention in companies due to stricter legal requirements and market pressure for operational excellence. On the other hand, the metrics to quantify process compliance have only been defined recently. A major criticism points to the fact that existing measures appear to be unintuitive. In this paper, we trace back this problem to a more foundational question: which notion of behavioural equivalence is appropriate for discussing compliance? We present a quantification approach based on behavioural profiles, which is a process abstraction mechanism. Behavioural profiles can be regarded as weaker than existing equivalence notions like trace equivalence, and they can be calculated efficiently. As a validation, we present a respective implementation that measures compliance of logs against a normative process model. This implementation is being evaluated in a case study with an international service provider.

1 Introduction

Compliance management is becoming increasingly important. Companies seek for a better control of their processes not only to satisfy new legal requirements but also to leverage cost-saving opportunities through standardization of business operations. Once non-compliant cases of operations are detected, the company can either update its process model to cover the respective case or it can impose new mechanisms to enforce best practice execution. In this way, compliance management is a central piece in the puzzle of advancing a company towards a higher degree of process maturity.

In order to make compliance management work in practice, it is required to gather detailed information on the execution of particular business processes. In recent years, process mining has emerged as a technique that automatically reworks process logs data such that managerial decision making can be supported [1]. Most importantly, certain key measures of compliance management can easily be quantified. Therefore, some compliance metrics have been proposed in recent literature, e.g., by Medeiros et al. [2], and Rozinat and van der Aalst [3].

These metrics check a set of logs against a normative process model to calculate the degree of compliance.

The aforementioned metrics have a few drawbacks. On the one hand, they rely on state-based techniques that involve replaying the logs. They are based on the notion of trace equivalence, which is a weak notion in the linear time – branching time spectrum [4]. As a consequence, these approaches have to cope with the state explosion problem in order to achieve efficient computation [5]. That, in turn, leads to the application of heuristics that have to be tuned for a certain setting. On the other hand, these metrics have been tested by Gerke et al. [6] for their applicability in a case study with a German air carrier. As it turned out, these metrics yielded non-compliance values that are significantly smaller than the degree of non-compliance perceived by the business users. While adaptations to the metrics have been proposed, the conceptual basis in terms of trace equivalence remains unchanged. These results, along with the inherent complexity of state-space based approaches, suggest to choose a different approach for measuring compliance.

In this paper, we approach the problem from the perspective of relations between pairs of activities or log events, respectively, instead of trying to replay logs according to a rather strict notion of equivalence. Thus, our contribution is a foundation of compliance measurement in a notion that is more relaxed than trace equivalence. We utilize behavioural profiles as a base line to calculate novel compliance metrics. As behavioural profiles can be calculated efficiently, we avoid performance issues of existing state-based metrics. We implemented our approach and validated it on an industry case study. In this context, our metrics showed a good approximation of the compliance as perceived by the business users.

Against this background, the paper is structured as follows. Section 2 discusses the challenge of measuring compliance by means of an example and elaborates on existing compliance metrics. Subsequently, Section 3 presents preliminaries for our investigations. Section 4 introduces compliance metrics based on behavioural profiles. Based thereon, Section 5 presents findings from our validation, for which we implemented a prototype and tested it on real-world logs. Section 6 discusses our approach in the light of related work. Finally, Section 7 concludes the paper and identifies topics for future research.

2 Background

In this section, we discuss the challenges of measuring behavioural compliance. Fig. 1 shows the example of a BPMN process model that includes 11 activities, all named with a capital letter. The diamonds define the routing behaviour of the BPMN model. Once *I* and *A* have been executed, there is a choice being made whether the branch including *B* is executed or as an exclusive alternative, the branch leading to the diamond with the plus sign. The latter option leads to a parallel execution of the sequences *C*, *D*, *E* and *F*, *G*, *H*. Finally, these branches are synchronized and control is passed through the merge node (with the X in the diamond) towards the completion of the process after execution of *O*.

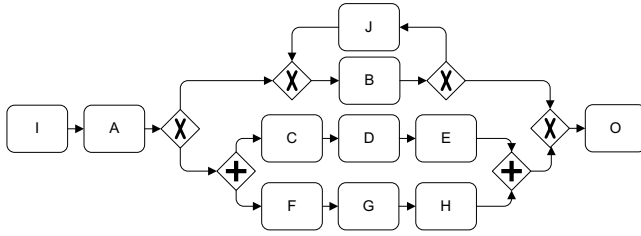


Fig. 1. Example of a BPMN process model

The challenge of compliance measurement is to quantify the degree to which a certain *set of logs* matches a normative process model. Logs (or log files) represent *observed* execution sequences of activities from the normative process model. In the desirable case the logs completely comply with the behaviour defined by the process model. In this case, the logs are *valid* execution sequences. In practice, however, observed execution sequences often deviate from predefined behaviour. This may be the case when the execution order is not explicitly enforced by the information system that records the logs. In fact, it is also possible that people deliberately work around the system [1]. This may result in logs such as the following for the process model in Fig. 1.

- Log $L_1 = I, A, C, D, F, G, E, H, O$
- Log $L_2 = I, A, C, B, E, F, H, O$
- Log $L_3 = I, A, E, D, C, H, G, O, F$
- Log $L_4 = I, C, D, F$

From these four logs, only the first one is also a valid execution sequence, i.e., it can be completely replayed by the process model. Still, the other logs capture a good share of the behaviour defined in the model. Such cases make it necessary to measure compliance *a posteriori*.

Compliance measurement has recently been approached using state-based concepts from the Petri nets theory. The idea of the fitness measure proposed by Medeiros et al. [2] is to replay the log through the model. Activities that are enabled in the model when they appear in the log are counted and related to the overall number of activities. If an activity is not enabled, the respective Petri net transition is forced to fire, which produces a token on each of its output places. In this way, one can quantify compliance of the logs against the process model as a ratio of enabled activities to the total number of activities. For example, in Fig. 1, log L_1 can be completely replayed and has therefore a compliance value of 1. Instead, log L_2 can be replayed solely until B appears in the log. This activity is then fired without being enabled. The same holds for E and H . Therefore, three firings are non-compliant from the eight firings altogether, yielding a fitness value of 0.63. In log L_3 , activities E, D, H , and G are forced to fire although they are not enabled. Thus, altogether, only I, A, C, O , and F are fired correctly from nine tasks. Therefore, the fitness is 0.56. Finally, for

log L_4 , the absence of activity A in the log implies a non-compliant firing of C and D , such that fitness is two out of four, which is 0.5.

As mentioned above, the metrics proposed by Medeiros et al. are based on computationally hard state space exploration, which has to be addressed using heuristics for the general case. In addition, other work reports on these metrics as yielding compliance values which are significantly lower than what is considered to be correct by domain experts [6]. Therefore, for our notion of log compliance, we do not try to replay the log through the model. Instead, we identify different types of constraints that a process model can impose for a pair of activities on the execution order activities, such as *exclusiveness* (B and C in Fig. 1) and *order* (C and E in Fig. 1). Based thereon, the preservation of these constraints is leveraged to assess the compliance of a given log. Besides constraints in terms of execution order, we also take the obligation to execute a single activity in a process model into account. Evidently, activities that are defined as being *mandatory* in the process model have to occur in a log, or can be expected to occur in future if the process has not yet completed. An example for such a mandatory activity in the model in Fig. 1 is activity A . A third group of constraints that we evaluate relates to the *causal coupling* of executions for a pair of activities. That is, the occurrence of one activity implies the occurrence of another activity. For instance, both activities, C and E , are optional for the completion of the example process. Still, the occurrence of C implies the occurrence of E , i.e., their execution is causally coupled.

While these behavioural constraints are not addressed in existing compliance metrics, we show how they can be leveraged in order to assess the compliance of a log in the remainder of this paper.

3 Preliminaries

This section gives preliminaries for our work in terms of a formal framework. For our investigations we use a notion of a process model that is based on a graph containing activity nodes and control nodes, which, in turn, captures the commonalities of process description languages. Thus, the subset of BPMN used in our initial example can be traced back to the following definition of a process model.

Definition 1. (*Process Model*) A process model is a tuple $P = (A, a_i, a_o, C, F, T)$ with

- A as a non-empty set of activity nodes, and C as a set of control nodes, A and C are disjoint,
- $a_i \in A$ as an initial activity, $a_o \in A$ as a final activity,
- $F \subseteq ((A \setminus \{a_o\}) \cup C) \times ((A \setminus \{a_i\}) \cup C)$ as the flow relation, and
- $T : C \mapsto \{\text{and, or, xor}\}$ as a function that assigns a type to control nodes.

In the remainder of this paper, the identity relation for activities is denoted by id_A , i.e., $(a, a) \in id_A$ for all $a \in A$. Further on, we do not formalise the execution semantics of a process model, but assume an interpretation of the

model following on common process description languages, such as BPMN, EPCs, or UML activity diagrams. It is worth to mention that we do not assume a certain definition of semantics for the inclusive OR construct, which raises serious issues in cyclic structures. We solely assume the existence of such a definition. Under the assumption of a definition of execution semantics, the set of all valid execution sequences, as well as the notion of a process log, can be defined as follows.

Definition 2. (*Execution Sequence, Execution Log*) *The set of execution sequences \mathcal{E}_P for a process model $P = (A, a_i, a_o, C, F, T)$ is the set of all lists of the form $\{a_i\} \cdot A^* \cdot \{a_o\}$, that can be created following on the execution semantics of P . An execution log L_P that has been observed for P is a non-empty list of the form A^* .*

Note that we speak of an *execution sequence* of a process model, solely in case the sequence is valid regarding the process model, i.e., it can be completely replayed by the model. In contrast, a *log* is a non-empty sequence over the activities of a process model. As a short-hand notation, we use $A_L \subseteq A$ to refer to the subset of activities of a process model that is contained in log L .

In order to capture the constraints imposed by a process model on the order of activity execution, we rely on the concept of a behavioural profile [7]. Behavioural profile defines relations for all pairs of activities of a process model. These relations, in turn, might be interpreted as the essential behavioural characteristics specified by the models. All behavioural relations are based on the notion of *weak order*. That is, two activities are in weak order, if there exists an execution sequence in which one activity occurs after the other. Here, only the existence of an execution sequence is required.

Definition 3 (Weak Order (Process Model)). *Let $P = (A, a_i, a_o, C, F, T)$ be a process model and \mathcal{E}_P its set of execution sequences. The weak order relation $\succ_P \subseteq (A \times A)$ contains all pairs (x, y) , such that there is an execution sequence $\sigma = n_1, \dots, n_m$ in \mathcal{E}_P with $j \in \{1, \dots, m - 1\}$ and $j < k \leq m$ for which holds $n_j = x$ and $n_k = y$.*

Based thereon, we define the relations of the behavioural profile for pairs of activities. Each pair can be related by weak order in three different ways.

Definition 4 (Behavioural Profile (Process Model)). *Let $P = (A, a_i, a_o, C, F, T)$ be a process model. A pair $(x, y) \in (A \times A)$ is in at most one of the following relations:*

- *The strict order relation \rightsquigarrow_P , iff $x \succ_P y$ and $y \not\succeq_P x$.*
- *The exclusiveness relation $+_P$, iff $x \not\succeq_P y$ and $y \not\succeq_P x$.*
- *The interleaving order relation \parallel_P , iff $x \succ_P y$ and $y \succ_P x$.*

The set $\mathcal{B}_P = \{\rightsquigarrow_P, +_P, \parallel_P\}$ is the behavioural profile of P .

Note that we say that a pair (x, y) is in *reverse strict order*, denoted by $x \rightsquigarrow_P^{-1} y$, if and only if $y \rightsquigarrow_P x$. Interleaving order is also referred to as observation concurrency. Further on, the relations of the behavioural profile along with reverse

strict order partition the Cartesian product of activities [7]. We illustrate the relations of the behavioural profile by means of our example model in Fig. 1. Here, for instance, it holds $I \rightsquigarrow D$. Evidently, strict order does not imply the actual occurrence, i.e., activity D might not be executed. Further on, $B + C$ as both activities will never occur in a single valid execution sequence of the model, and $C || G$ as C might occur before G and vice versa. Note that it holds $B || J$ due to the control flow cycle. That is, an occurrence of J is followed by an occurrence of B . Further on, an activity is either said to be *exclusive to itself* (e.g., $I + I$) or *in interleaving order to itself* (e.g., $B || B$). The former holds, when an activity cannot be repeated, whereas the latter implies that the activity is part of a control flow cycle.

The concept of a behavioural profile relates pairs of activities according to their *order* of potential occurrence, whereas further behavioural characteristics are not considered. Both, the fact that an activity is *mandatory* for process completion and *causality* between activities are not covered. Causality, in turn, involves two orthogonal aspects, i.e., the *order* of activity occurrences and their *causal coupling* (the occurrence of one activity enforces the occurrence of another activity). While the former is already addressed by the behavioural profile in terms of the (reverse) strict order relation, the latter is not captured. In order to cope with these aspects, the behavioural profile is extended by a fourth relation, yielding the causal behavioural profile.

Definition 5 (Causal Behavioural Profile (Process Model))

Let $P = (A, a_i, a_o, C, F, T)$ be a process model.

- A pair $(x, y) \in (A \times A)$ is in the co-occurrence relation \gg_P , iff for all execution sequences $\sigma = n_1, \dots, n_m$ in \mathcal{E}_P it holds $n_i = x$ with $1 \leq i \leq m$ implies that there is an index $j \in \{1, \dots, m\}$, such that $n_j = y$.
- The set $\mathcal{B}_P^+ = \mathcal{B}_P \cup \{\gg_P\}$ is the causal behavioural profile of P .

Given a process model $P = (A, a_i, a_o, C, F, T)$, the set of mandatory activities $A_M \subseteq A$ is given by all activities that are co-occurring with the initial activity, that is, $A_M = \{a \in A \mid a_i \gg_P a\}$. Moreover, causality holds between two activities $a_1, a_2 \in A$, if they are in strict order, $a_1 \rightsquigarrow_P a_2$ and the occurrence of the first implies the occurrence of the second, $a_1 \gg_P a_2$. Again, we refer to the example in Fig. 1 for illustration purposes. In this model, there are only three mandatory activity, namely $A_M = \{I, A, O\}$. Moreover, it holds $C \gg E$ and $E \gg C$. Thus, all complete execution sequences of the process model that contain activity C are required to also contain activity E , and vice versa. In addition, the model specifies a strict order relation between both activities, $C \rightsquigarrow E$, such that we speak of a causal dependency from C to E .

4 Measuring Compliance Based on Behavioural Profiles

This section introduces compliance metrics based on behavioural profiles. First, Section 4.1 shows how the concepts introduced in the previous section can be lifted to logs. Second, we elaborate on a hierarchy between the relations of the

behavioural profile in Section 4.2. Based thereon, we introduce the compliance metrics in Section 4.3.

4.1 Causal Behavioural Profiles for Logs

In order to lift the concept of behavioural profiles to logs, first and foremost, we have to clarify the notion of weak order for logs. Following on the definition given for process models, two activities are in weak order in a log, if the first occurs before the second.

Definition 6 (Weak Order (Log)). *Let $L_P = n_1, \dots, n_m$ be a log of a process model $P = (A, a_i, a_o, C, F, T)$. The weak order relation $\succ_L \subseteq (A_L \times A_L)$ contains all pairs (x, y) , such that there exists two indices $j, k \in \{1, \dots, m - 1\}$ with $j < k \leq m$ for which holds $n_j = x$ and $n_k = y$.*

Based thereon, we define the behavioural profile of a log.

Definition 7 (Behavioural Profile (Log)). *Let $L_P = n_1, \dots, n_m$ be a log of a process model $P = (A, a_i, a_o, C, F, T)$. A pair $(x, y) \in (A_L \times A_L)$ is in at most one of the following relations:*

- The strict order relation \rightsquigarrow_L , iff $x \succ_L y$ and $y \not\prec_L x$.
- The interleaving order relation \parallel_L , iff $x \succ_L y$ and $y \succ_L x$.

The set $\mathcal{B}_L = \{\rightsquigarrow_L, \parallel_L\}$ is the behavioural profile of L .

Again, the pair (x, y) is in *reverse strict order*, denoted by $x \rightsquigarrow_L^{-1} y$, if and only if $y \rightsquigarrow_L x$. The relations of the behavioural profile along with reverse strict order partition the Cartesian product of elements of a log. For the exemplary log $L_3 = I, A, E, D, C, H, G, O, F$, for instance, it holds $C \rightsquigarrow F$ and $D \rightsquigarrow^{-1} A$.

It is worth to mention that there are fundamental differences when interpreting the behavioural profile for a process model and a log. On the one hand, in contrast to the profile of a process model, we cannot observe exclusiveness between two activities in a single log. On the other hand, activities that might be enabled concurrently in a process model (e.g., C and G in our example) are related by interleaving order in the behavioural profile of the model ($C \parallel G$). However, if both activities are not part of a control-flow cycle, they might be related by strict order or reverse strict order in the profile of a corresponding log. For instance, for the log $L_3 = I, A, E, D, C, H, G, O, F$, we observe $C \rightsquigarrow G$ as the behavioural relation for activities C and G .

Moreover, we can also lift the definition of a causal behavioural profile to process logs. Obviously, all elements of a log are co-occurring.

Definition 8 (Causal Behavioural Profile (Log)). *Let $L_P = n_1, \dots, n_m$ be a log of a process model $P = (A, a_i, a_o, C, F, T)$*

- The co-occurrence relation $\gg_L = (A_L \times A_L)$ contains all pairs of log elements.
- The set $\mathcal{B}_L^+ = \mathcal{B}_L \cup \{\gg_L\}$ is the causal behavioural profile of L .

4.2 A Hierarchy of Behavioural Relations

As mentioned above, there is a fundamental difference between behavioural profiles of process models and of logs. The former defines relations based on the set of *all* possible execution sequences, whereas the latter considers only one observed execution sequence as defined by the log. In order to cope with this phenomenon, we introduce a hierarchy between the relations of behavioural profiles (we neglect the co-occurrence relation at this stage). The idea is to order the behavioural relations based on their *strictness*. We consider the exclusiveness relation as the strongest relation, as it completely disallows two activities to occur together in an execution sequence. In contrast, the interleaving order relation can be seen as the weakest relation. It allows two activities to occur in any order in an execution sequence. Consequently, the strict order and reverse strict order relation are intermediate relations, as they disallow solely a certain order of two activities. We formalize this hierarchy between behavioural relations as a *subsumption predicate*. Given two behavioural relations between a pair of two activities, this predicate is satisfied, if and only if the first relation is equal or weaker than the second.

Definition 9 (Subsumption Predicate). *Given two behavioural relations $R, R' \in \{\rightsquigarrow, \rightsquigarrow^{-1}, +, \|\}$ of the same or different behavioural profiles, the subsumption predicate $\mathcal{S}(R, R')$ is satisfied, iff $(R \in \{\rightsquigarrow, \rightsquigarrow^{-1}\} \wedge R' = +)$ or $R = R'$ or $R = \|\$.*

Again, we illustrate this concept using the example model in Fig. 1 and the log $L_3 = I, A, E, D, C, H, G, O, F$. As mentioned above, for activities C and G , it holds $C\|G$ in the profile of the process model and $C \rightsquigarrow G$ in the profile of the log. The former specifies that C and G might occur in any order in an execution sequence, owing to the interleaving semantics of activities that are enabled concurrently. The latter, in turn, captures the fact that the occurrences of C and G in the log are ordered. However, we see that there is a subsumption relation between both relations, as $\mathcal{S}(\|, \rightsquigarrow)$ is satisfied. Evidently, this information has to be taken into account when assessing compliance of logs. This stems from the fact that logs do not hint at potential interleaving execution of activities.

4.3 Compliance Metrics

For our measurements of compliance between a process model and a log, we consider three aspects separately, namely *execution order*, *mandatory activities*, and *casual coupling*. While the last two aspects address the question *what* should be contained in the log (activities that are mandatory in a global sense or that are implied by the occurrence of other activities in the log), the first aspect clarifies *how* these activities should be ordered in the log. While each of these aspects is assessed by a separate compliance degree, their values may be aggregated into a single compliance degree.

Execution order compliance is motivated by the fact that the order of execution of activities as specified by the log should be in line with the ordering

constraints as imposed by the process model. Evidently, the question whether there is such a difference is of limited usefulness. Instead, any deviation has to be quantified to allow for thorough compliance analysis. We achieve such a quantification based on the notion of behavioural profiles and the hierarchy of behavioural relations. That is, we analyse the Cartesian product of activities that are contained in a log and determine, whether the behavioural relation for a pair of activities in the log is subsumed by the relation specified in the process model. Based thereon, the degree of behavioural compliance is defined as a ratio of consistent behavioural relations relative to the number of activity pairings in the log. Note that for the case of a behavioural relation that holds between an activity and itself, the subsumption predicate cannot be applied. If an activity can occur at most once in the process model (it is exclusive to itself), it is not allowed to occur multiple times in the log (it cannot be in interleaving order to itself), whereas the opposite case (exclusive in the log, in interleaving order in the model), as well as equal relations, are considered to be compliant. Based thereon, we define execution order compliance.

Definition 10 (Execution Order Compliance). *Let $L_P = n_1, \dots, n_m$ be a log of a process model $P = (A, a_i, a_o, C, F, T)$. Then, the set $SR \subseteq (A_L \times A_L)$ contains all pairs of activities (x, y) , for which the behavioural relation in L_P is subsumed by the relation in P , i.e., $\forall R \in (\mathcal{B}_P \cup \{\rightsquigarrow_P^{-1}\}), R' \in (\mathcal{B}_L \cup \{\rightsquigarrow_L^{-1}\}) [(xRy \wedge xR'y) \Rightarrow (S(R, R') \vee (x = y \wedge R = ||_P \wedge R' = +_L))]$. The degree of execution order compliance of L_P to P is defined as*

$$\mathcal{E}C_{L_P} = \frac{|SR|}{|(A_L \times A_L)|}.$$

Note that the compliance degree $\mathcal{E}C_{L_P}$ for a log is between zero, i.e., no execution order compliance at all, and one indicating full execution order compliance. It is worth to mention that this degree is independent of the size of the process model, as solely the activity pairs found in the log are considered in the computation. Given the log $L_3 = I, A, E, D, C, H, G, O, F$ and our initial example (cf., Fig. 1), we see that various order constraints imposed by the model are not satisfied. For instance, $D \rightsquigarrow E$ and $G \rightsquigarrow H$ are specified in the model, whereas we have $D \rightsquigarrow^{-1} E$ and $G \rightsquigarrow^{-1} H$ in the profile of the log. That, in turn, leads to an execution order compliance degree of $\mathcal{E}C = \frac{67}{81} \approx 0.83$ for this particular log.

Due to the fact that execution order compliance relates solely to the ordering constraints, we embrace further process characteristics for compliance measurements. In particular, we take the mandatory activities that are required for completion of the process into account. The ratio of mandatory activities of the process model that are contained in the log, and all mandatory activities would be a straight-forward measure for this aspect. However, we want to consider also logs that might not have completed yet, such that missing mandatory activities might be added later on. In order to cope with this, not all mandatory activities of the process model are required to occur in the log. Instead, we consider only those mandatory activities, for which we can deduce from the log that they

should have been observed already. That is, a mandatory activity is considered, if it is either in the log, or it is in strict order with one of the activities in the log.

Definition 11 (Mandatory Execution Compliance). *Let $L_P = n_1, \dots, n_m$ be a log of a process model $P = (A, a_i, a_o, C, F, T)$ and $A_M = \{a \in A \mid a_i \gg_P a\}$ the set of mandatory activities of P . Then, the set $EA_M \subseteq A_M$ contains all mandatory activities a_1 that are in the log or can be expected to be in the log, i.e., $a_1 \in A_L$ or $\exists a_2 \in A_L [a_1 \neq a_2 \wedge a_1 \rightsquigarrow_P a_2]$. The degree of mandatory execution compliance of L_P to P is defined as*

$$\mathcal{MC}_{L_P} = \begin{cases} 1 & \text{if } EA_M = \emptyset, \\ \frac{|A_L \cap A_M|}{|EA_M|} & \text{else.} \end{cases}$$

Consider the log $L_4 = I, C, D, F$ of our initial example. The process model in Fig. 1 specifies three mandatory activities, i.e., activities I , A , and O . While activity I is in the log, activity A is not, although we know that it should have been observed already owing to the strict order relation between A and C , D , and E , respectively. The mandatory activity O is not required to occur in the log, as the log does not contain any activity that is in strict order with O . Therefore, the log contains one out of two expected mandatory activities, such that the execution compliance degree is $\mathcal{MC} = 0.5$. Note that the mandatory execution compliance degree might be overestimated. That is, a mandatory activity might be part of a control flow cycle, while the log does not contain any activity that is in strict order with the mandatory activity. In this case, the mandatory activity will not be considered in the mandatory execution compliance degree. This holds, even though the log might already contain activities that can only be reached after executing the mandatory activity in the process model.

The mandatory execution compliance relates to activities that should be observed in each log as they are required for completion of the process. Still, there might be dependencies between the occurrences of activities even though they are not mandatory. In the model in Fig. 1, for instance, neither activity C nor E are mandatory for completion. However, there is a causal relation between both, as the occurrence of C implies the occurrence of E . While the correctness of the order of occurrence for both activities has already been incorporated in the notion of execution order compliance, the causal coupling of their occurrences has still to be taken into account. Therefore, for each activity a in the log, we also check whether activities for which the occurrence is implied by a are also in the log. Again, we have to consider the eventuality of incomplete logs, such that this requirement must hold only in case there is sufficient evidence that the implication is not satisfied. That is, there is another activity in the log for which the model specifies strict order to the activity under investigation.

Definition 12 (Causal Coupling Compliance). *Let $L_P = n_1, \dots, n_m$ be a log of a process model $P = (A, a_i, a_o, C, F, T)$. Then, the set $EC_M \subseteq (\gg_P \setminus id_A)$ contains all pairs of co-occurring activities (a_1, a_2) , such that the first activity is in the log, while the second activity is in the log or can be expected to be in the*

Table 1. Compliance results for the logs of the initial example (cf., Section 2)

| Log | \mathcal{EC} | \mathcal{MC} | \mathcal{CC} | \mathcal{C} |
|-----------------------------------|-----------------|---------------------|-----------------|--------------------|
| | Execution Order | Mandatory Execution | Causal Coupling | Overall Compliance |
| $L_1 = I, A, C, D, F, G, E, H, O$ | 1.00 | 1.00 | 1.00 | 1.00 |
| $L_2 = I, A, C, B, E, F, H, O$ | 0.88 | 1.00 | 0.80 | 0.89 |
| $L_3 = I, A, E, D, C, H, G, O, F$ | 0.83 | 1.00 | 1.00 | 0.94 |
| $L_4 = I, C, D, F$ | 1.00 | 0.50 | 0.69 | 0.73 |

log, i.e., $a_1 \in A_L$ and $a_2 \in A_L \vee \exists a_3 \in A_L [a_2 \rightsquigarrow_P a_3]$. The degree of causal coupling compliance of L_P to P is defined as

$$\mathcal{CC}_{L_P} = \begin{cases} 1 & \text{if } EC_M = \emptyset, \\ \frac{|(A_L \times A_L) \setminus id_{A_L} \cap \gg_P|}{|EC_M|} & \text{else.} \end{cases}$$

We illustrate causal coupling compliance using our initial example and the log $L_2 = I, A, C, B, E, F, H, O$. Although all mandatory activities are contained in the log, we see that, for instance, $C \gg D$ is not satisfied. This is penalised as the log contains E and it holds $D \rightsquigarrow E$ in the process model. In other words, the occurrence of E in the log provides us with evidence that we should have observed D , too. Thus, the absence of this activity impacts on the causal coupling compliance. Note that the same holds true for all co-occurrence relations involving activity G , such that the degree of causal coupling compliance is $\mathcal{CC} = \frac{33}{41} \approx 0.80$ for this particular log. It is worth to mention that the causal coupling compliance degree might be *overestimated*. An example for this phenomenon would be the log I, A, J for the model in Fig. 1. There is a causal coupling $J \gg B$ in this model. However, the absence of B would not be penalised as there is no activity in the log that is in strict order for B and, therefore, would provide us with sufficient evidence that B should have been observed already.

Given compliance degrees for the three aspects, *execution order*, *mandatory activities*, and *casual coupling*, we can aggregate the compliance degree of a log.

Definition 13 (Log Compliance). Let $L_P = n_1, \dots, n_m$ be a log of a process model $P = (A, a_i, a_o, C, F, T)$. The compliance of L_P to P is defined as

$$\mathcal{C}_{L_P} = \frac{1}{3}(\mathcal{EC}_{L_P} + \mathcal{MC}_{L_P} + \mathcal{CC}_{L_P}).$$

Applied to our example process model and the four exemplary logs introduced in Section 2, our compliance metrics yield the results illustrated in Table 1. As expected, the first log L_1 , which represents a valid execution sequence of the process model satisfies all constraints, such that our metric indicates full overall compliance. In contrast, the execution order constraints are not fully satisfied in the second log L_2 (e.g., the exclusiveness in the model between B and C is broken in the log). In addition, the causal coupling is not completely respected in the

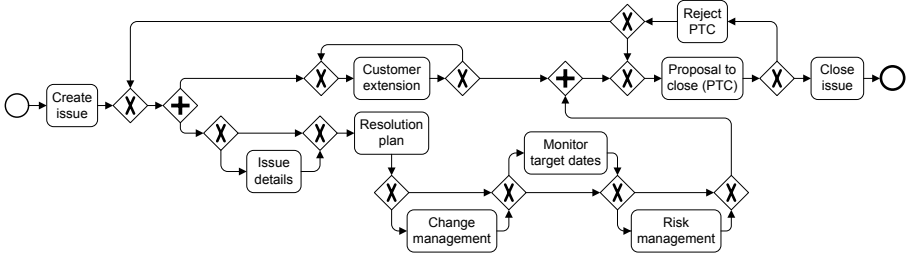


Fig. 2. BPMN model of the Security Incident Management Process (SIMP)

log either, as discussed above. Further on, for the log L_3 , the execution order is not completely in line with the process model. Regarding the log L_4 , we already discussed the absence of a mandatory activity that should have been observed, leading to a mandatory execution compliance degree below one. Note that this also impacts on the causal coupling compliance. Thus, a mandatory execution compliance degree smaller than one, will also be reflected in the causal coupling compliance degree. However, the opposite does not hold true, as illustrated for the log L_2 . Although this log shows full mandatory execution compliance, its causal coupling compliance degree is below one.

5 Case Study: Security Incident Management Process

To demonstrate and evaluate our approach, we apply it to the Security Incident Management Process (SIMP) — which is an issue management process used in global service delivery centres. The process and the logs have been minimally modified to remove confidential information. Fig. 2 shows the BPMN model of SIMP solicited from domain experts.

SIMP is used in one of IBM’s global service delivery centres that provides infrastructure management and technical support to customers. When a customer reports a problem or requests a change, an *issue* is created, spawning a new instance of the process. Details about the issue *may* be updated, a plan to resolve the issue *must* be created, and change management related activities *may* be performed if required. Then, target dates for issue resolution *may* be monitored and relevant risks *may* be documented. A *Customer Extension* of target dates may be processed during any of the above activities (parallel path). Once the steps for resolution are taken and verified, the resolver *must* propose to close the issue. Based on the evidence that the issue is indeed resolved, the issue creator *may* close the issue. Otherwise, the proposal *must* be rejected.

For the SIMP process we analysed 852 logs, each consisting of a set of log entries. Such a log entry has an activity name, activity description, and the time-stamp marking the time of execution of the activity. Although the process is standardized and documented, it is not orchestrated via workflow tools in the IBM’s global service delivery centre under investigation. Instead, it is manually

Table 2. Compliance results for the SIMP process derived from 852 logs

| | \mathcal{EC} | \mathcal{MC} | \mathcal{CC} | \mathcal{C} |
|-------------------------|--------------------|------------------------|--------------------|-----------------------|
| | Execution Order | Mandatory Execution | Causal Coupling | Overall Compliance |
| Average Value | 0.99 | 0.96 | 0.95 | 0.96 |
| Min Value | 0.31 | 0.60 | 0.50 | 0.61 |
| Max Value | 1.00 | 1.00 | 1.00 | 1.00 |
| Logs with Value of 1.00 | 78.64% | 84.15% | 84.15% | 76.17% |

carried out. Hence, the employees are free to deviate from the process. As a result, the logs may or may not specify valid execution sequences of the process model.

For each log, we analysed its compliance using the metrics proposed in this paper. Table 2 gives a summary of this analysis in terms of the average compliance value of all logs, the observed minimal and maximal compliance values, and the share of fully compliant logs. The compliance values were discussed with the manager of the process. The average values reflect the manager’s perception that SIMP is running satisfactory and most cases are handled in a compliant way. As the minimum values show, it was also possible to identify cases of low compliance.

We are not able to directly compare our results with the approach proposed in [2]. This is mainly due to the inherent complexity of the state space exploration, which is exponential in the general case. Even a maximally reduced Petri net of our SIMP process contains a lot of silent steps owing to several activities, for which execution is optional. That, in turn, leads to a significant increase of the state space to investigate when trying to replay a log. While compliance values might still be derived following the most greedy strategy, these results are of a limited validity as they highly underestimate the degree of compliance. However, it is worth to mention that even with the most greedy strategy, computation of the compliance values for all logs took around 15 seconds. In contrast, computing the compliance metrics proposed in this paper for all logs, in turn, could be done within milliseconds. Moreover, an isolated analysis of a sample of 30 logs for which computation of the fitness metric is possible with a 5-step-ahead strategy revealed that the fitness compliance values are all lower than the values derived by our metrics. This is in line with the criticism of [6] that the fitness concept appears to be too strict. These differences probably stem from the different normalisation of any behavioural deviation. The fitness metric relates tokens additionally created (removed) to the number of all consumed (produced) tokens when replaying a log. In contrast, our approach considers violations on the level of relations between activities. Therefore, a single violation according to our notion, can be reflected multiple times in the fitness metric.

The adaptations of [6] could not be compared either to our values since the respective implementation is not publicly available. But as they are also defined on state concepts, we can assume results similar to those obtained above.

6 Related Work

In this section we discuss the relation of our work to compliance measurement, behavioural equivalence, and process similarity.

Compliance measures are at the core of process mining. Similar relations, but not exactly those of behavioural profiles, are used in [8] to characterise a process as a pre-processing step for deriving a model. As mentioned before, the work in [2,3] proposes a fitness measure for process mining, while, based thereon, adaptations are discussed in [6]. In this paper, we demonstrated that our approach benefited from the efficient calculation of the behavioural profiles from free-choice process models as defined in [7]. In fact, behavioural profiles can be derived in $O(n^3)$ time with n as the number of activities of a process model. Therefore, in contrast to the fitness calculation, our metrics can be computed within milliseconds. Further on, following on the criticism of [6], our case study provided us with evidence that our metrics are close to managers' perception of compliance. The detection of differences between process models, not their measurement, is also discussed in related work. The approaches presented in [9,10] provides a systematic framework of diagnosis and resolution of such mismatches.

The concept of behavioural profile in general relates to different notions of *behavioural equivalence* such as trace equivalence and bisimulation. These notions build on state concepts and can often not be calculated as efficiently as behavioural profiles. They also yield only a true or false answer and they are not directly applicable to execution sequences [11,12]. *Behaviour inheritance* is closely related to these notions. Basten et al. [13] define protocol inheritance and projection inheritance based on labelled transition systems and branching bisimulation. A model inherits the behaviour of a parent model, if it shows the same external behaviour when all actions that are not part of the parent model are either *blocked* (protocol inheritance) or *hidden* (projection inheritance). Similar ideas have been presented in [14,15]. The boolean characteristics of these notions have been criticized as inadequate for many process measurement scenarios [2].

The question of *process similarity* has been addressed from various angles. Focussing on behavioural aspects, [16,17] introduce similarity measures based on an edit distance between workflows. Such an edit distance might be based on the language of the workflow, the underlying automaton, or based on the n-gram representation of the language. A similar approach is also taken in [18], in which the authors measure similarity based on high-level change operations that are needed to transform one model into another. Close to our behavioural abstraction of a behavioural profile are causal footprints as introduced in [19]. The authors also show how the footprints can be leveraged to determine the similarity between process models. All these similarity notions are either expensive in terms of calculation, whereas behavioural profiles can be calculated efficiently.

7 Conclusion

In this paper, we have discussed the challenges of providing compliance measurements in an appropriate and efficient way. Our contribution is a novel proposal

for metrics based on behavioural constraints on pairs of tasks for compliance measurement. In this way, we avoided performance problems of state-based metrics by using behavioural profiles as the underlying equivalence notion. Our log compliance metric covers three aspects including execution order, mandatory execution, and causal coupling, which can all be derived from the causal behavioural profile. We implemented these metrics and tested them in a case study.

In future work, we aim to study the merits of our novel approach in further industry collaborations. The performance gain of using behavioural profiles is of serious importance for various use cases. Up until now, compliance measurement had to be conducted offline in a batch mode due to being very time consuming. We aim to investigate those scenarios where an instantaneous compliance measurement is valuable. In particular, compliance measurement in the financial industry might eventually benefit from this innovation, e.g., to cancel running transactions that exhibit non-compliant behaviour.

References

1. van der Aalst, W., Reijers, H., Weijters, A., van Dongen, B., de Medeiros, A., Song, M., Verbeek, H.: Business process mining: An industrial application. *Inf. Syst.* 32(5), 713–732 (2007)
2. de Medeiros, A., van der Aalst, W., Weijters, A.: Quantifying process equivalence based on observed behavior. *Data Knowl. Eng.* 64(1), 55–74 (2008)
3. Rozinat, A., van der Aalst, W.M.P.: Conformance checking of processes based on monitoring real behavior. *Inf. Syst.* 33(1), 64–95 (2008)
4. Glabbeek, R.: The Linear Time – Branching Time Spectrum I. The semantics of concrete, sequential processes. In: *Handbook of Process Algebra*. Elsevier, Amsterdam (2001)
5. Valmari, A.: The state explosion problem. In: Reisig, W., Rozenberg, G. (eds.) *APN 1998*. LNCS, vol. 1491, pp. 429–528. Springer, Heidelberg (1998)
6. Gerke, K., Cardoso, J., Claus, A.: Measuring the compliance of processes with reference models. In: Meersman, R., Dillon, T., Herrero, P. (eds.) *OTM 2009, Part I*, LNCS, vol. 5870, pp. 76–93. Springer, Heidelberg (2009)
7. Weidlich, M., Mendling, J., Weske, M.: Computation of behavioural profiles of process models. Technical report, Hasso Plattner Institute (June 2009)
8. Aalst, W., Weijters, A., Maruster, L.: Workflow mining: Discovering process models from event logs. *IEEE Trans. Knowl. Data Eng.* 16(9), 1128–1142 (2004)
9. Küster, J.M., Gerth, C., Förster, A., Engels, G.: Detecting and resolving process model differences in the absence of a change log. In: [20], pp. 244–260
10. Dijkman, R.M.: Diagnosing differences between business process models. In: [20], pp. 261–277
11. Glabbeek, R., Goltz, U.: Refinement of actions and equivalence notions for concurrent systems. *Acta Inf.* 37(4/5), 229–327 (2001)
12. Hidders, J., Dumas, M., Aalst, W., Hofstede, A., Verelst, J.: When are two workflows the same? In: *CATS. CRPIT*, vol. 41, pp. 3–11 (2005)
13. Basten, T., Aalst, W.: Inheritance of behavior. *JLAP* 47(2), 47–145 (2001)
14. Ebert, J., Engels, G.: Observable or Invocable Behaviour - You Have to Choose. Technical Report 94-38, Leiden University (December 1994)

15. Schrefl, M., Stumptner, M.: Behavior-consistent specialization of object life cycles. *ACM Trans. Softw. Eng. Methodol.* 11(1), 92–148 (2002)
16. Wombacher, A.: Evaluation of technical measures for workflow similarity based on a pilot study. In: Meersman, R., Tari, Z. (eds.) *OTM 2006, Part I, LNCS*, vol. 4275, pp. 255–272. Springer, Heidelberg (2006)
17. Wombacher, A., Rozie, M.: Evaluation of workflow similarity measures in service discovery. In: *Service Oriented Electronic Commerce. LNI*, vol. 80, pp. 51–71 (2006)
18. Li, C., Reichert, M., Wombacher, A.: On measuring process model similarity based on high-level change operations. In: Li, Q., Spaccapietra, S., Yu, E., Olivé, A. (eds.) *ER 2008. LNCS*, vol. 5231, pp. 248–264. Springer, Heidelberg (2008)
19. Dongen, B., Dijkman, R.M., Mendling, J.: Measuring similarity between business process models. In: Bellahsene, Z., Léonard, M. (eds.) *CAiSE 2008. LNCS*, vol. 5074, pp. 450–464. Springer, Heidelberg (2008)
20. Dumas, M., Reichert, M., Shan, M.-C. (eds.): *BPM 2008. LNCS*, vol. 5240. Springer, Heidelberg (2008)

Business Trend Analysis by Simulation

Helen Schonenberg, Jingxian Jian, Natalia Sidorova, and Wil van der Aalst

Eindhoven University of Technology,
Department of Mathematics & Computer Science,
Den Dolech 2, 5600 MB Eindhoven, The Netherlands
{m.h.schonenberg,n.sidorova,w.m.p.v.d.aalst}@tue.nl

Abstract. Business processes are constantly affected by the environment in which they execute. The environment can change due to seasonal and financial trends. For organisations it is crucial to understand their processes and to be able to estimate the effects of these trends on the processes. Business process simulation is a way to investigate the performance of a business process and to analyse the process response to injected trends. However, existing simulation approaches assume a steady state situation. Until now correlations and dependencies in the process have not been considered in simulation models, which can lead to wrong estimations of the performance. In this work we define an adaptive simulation model with a history-dependent mechanism that can be used to propagate changes in the environment through the model. In addition we focus on the detection of dependencies in the process based on the executions of the past. We demonstrate the application of adaptive simulation models by means of an experiment.

1 Introduction

Business processes are often the result of a fit between the needs and capabilities of the internal stakeholders of the business and the opportunities and threats the business identifies in its environment [14]. The environment in which these processes operate is typically unstable and business processes should be robust enough to cope with a variable and changing environment. The behaviour of the environment can be subject to seasonal or financial trends, such as customers not booking expensive holidays due to financial crisis. The most interesting question from the business point of view is: “*How will these trends affect the performance of my business process?*”.

This paper aims at detecting dependencies in the business process that can be used to accurately analyse the effects of environmental trends on the business process performance. Nowadays, most business processes are supported by information systems that store information about the process execution in logs. We can use this historical information to estimate the effect of trends on business process performance and to help organisations with obtaining insight in questions such as: “*Do we have enough resources available to execute the process during the holiday season?*”.

Real-life business processes usually contain many execution alternatives due to choices, parallelism, iterations and (data) dependencies in the process. For example the choice the environment makes in some part of the process might be correlated with a choice that the environment made earlier in the process, i.e. a customer who books an expensive mean of transportation is more likely to book more expensive hotels. Performance analysis of such systems by using analytical models is often intractable and simulation is used instead. For accurate simulation, i.e. simulation that is close to reality, it is of crucial importance to capture the real behaviour of the process. It is not sufficient to only include the actual execution alternatives in the simulation model to analyse performance, also information about decisions, costs, resources and stochastic aspects of the behaviour need to be included [17].

In most simulation tools for business process management, simulation parameters, like activity cost, duration and probability, are variables that are assumed to be independent, which is often not the case. Incorrect assumptions about correlations and dependencies can lead to over or underestimation of the outcome [9,18,12]. In this paper we show how dependencies are mined from a log of a business process and how they are incorporated into the simulation model. Our approach is to create an adaptive simulation model with parameters that adapt according to the information obtained by the simulation steps executed so far (history-dependency), based on the dependencies found in the log.

An adaptive simulation model is created from two components. The first component is the control flow model that can be either given (predefined models) or mined from the log using standard process mining techniques [2]. The second component consists of information about the simulation parameters. Again, the parameters can be predefined or the log can be used to determine the parameters. In this paper we focus on the latter case, where we consider the simulation parameters as random variables that we are going to estimate on the log. Both components are integrated into an adaptive simulation model by a history-dependent mechanism, that, for each instance, estimates the parameters on the (partial) simulation trace of that instance.

The outline of the paper is as follows. First we give some preliminaries in Section 2. In Section 3 we describe the adaptive simulation model. In practice abstractions will be needed to detect dependencies and Section 4 gives an overview of some elementary abstractions and shows how they can be combined. Section 5 illustrates the use of adaptive simulation models in the experimental setting. Related work will be presented in Section 6. Finally, we conclude the paper in Section 7.

2 Preliminaries

\mathbb{N} denotes the set of natural numbers. A bag (or multiset) over some set S is a mapping $\mathcal{B} : S \rightarrow \mathbb{N}$ that maps each element to the number of times it occurs. The size of a bag $|\mathcal{B}|$ is the total number of elements in the bag including duplicates. $[a^2, b^4, c]$ represents the bag with $\mathcal{B}(a) = 2, \mathcal{B}(b) = 4, \mathcal{B}(c) = 1$ and

$\mathcal{B}(d) = 0$, where $a, b, c, d \in S$ and $[[a^2, b^4, c]] = 7$. The bag where all elements occur exactly once corresponds to a set. A sequence of length n over elements of set S is a mapping $\sigma \in \{1, \dots, n\} \rightarrow S$. We denote the empty sequence by ϵ and non-empty sequences by listing their elements, e.g. $\sigma = \langle e_1, e_2, \dots, e_n \rangle$, where $e_i = \sigma(i)$ for $1 \leq i \leq n$. The size of a sequence $|\sigma|$ corresponds to the length of the sequence. $\sigma \uparrow S$ is the projection of σ onto elements of set S , e.g. $\langle a, b, e, a, b, c \rangle \uparrow \{a, c\} = \langle a, a, c \rangle$. The set of all sequences over S is denoted as S^* , the set of all sets over S as 2^S and the set of all bags over S as \mathbb{N}^S . The Parikh vector $parikh(\sigma) \in \mathcal{B}(S)$ denotes the number of occurrences of element s in a sequence σ , i.e. for $s \in S$: $parikh(\sigma)(s) = |\sigma \uparrow \{s\}|$. The $set : S^* \rightarrow 2^S$ is a function that transforms a sequence over S to a set of S , i.e. $set(\sigma) = \{a | a \in \sigma\}$. Functions can be composed by function composition. Let $f : A \rightarrow B$ and $g : B \rightarrow C$, then $g \circ f : A \rightarrow C$, such that $\forall x \in A : (g \circ f)(x) = g(f(x))$.

Current information systems log their activities (process steps) occurring in the context of the business processes they support. We assume that events in the log are uniquely associated with the activities in the process.

Definition 1 (Event). *Let \mathcal{E} be the universe of events, i.e. the set of all possible event identifiers. Events are executed in the context of an instance of a process. Let \mathcal{I} be the universe of process instance identifiers. We assume there is a function $pid : \mathcal{E} \rightarrow \mathcal{I}$ that maps each event to its process instance. Events can have additional parameters such as activity name, time stamp, executing resource and data attributes. We use \mathcal{V}_θ to denote the universe of values for a parameter θ . For each parameter θ we assume there exists a function π that maps an event to its parameter value, i.e. $\pi_\theta : \mathcal{E} \rightarrow \mathcal{V}_\theta$, e.g. $\pi_{cost} : \mathcal{E} \rightarrow \mathbb{Z}$.*

Events are linked (by pid) to a particular instance (or case) of a process. A log is basically a sequence over events from which event traces can be derived: A trace is an ordered sequence of events belonging to the same process instance where time is non-decreasing.

Definition 2 (Event trace). *An event trace is sequence of events $\sigma \in \mathcal{E}^*$, such that each event belongs to the same process instance, appears only once in the sequence and time is non-decreasing, i.e., σ is such that for $1 \leq i < j \leq |\sigma|$: $pid(\sigma(i)) = pid(\sigma(j))$, $\sigma(i) \neq \sigma(j)$ and $\pi_{time}(\sigma(i)) \leq \pi_{time}(\sigma(j))$. The universe of all event traces over \mathcal{E} is denoted as $\mathcal{T}^\mathcal{E}$.*

In absence of time stamps, we assume events are ordered by their occurrence in the log, i.e. $\pi_{time}(\sigma(i)) = i$.

Definition 3 (Event log). *An event log (in the remainder referred to as log) is a set over event traces, formally $L \subset \mathcal{T}^\mathcal{E}$, such that each event occurs in at most one trace. $\forall \sigma_1, \sigma_2 \in L : set(\sigma_1) \cap set(\sigma_2) = \emptyset$ or $\sigma_1 = \sigma_2$.*

3 Adaptive Simulation Model

In this section we will elaborate on the definition of an adaptive simulation model that supports the (re-)estimation (adaptation) of simulation parameters during execution.

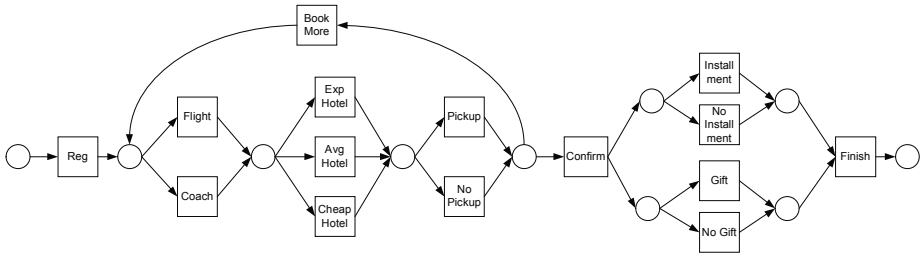


Fig. 1. The travel agency process

As running example, we consider a simple travel agency process where customers can compose a trip by booking a flight or coach transport, and a hotel (luxury, middle class or budget), for one or more days. The trip can be composed of multiple transport-hotel combinations. For each hotel stay customers can make use of a pickup service that transports them to the city centre or the airport. Clients may choose to pay their holidays by installment. In addition customers who spend much money are rewarded with a gift. A good estimation for the number of pickups is necessary for arranging a suitable contract with one of the local taxi companies. The agency would also like to estimate the number gifts to be purchased. Early market research indicates a trend towards a decreasing budget for clients. The agency has disposal of a log containing information about customers of the past years. How will the expected trend affect the number of pickups and gifts for the next year?

Figure 1 depicts the control flow for the travel agency. It contains all typical routing constructs such as sequentiality, parallelism, iteration, synchronisation, deterministic and non-deterministic choices of the environment. For meaningful analysis parameters (e.g. activity durations and probabilities) should be added to the control flow that reflect the real execution of the process. Moreover, for accurate simulation, we need to incorporate existing execution dependencies into the model. We use the log of the process that contains past executions to detect dependencies and to estimate the simulation parameters.

Static models cannot capture correlations between the parameters, such as a decreasing probability to choose going on with booking after every iteration, or a correlation between the choice of transportation and the hotel class. This results in inaccuracies of the analysis, e.g. the estimation of the number of gifts that should be ordered.

In adaptive simulation models, we incorporate dependencies into the model. This allows simulation parameters to be updated during the simulation execution by considering the predictors that influence the parameter value and some equation describing how the simulation parameter changes in terms of the values of the predictors, e.g. the probability for booking an expensive hotel for customers who booked a flight is 60%, and for those who chose a coach it is 10%. Historical execution data, captured in a log, is used to find an equation

that predicts the parameter value based on the value the predictor chosen for this parameter. During the simulation, predictor values are derived from the trace of the running simulation instance, e.g. flight is the transportation type booked for some instance. The equations to determine the values of simulation parameters and the equations for deriving predictor values from the (prefix) of the simulation trace are included in the model. Note that each parameter can depend on multiple predictors and different parameters can have different predictors.

Note that predictors are not simply case variables that are defined by the designer of the process, nor are correlations the decision rules for the process. For example, the duration of an activity is the time that is actually needed for its execution rather than a predefined value. Different resources might need different time periods to execute the activity, which is also not something typically predefined.

In the remainder of this section we represent steps of the business process as activities in the simulation model, e.g. a (coloured) Petri net or any other formalism with clear execution semantics that allow for simulation. Activities can be associated to parameters such as cost, duration and execution probability.

Definition 4 (Model Parameters). *Let \mathcal{A} be the universe of activities. The set of activities in model M is denoted as A_M , where $A_M \subseteq \mathcal{A}$. Activities can have additional parameters and we use Θ_M to denote the parameters of model M . The domain of parameter values of M is denoted as \mathcal{V}_Θ . The values for the parameters of the model are stochastic values that we estimate based on a log (L) of some process associated to M .*

One can annotate the model with *fixed* values for each simulation parameter, following e.g. the approach proposed in [17].

Definition 5 (Annotated Static Model). *Let M be a model describing the relation between the set of activities $A \subseteq \mathcal{A}$. The static annotated version of M is described by $\mathcal{M}_s = (M, val_s)$ where $val_s : \Theta_M \rightarrow \mathcal{V}_\Theta$ is a function that maps parameters to parameter values.*

Parameters are mapped to the average value of this parameter in the log. The values of the static model are fixed, regardless the current simulation instance.

In adaptive models we assume that the values of simulation parameters depend on the state of the instance and can change during the development of the instance. A regression equation describes the relation between a response variable (here simulation parameter) and explanatory (or predictor) variables in a data set.

Definition 6 (Regression Equation for Parameters). *The regression equation for a parameter θ is a function $f_\theta : \mathbf{X} \rightarrow \mathcal{V}_\theta$, describing the response of parameter θ to some experimental setting specified by a vector of predictor variables $\mathbf{x} \in \mathbf{X}$. We use \mathcal{R} to denote the universe of regression equations.*

The selection of f_θ is a choice that should be carefully matched with the data set and assumptions on the data. A general additive multiple regression model

which relates a dependent variable y to k predictor variables x_1, x_2, \dots, x_k , is given by the model equation $y = a + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_k x_k + e$, where a is the intercept, e is random deviation and each β_i is a population regression coefficient for predictor x_i [7]. In this model the right hand side of the equation is the population regression function (f_θ) that determines the outcome given a vector of predictor variables \mathbf{x} . Qualitative predictor variables (e.g. the name of an activity) can be encoded [7]. To predict the probability we use the generalized logit model for multinomial response [3]. Statistical packages such as R [13], a software environment for statistical computing and graphics, contain functionality for encoding data and fitting models on data.

Definition 7 (Predictor Value Function). *The predictor value function ϑ is a function that maps a trace to a vector of predictor values, $\vartheta : \mathcal{T}^\mathcal{E} \rightarrow \mathbf{X}$. We use \mathcal{V} to denote the universe of predictor value functions.*

The predictor value function extracts the value of the predictor from a partial trace. For example for pickup probability in Figure 1 the predictor value function could be defined as $\vartheta(\eta) = [\text{lastHotel}(\eta), \text{lastTravel}(\eta)]$, where η is the partial trace of the instance, lastHotel determines the type of the last hotel that was booked and lastTravel determines the type of the last travel that was taken.

Definition 8 (Annotated Adaptive Model). *Let M be a model describing the control flow based on the selection of activities $A \subseteq \mathcal{A}$. The adaptive annotated version of M is described by $\mathcal{M}_a = (M, \text{val}_a, \text{regr}, \text{pred})$ where $\text{val}_a : \Theta \times \mathcal{T}^\mathcal{E} \rightarrow \mathcal{V}_\Theta$ is a function that maps parameters to parameter values, depending on a (partial) simulation trace, $\text{regr} : \Theta \rightarrow \mathcal{R}$ is a function that maps parameters to corresponding regression functions and $\text{pred} : \Theta \rightarrow \mathcal{V}$ maps a parameter to a predictor value function. Assume $\text{regr}(\theta) = f_\theta$ and $\text{pred}(\theta) = \vartheta$. Then the parameter value for θ , given trace $\eta \in \mathcal{T}^\mathcal{E}$, is defined as $\text{val}_a(\theta, \eta) = f_\theta \circ \vartheta(\eta)$.*

Values for parameters of the adaptive model can be obtained by applying the associated regression equation on the current predictor values.

Example 1 (Adaptive Parameter Value). Consider again the travel agency process depicted in Figure 1. From the log we derive a regression equation for the pickup probability parameter. Suppose the regression equation to estimate the pickup probability is based on the type of the last booked hotel and the type of the last travel. The predictor value function extracts this property from the instance history. The parameter is estimated by applying the regression function on the current predictor values obtained from the current simulation trace, i.e. $\text{val}_a(\text{prob_Pickup}, \eta) = f_{\text{prob_Pickup}} \circ \vartheta(\eta)$, where the predictor value function is defined by $\vartheta(\eta) = [\text{lastHotel}(\eta), \text{lastTravel}(\eta)]$ and extracts the type of the last booked hotel and the type of the last travel from the trace. The regression function $f_{\text{prob_Pickup}}$ predicts the value of the pickup probability, given the last type of hotel and travel.

4 Mining Dependencies

For model M with parameters Θ_M and log L we mine an adaptive simulation model $\mathcal{M}_a = (M, val_a, regr, pred)$, where we set the adaptive parameter value for each parameter according to the current simulation trace η using the predictor value function and the regression function. Using regression analysis we can find dependencies between parameter and predictor values.

What can be suitable predictor candidates in terms of event traces of business processes? An obvious predictor candidate seems to be the partial trace of the instance. For real-life processes however, the log contains a wide variety of traces, but typically not many of them follow the same execution scheme and not all possible traces are contained in the log. As we showed in [18], trace abstractions can be applied to tackle this issue. Such abstractions consider some characteristics of the trace rather than the exact trace; the occurrence of a single activity, or a choice that was made at some point in the process are examples of abstractions that can be used as predictor. The goal is to find those trace characteristics (captured by a trace abstraction) that are good predictors for a simulation parameter. We do this by applying existing statistical methods where we define regression models for different predictor combination and determine which regression model fits the data set best. During simulation, parameters in the adaptive simulation model are determined by the associated regression model, based on the abstraction values for the current simulation trace.

Abstractions we consider are functions that map one representation of a partial trace to another, leaving out irrelevant details. Regression analysis is used to detect which abstractions are good predictors for a parameter. The input values for the regression equation are defined by the predictor value function ϑ (cf. Definition 7).

Definition 9 (Predictor values for Abstractions). *Let the predictors for a regression equation be given by a vector of k abstractions $[\alpha_1, \dots, \alpha_k]$. Then for all traces $\eta \in \mathcal{T}^{\mathcal{E}}$ the input for the regression equation is defined as $\vartheta(\eta) = [\alpha_1(\eta), \dots, \alpha_k(\eta)]$, where $\alpha_i(\eta)$ denotes the value of the i^{th} abstraction applied to η .*

4.1 Abstractions

This section presents an overview of elementary abstractions and compositions thereof.

Property projection ($property_p^\alpha : \mathcal{T}^{\mathcal{E}} \rightarrow \mathcal{V}^*$) converts a sequence of events to a sequence of their properties, e.g. a sequence of data attributes or time stamps, i.e. $property_p^\alpha(\langle a_1, \dots, a_n \rangle) = \langle \pi_p(a_1), \dots, \pi_p(a_n) \rangle$.

Event projection is an abstraction that can be used to extract specified elements from a trace. Event projection is a function ($event_A^\alpha : \mathcal{V}^* \rightarrow \mathcal{V}^*$) that retains elements of $\sigma \in \mathcal{V}^*$ that are in A , i.e. $event_A^\alpha(\sigma) = \sigma \uparrow A$.

Window abstraction defines the region of interest within a trace as sub-trace. Window abstraction takes some (or all) consecutive elements of the trace ($window^\alpha : \mathcal{V}^* \rightarrow \mathcal{V}^*$). The window can be specified by an interval between two points (denoted as $window_{P_1, P_2}^\alpha$), or by one point, a direction and a width (denoted as $window_{P, d, w}^\alpha$). A point P can be a concrete event (e.g. the last occurrence of an event with a certain event name) or it can be characterised by some condition on the event (e.g. the i^{th} event from the trace). The direction d of the window is specified prior to ($<$) or after ($>$) the point. The width w of the window is specified by a time interval or some condition.

Bag, set and cardinality abstractions Abstraction from the event ordering can be done by the bag abstraction. Set abstraction abstracts both from event ordering and their frequencies. Cardinality abstraction can be used to focus on the size of sequences, sets and bags.

Bag abstraction ($bag^\alpha : \mathcal{V}^* \rightarrow \mathbb{N}^{\mathcal{V}}$) is a function that transforms a sequence $\sigma \in \mathcal{V}^*$ into a bag, i.e. $bag^\alpha(\sigma) = par(\sigma)$.

Set abstraction ($set^\alpha : \mathcal{V}^* \rightarrow 2^{\mathcal{V}}$) is a function that transforms a sequence $\sigma \in \mathcal{V}^*$ into a set, i.e. $set^\alpha(\sigma) = set(\sigma)$.

Cardinality abstraction ($cardinality^\alpha : \mathcal{C} \rightarrow \mathbb{N}$) is a function that gives the size of a collection \mathcal{C} , where \mathcal{C} is \mathcal{V}^* , $2^{\mathcal{V}}$, or $\mathbb{N}^{\mathcal{V}}$, i.e. $\forall c \in \mathcal{C} : cardinality^\alpha(c) = |c|$.

Last occurrence abstraction considers the last occurring element from a specified set. This abstraction allows us to look, for example, at the most recent value of a data element associated with an activity that can re-occur in an iterative process, such as the last test outcome in Figure 2.

Last occurrence abstraction ($last_A^\alpha : \mathcal{V}^* \rightarrow A \cup \{\perp\}$) is a function that gives the last occurring element of set A in trace σ over E , if any, otherwise undefined \perp .

$$last_A^\alpha(\sigma) = \begin{cases} t, & \text{if } \exists t \in (A \cap E), \gamma \in E^*, \delta \in (E \setminus A)^*, \text{ such that } \sigma = (\gamma; t; \delta) \\ \perp, & \text{if } \sigma \in (E \setminus A)^*. \end{cases}$$

Existence abstraction detects the occurrence of a specified event. For example, for an insurance company the probability that the extensive procedure for handling claims will be chosen is higher if the person has already committed fraud. Existence abstraction is a function ($existence_e^\alpha : \mathcal{V}^* \rightarrow \{true, false\}$) that indicates whether element e is part of a sequence, i.e. $existence_e^\alpha(\sigma) = e \in \sigma$.

Duration abstraction can be used to obtain the duration between two events, e.g. the shorter the test procedure is, the more likely its result will be negative and another repair try will be needed. Duration abstraction ($duration_{e_1, e_2}^\alpha : \mathcal{V}^* \rightarrow \mathbb{N}$) is a function that indicates the time duration between two events e_1 and e_2 , i.e. $duration_{e_1, e_2}^\alpha(\sigma) = \pi_{time}(event_{e_2}^\alpha) - \pi_{time}(event_{e_1}^\alpha)$.

In practical applications combinations of abstractions, constructed as function compositions, are often used.

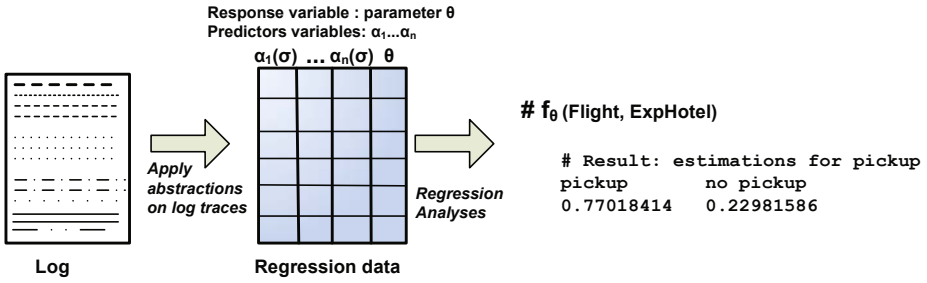


Fig. 2. Mining predictors (abstractions)

Example 2 (Combination of abstractions). The number of iterations already taken when executing a process from Figure 1 can be computed from a partial trace as the number of occurrences of the *BookMore* transition, i.e. $cardinality^\alpha \circ event_{\{BookMore\}}^\alpha(\sigma)$.

4.2 Mining Predictors

Section 4.1 identifies a collection of possible abstraction candidates that can be used to predict a parameter. For the selection of suitable predictors for a parameter we can define regression models with different combinations of predictors and find the model that best fits the data set that is derived from the execution log. In fact the execution log is converted into a list of parameter observations where each observation contains the value of the parameter and the values of all predictors under consideration. The predictor values for a parameter can be obtained by applying each predictor on the prefix of the parameter. The data set consists of the observations of all log traces for a parameter and a collection of abstractions.

Suppose, we want to observe the probability of selecting option c_1, c_2 or c_3 (in a free choice construct) where we consider the set abstraction set^α and cardinality abstraction $cardinality^\alpha$, which is used to count the number of times activity a occurs. Consider the observations for log trace $\sigma = \langle a, b, e, c_1, b, f, c_3, a, b, f, c_1 \rangle$. In this trace there are three observations, one for each occurrence of c_1, c_2 or c_3 . For the first occurrence of c_1 the observation is $[\{a, b, e\}, 1, c_1]$, determined by the set^α and $cardinality^\alpha$ on prefix $\langle a, b, e \rangle$, similarly we can observe $[\{a, b, e, f\}, 1, c_3]$ and $[\{a, b, e, f\}, 2, c_1]$.

The data set, which can be obtained by traversing the log once, is the input for regression analysis. There are different methods to systematically determine the model with the best combination of predictors for a data set. One can step-wise eliminate or add predictors to the model based on statistical relevance with respect to a reference model, or define all models, based on the power set of predictors, and find the best fitting model. In the adaptive simulation model the best fitting regression model is used for predicting the parameter value. Recall that the predictors of a regression model are in fact abstractions. Applying these abstractions on the simulation trace yields the predictor values for the regression model.

5 Experiments

In this section we illustrate and validate our approach by using adaptive simulation models to analyse the effect of trends on a business process. Since it is infeasible to expose a real-life process to trends for the purpose of validating our approach, we conduct our experiments on a reference model which is based on the control flow given in Figure 1. From the log of the reference model (M_r) we derive a static simulation model (M_s) and an adaptive simulation model (M_a). The three models are then exposed to the same trend and the results are compared.

5.1 Experiment Set-Up

For the experiments we define a reference model M_r that emulates a real process and produces logs. From these logs we build a static M_s and an adaptive simulation model M_a . To evaluate the suitability of the simulation models for evaluating business trends we expose the business process (the reference model) and the simulation models to some trends, i.e. a change in the environment of the processes such as a customer bias towards cheaper transportation. We compare the capabilities of the adaptive and static model by comparing their performance with respect to the performance of the reference model. The performance is measured in occurrence ratio of *Pickup* and *Gift* transitions.

Reference Model. The role of the reference model is to produce logs by emulating a real life process and its environment. A complex stochastic scheme has been added to the control flow depicted in Figure 1 in order to equip the process with non-trivial dependencies, so that it became impossible to analytically compute dependencies between different choices and parameter values. In real life these dependencies are not known; in the reference model we in fact use them to emulate human decisions. Furthermore the reference model has been equipped with logging functions that log every activity that is executed during simulation.

We run 30 replications of 10.000 instances on M_r , each replication creating a log. We randomly select one of these logs to detect dependencies and to create an adaptive model M_a and a static model M_s . To create these models one can annotate the *control flow* of the travel agency process based on the log of M_r . (Note that the control flow can also be mined from this log by standard process mining techniques.) It is important to note that these models are created based on the log without using any knowledge of the stochastic scheme of M_r .

Static Model. We mine the log of M_r to find the values for the parameters of the static model M_s , using techniques from [17]. From the log of M_r we can determine such parameters as the probability to book an expensive hotel as the percentage of cases in which expensive hotels were booked and annotate the control flow to obtain the static model, cf. Def. 5. Recall that in the static model the values are fixed and that the partial simulation trace (or the history) is not used to estimate their values.

Adaptive Model. For the adaptive model we use the log of M_r to determine the regression model that best fits the log, cf. Section 4. We annotate the control flow model with the obtained model, and the predictors (abstractions), cf. Def. 8, so that the parameter values can now be determined during the simulation, depending on the partial simulation trace.

Mining Simulation Model Parameters. The probabilities for the static model for firing the *Pickup* and the *Gift* transitions can be mined using the **Performance Analysis with Petri net** plugin of ProM [1]. For the adaptive model parameters we determine suitable predictors for those transitions with regression analysis using the `multinom` function from the `nnet` library in R [13]. The R data is created using our ProM R Data plugin [1]. This function fits multinomial logit models with nominal response categories.

Injecting the Trend. To evaluate the suitability of the simulation models for evaluating business trends we expose the reference model to a lower customer budget, affecting the way of travelling, hotel type and number of composed travel-hotel combinations. On the reference model the trend results in decreasing the share of flights from 50% to 10%. The shares of expensive, medium and cheap hotels change from 33% to 10%, 25% and 65%, respectively. Finally, the number of booked combinations drops. The trend of booking less flights is injected in the simulation models in a consistent manner; the bounds for the guards that control the transportation choice are set to the mentioned probabilities. Note that, except for the injected trend, the simulation models do not change.

Running the simulations. All models have been implemented as coloured Petri net in CPN tools [11], which is a well established tool for modelling and analysis of Coloured Petri Nets. We refer the interested reader for implementation details of the models to [12]. For each model we run 30 replications of 10.000 instances. For each replication we determine the occurrence ratio of *Pickup* and *Gift* transitions. The result of all the replications are depicted as confidence intervals.

5.2 Results

The procedure to mine correlations starts with the definition of the response variable and the predictors for the estimation for the response variable.

Estimating the Pickup Probability. To convert the log into regression data, the response variable and the predictor variables have to be chosen. For the probability of *Pickup* we consider the following abstractions: (1) which hotel was the last one that was booked, (2) last type of travel and (3) the number of iterations. We convert the log for these abstractions to a data format that can be used for regression analysis. The data is imported into R where we fit the data using `multinom` function from the `nnet` library. For each combination of abstractions we defined a logit model. From the models with a single predictor, the model

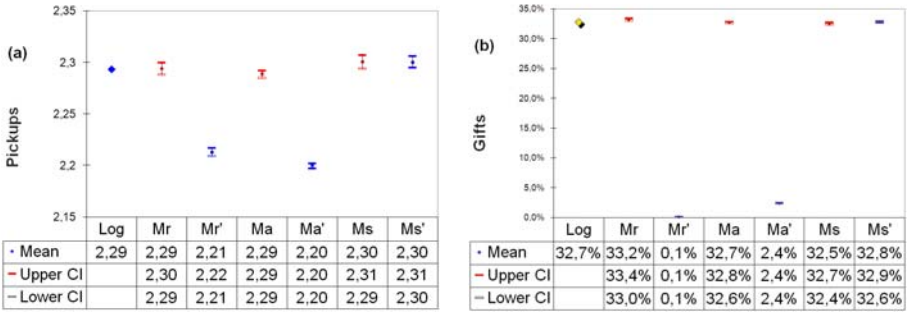


Fig. 3. Simulating the effect of clients becoming more poor. M_r is the reference model, M_a is the history-dependent model and M_s a history-independent model. After inserting the new trend referred to as M'_r , M'_a and M'_s . (a) Depicts the effect on the number of *Pickup* using the last occurrence of the hotel type as predictor and (b) depicts the effect on percentage clients that get a *Gift*, using the number of iterations as predictor.

that considered the last hotel fitted the data best. Moreover, adding more predictors did not significantly improve the results. Therefore, only the last hotel was used for estimating the response probabilities for all predictor combinations, e.g. the probability for *Pickup* given that the last hotel was an expensive hotel is 53%, for a budget hotel this probability is 44%.

The conversion of the log took 1 minute and 24 seconds and the execution of all R commands (including the fitting and testing of other models) took 39 seconds.

Estimating the Gift Probability. Similarly we determine the predictors for estimating the probability for gifts. Different abstractions and combinations thereof have been considered, including (1) the set abstraction on the hotels, (2) the sequence of the last two hotels, (3) the set abstraction on the travel types and (4) the sequence of the last two travel types and (5) the number of booked combinations. The model using the last abstraction as predictor was the one with the best fit.

Simulated Process Performance. Figure 3 depicts the results of simulation of the different models before and after the injected trend of customers with a lower budget. The results are depicted as 95% confidence intervals, that depict the occurrence ratios of the *Pickup* and the *Gift* transitions. On the left side of the figures we depict the value for the log produced by M_r that was (randomly) selected to derive the adaptive M_a and static model M_s . The confidence intervals shown for M_r , M_a and M_s are based on simulation without introducing the trend; here both M_a and M_s approximate the behaviour of M_r well. After the trend is introduced, only M_a is able to follow the direction of the trend whereas M_s is unable to do so, because essential correlations are not taken into account as all choices are considered mutually independent.

Notice that M_a slightly underestimates the number of pickups. This is caused by the fact that the log randomly selected from 30 replications contains slightly less pickups than obtained on all the 30 logs on average. Also note that M_a overestimates the number of gifts for the new situation due to the fact that we try to capture a complex data dependency by a very simple abstraction, which does not exactly captures the dependency but approximate it.

It is clear the the adaptive model gives a much better approximation than the static model. Our experiments show that the history-dependent mechanism adapting simulation parameters according to the developments in the running instance is able to propagate environmental changes in the simulation. For logs containing data, abstractions on data can be used to obtain even more precise results.

6 Related Work

Business Process Simulation (BPS) has been indicated by [10] as an essential technique for Business Process Re-engineering (BPR) where it is not only important to understand the static behaviour of the process, but also to accurately predict the outcome of proposed and/or expected changes for the process to judge the effect on the organisation performance. This does not only apply to the area BPR, which traditionally focuses on complete redesign of existing processes, but it is also interesting in a less radical setting: *“How will a trend affect the performance of my existing business process?”*. Simulation offers support for randomness, uncertainty and interdependencies, making it a valuable technique for business process management.

The biggest challenge in the development of a simulation model is obtaining an accurate model that is close to reality. To tackle the problem of creating realistic simulation models, [17] present a method to generate simulation models based on actual information from logs. The authors create simulation models in CPN tools [11] capturing the control flow perspective, the resource perspective and the data perspective and the current state. In their approach they assume all variables to be independent. This assumption is, however, unrealistic for real-life business processes, as [5] explains, dependencies and correlations present in business processes.

History-dependent Petri nets [18] (HDSPNs) are an extension of classical Petri nets [6,16] that use a history-dependent mechanism to model history-dependent choices. This approach can easily be extended to model cost and dependencies for activities.

Detection of correlations from process logs and using them for business process simulation has not been studied extensively yet. Usually assumptions are made about the dependency and/or distributions of the variables [4,17,15]. Correlations between quantitative variables in business processes can be used to obtain more accurate settings for cost and durations of variables. These correlations can be derived from data with simple statistical techniques. When qualitative variables (e.g. resource and activity names) are involved, more advanced techniques such as regression analysis are required. Closest related work is the one

on predictions in business processes [8]. There non-parametric regression analysis is used to predict the finishing time of an instance. Intermediate process dependencies are not considered. The complexity of constructing the regression model is a serious limitation of that approach.

7 Conclusion

In this paper we focused on analysing the effects of trends on existing business processes. The analysis is performed on simulation models based on information obtained by actual executions (a log) of the process. The main idea is that dependencies need to be included into the simulation model to accurately predict the global effect of new trends. For this purpose we introduce adaptive simulation models that have simulation parameters (re-)estimated during execution. We discussed how dependencies can be derived from a log of a business process using regression analysis. We use abstractions on traces to balance between the amount of data available in the log and the amount of information necessary to make good predictions. The conversion of log data to R data, for selected abstractions candidates is implemented as the R `Data` plugin in the ProM framework [1]. This data can be used directly in R to determine the best fitting regression model.

The obtained dependencies are included into the simulation model by means of a history-dependent mechanism that uses the partial simulation trace to determine further simulation parameters. We have demonstrated the application of dependencies in adaptive simulation models on a reference model from whose log we created an adaptive and a static simulation model. Only the adaptive simulation model was able to propagate the trend into the correction direction.

An important direction for future work goes into the direction of the generation of R data given some abstractions. Currently abstractions can have many values (levels) and the generated data can be sparse, making it unsuitable for regression. As future work we plan to look into data mining techniques to cluster abstraction levels. Furthermore, we will focus on doing experiments that consider the probability, cost and duration of parameters and predictors and where more complex abstraction compositions are considered.

References

1. ProM Nightly Builds (2006), <http://prom.win.tue.nl/tools/prom/nightly/>
2. 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 and Knowledge Engineering* 47(2), 237–267 (2003)
3. Agresti, A.: *Categorical Data Analysis*, 2nd edn. Wiley Series in Probability and Statistics. Wiley-Interscience, Hoboken (2002)
4. Baccelli, F., Konstantopoulos, P.: *Estimates of Cycle Times in Stochastic Petri Nets*. Rapport de recherche 1572, INRIA, Rocquencourt (1992)
5. Barros, A.P., Decker, G., Grosskopf, A.: Complex Events in Business Processes. In: Abramowicz, W. (ed.) *BIS 2007*. LNCS, vol. 4439, pp. 29–40. Springer, Heidelberg (2007)

6. Desel, J., Esparza, J.: Free Choice Petri Nets. Cambridge Tracts in Theoretical Computer Science, vol. 40. Cambridge University Press, Cambridge (1995)
7. Devore, J., Farnum, N.: Applied Statistics for Engineers and Scientists, 1st edn. Duxbury, Boston (1999)
8. van Dongen, B.F., Crooy, R.A., van der Aalst, W.M.P.: Cycle Time Prediction: When Will This Case Finally Be Finished? In: Meersman, R., Tari, Z. (eds.) OTM 2008, Part I. LNCS, vol. 5331, pp. 319–336. Springer, Heidelberg (2008)
9. Ferson, S., Burgman, M.A.: Correlations, Dependency Bounds and Extinction Risks. *Biological Conservation* 73(2), 101–105 (1995); Applications of Population Viability Analysis to Biodiversity
10. Gladwin, B., Tumay, K.: Modeling Business Processes with Simulation Tools. In: WSC 1994: Proceedings of the 26th conference on Winter simulation, San Diego, CA, USA, pp. 114–121. Society for Computer Simulation International (1994)
11. Jensen, K., Kristensen, L.M., Wells, L.: Coloured Petri Nets and CPN Tools for Modelling and Validation of Concurrent Systems. *International Journal on Software Tools for Technology Transfer* 9(3-4), 213–254 (2007)
12. Jian, J.: Mining Simulation Models with Correlations. Master's thesis, Eindhoven University of Technology, Eindhoven, The Netherlands (2009)
13. R Development Core Team. R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria (2006), ISBN 3-900051-07-0
14. Regev, G., Wegmann, A.: Why Do We Need Business Process Support? Balancing Specialization and Generalization with BPS Systems (Introductory note). In: CAiSE Workshops (2003)
15. Reijers, H.A.: Case Prediction in BPM Systems: A Research Challenge. *Journal of the Korean Institute of Industrial Engineers* 33, 1–10 (2006)
16. Reisig, W., Rozenberg, G. (eds.): APN 1998. LNCS, vol. 1491. Springer, Heidelberg (1998)
17. Rozinat, A., Mans, R.S., Song, M., van der Aalst, W.M.P.: Discovering Simulation Models. *Inf. Syst.* 34(3), 305–327 (2009)
18. Schonenberg, M.H., Sidorova, N., van der Aalst, W.M.P., van Hee, K.M.: History-Dependent Stochastic Petri Nets. In: Voronkov, A. (ed.) PSI 2009. LNCS, vol. 5947, pp. 366–379. Springer, Heidelberg (2010)

Workflow Soundness Revisited: Checking Correctness in the Presence of Data While Staying Conceptual

Natalia Sidorova, Christian Stahl, and Nikola Trčka

Department of Mathematics and Computer Science
Eindhoven University of Technology
P.O. Box 513, 5600 MB Eindhoven, The Netherlands
{N.Sidorova,C.Stahl,N.Trcka}@tue.nl

Abstract. A conceptual workflow model specifies the control flow of a workflow together with abstract data information. This model is later on refined to be executed on an information system. It is desirable that correctness properties of the conceptual workflow would be transferrable to its refinements. In this paper, we present *classical workflow nets extended with data operations* as a conceptual workflow model. For these nets we develop a novel technique to verify *soundness*. This technique allows us to conclude whether at least one or any refinement of a conceptual workflow model is sound.

1 Introduction

Information systems are a key technology in today's organizations. Prominent examples of information systems are Enterprise Resource Planning Systems and Workflow Management Systems. Processes are the core of most information systems [9]. They orchestrate people, information, and technology to deliver products. In this paper, we focus on *workflows*—processes that are executed by an IT infrastructure.

A workflow is usually iteratively designed in a bottom-up manner. First the control flow of the workflow is modeled. The control flow consists of a set of coordinated tasks describing the behavior of the workflow. Later the control flow is extended with some data information. The resulting model is an abstract or *conceptual workflow model*, which is typically constructed by a business analyst. This conceptual model can be used for purposes of documentation, communication, and analysis. It may abstract from concrete data values, such as the condition of an if-then-else construct, and it does usually not specify how concrete data values are stored.

To actually execute this workflow on a Workflow Management System, the conceptual workflow model is instantiated with full details, a task typically done by business programmers (who often have insufficient background knowledge of the process) and not by the business analysts themselves. For instance, the business programmer specifies concrete data values and how they are stored.

Modeling both abstract and executable workflows is supported by industrial workflow modeling languages available on the market.

Designing a workflow model is a difficult and error-prone task even for experienced modelers. For a fast and thus cost-efficient design process it is extremely important that errors in the model are detected during the design phase rather than at runtime. Hence, verification needs to be applied at an early stage—that is, already on the level of the conceptual model.

Formal verification of a conceptual workflow model imposes the following two challenges. First, it requires an adequate formal model. With adequate we mean that the model captures the appropriate level of abstraction and enables efficient analysis. So the question is, *how can we formalize commonly used conceptual workflow models?* Second, verification must not be restricted to the control flow, but should also incorporate available data information. Thereby the conceptual workflow model may specify abstract data values that will be *refined* later on. Here the question is, *can we verify conceptual workflows in such a way that the results hold in any possible data refinement* (i.e., in all possible executable versions)?

In this paper, we investigate these two questions. We focus on one of the most important requirements for workflow correctness, namely the *soundness property* [1]. Soundness guarantees that every task of the workflow can be potentially executed (i.e., it is not dead), and that the workflow can always terminate (i.e., it is free of deadlocks and livelocks). However, current techniques for verifying soundness are restricted to control flow only.

To answer the first question, we propose *workflow nets with data* (WFD-nets) as an adequate formalism for modeling conceptual workflows. A WFD-net is basically a workflow net (i.e., a Petri net tailored towards the modeling of the control flow of workflows) extended with conceptual read/write/delete data operations. WFD-nets generalize our previous model in [18,19] by means of supporting arbitrary guards (previously only predicate-negation and conjunctions were allowed). As a second contribution, we develop a novel technique for *analyzing soundness of WFD-nets*. Unlike existing approaches which could give false positives (i.e., the analysis gives sound, but the workflow is actually unsound when the data information is refined) or false negatives (i.e., the analysis gives unsound, but the workflow with refined data is sound), our proposed technique gives neither false positives nor false negatives. It is based on may/must semantics [14] that guarantees the results to be valid in *any* concrete data refinement. If a WFD-net is proven must-sound, then it is sound in any possible data refinement; if it is not may-sound, then no data refinement can make it sound. In case where a WFD-net is may-sound but not must-sound, our approach gives an honest answer “I do not know; it is sound in some data refinement and unsound in some other”. We interpret WFD-net as hyper transition systems, not as standard may/must transition systems. Doing this we achieve much better precision (i.e., less “I do not know” answers).

The paper is structured as follows. In Sect. 2, we show the potential problems with soundness verification for conceptual workflow models by means of two

examples. Afterwards, we introduce the conceptual workflow model in Sect. 3 and its semantics in Sect. 4. In Sect. 5 we define soundness for workflow nets with data in the may/must setting. Finally, Sect. 6 draws the conclusion and discusses future work.

2 Motivating Examples

We illustrate the idea of modeling conceptual workflows with WFD-nets on the WFD-net in Figure 1 modeling a shipping company. Ignoring the transition guards (shown within squared brackets above transitions), and the read and write operations (denoted by `rd` and `wt` inside the transitions), Figure 1 depicts an ordinary workflow net that consists of 10 places (depicted as circles) and 9 transitions (depicted as squares). There are two distinguished places `start` and `end` modeling the initial and the final state, respectively.

Initially there is a token on `start`. The shipper receives goods from a client (`receive good`) to be shipped. In the model, transition `receive good` writes the data of the client (`cl`), the goods (`gds`), and the destination of the goods (`ads`). Then the shipper calculates the price (`calculate price`). Afterwards the shipment department and the customer adviser execute their tasks concurrently. If the price of the goods is high (i.e., `isHigh(price)` evaluates to true; the exact bound is left unspecified), express shipment is used (`ship express`). Otherwise, the cheaper standard shipment is used (`ship normal`). Based on the same condition, the customer advisor either calculates a bonus (`calculate bonus`) for the client and registers this bonus (`register bonus`) or no bonus is calculated (`no bonus`). In addition, clients sending goods of a high price are notified about their bonus and the shipment (`inform by call`), other clients receive only a notification of the shipment (`inform by mail`).

Clearly, this WFD-net is sound, i.e., starting with a (colored) token on `initial` it is always possible to reach a marking where only one token is on place `end`. In contrast, if we abstract from data and consider only the underlying workflow net, the net may deadlock. For example, without data the shipment department may decide to use express shipment, but the customer advisor does not calculate any bonus. This yields a token in `p5` and in `p8`, and the net gets stuck. This shows that ignoring data information in verification can lead to obtaining false negatives.

Suppose now that instead of the same predicate `isHigh(price)`, two possibly different predicates (say `isHighLeft(price)` and `isHighRight(price)`) are used in the left and the right part of the workflow (this is realistic, because these parts correspond to two different departments of the shipper). As we did not change the control flow, the classical WF-net method would still say that the workflow is not sound. Our previous work [19] on soundness of (simplified) WFD-nets would give the same verdict: the workflow is not sound due to the possibility that the predicates can have different truth values. With the methods we will introduce in this paper we will be able to give the correct answer: “I do not know—the workflow is sometimes sound (when `isHighLeft(price)`

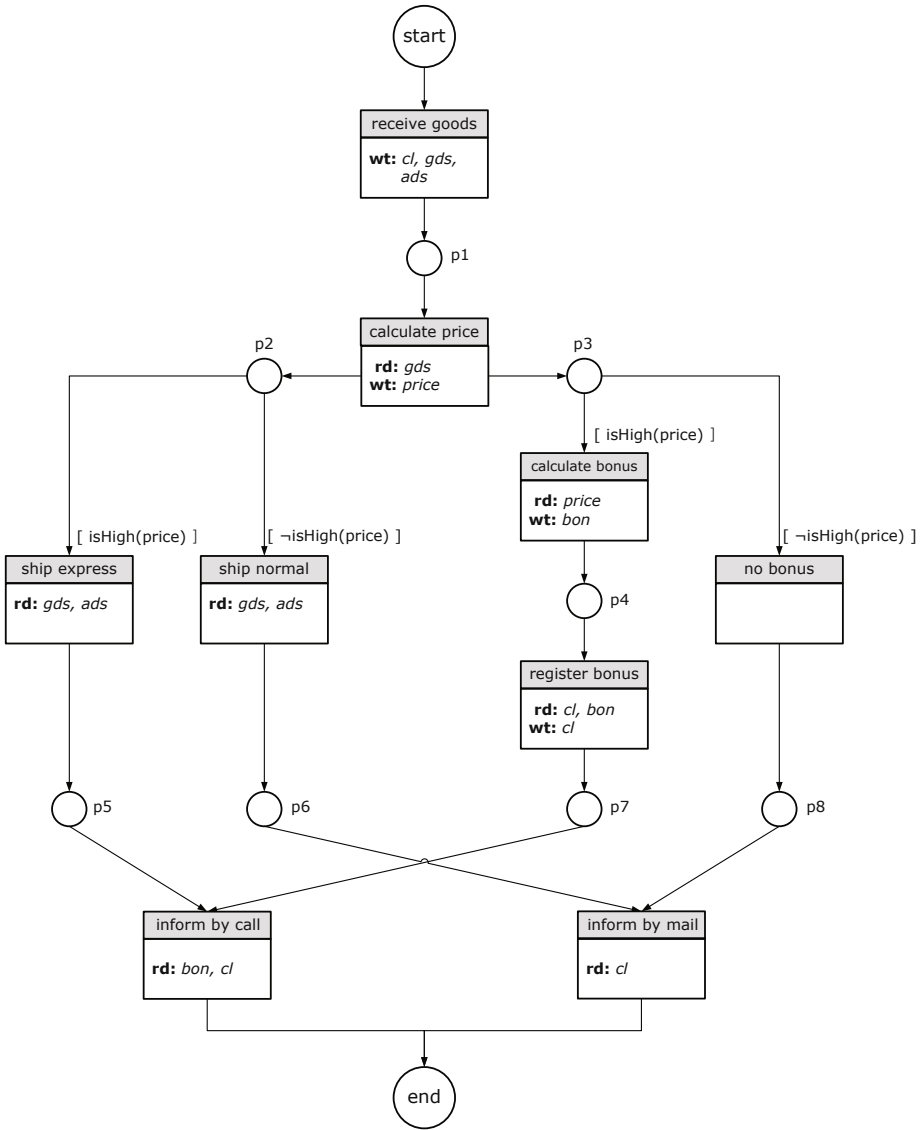


Fig. 1. WFD-net modeling a shipper – The workflow is sound, but verification ignoring data leads to the verdict “unsound”

and $\text{isHighRight}(\text{price})$ always evaluates to the same values), and sometimes it is not (when $\text{isHighLeft}(\text{price})$ and $\text{isHighRight}(\text{price})$ valuations can differ)”.

It is also possible to construct examples where the verification of classical WF-nets (without data information) would give a false positive, and our previous work on WFD [19] would give a false negative. For instance, consider the loop

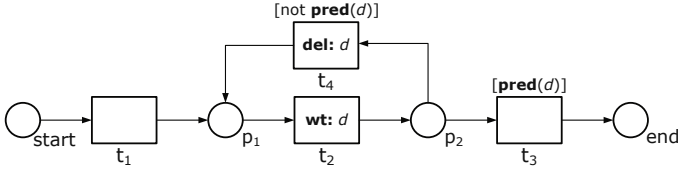


Fig. 2. WFD-net that is (non-)sound in some, but not in all, data refinements

in Figure 2 that is exited when some predicate depending on a data element d evaluates to true. The data element d is initialized inside this loop (transition t_2) and it is deleted (denoted by **del** inside transition t_4) and written in every loop cycle. In this case, proper termination is only possible if d eventually sets the exiting predicate to true. Therefore, the correct answer is again, “I do not know”, as the soundness property depends on the concrete data refinement. However, when using classical WF-nets, the data information is ignored and the possibility to exit the loop is considered to be available, which results in the verdict that the WF-net is sound. Similarly, the previous WFD-net method would see that there is a possible bad execution sequence (namely the infinite looping) and would report non-soundness.

Remark 1. It is important to note that the problems addressed in this paper are independent of the actual version of workflow soundness definition used. Although we restrict ourselves to classical soundness, ignoring data when considering other notions of soundness can result in obtaining false positive or false negative answers. If we, for example, consider relaxed soundness (requiring that for every task there is a completing path executing this task) (see 8), the workflow in Figure 1 would be correctly reported relaxed sound. However, if we used different predicates in the left and in the right side, the workflow was not relaxed sound. The verification result on the workflow without data is then obviously incorrect. Moreover, by swapping the guards on the right side of the net in Figure 1 we construct a workflow that has no completing execution at all. Note that also the unfolding of a workflow to have the data information incorporated into the control-flow (by the means of 18,19) does not help in this situation either: Consider again the example in Figure 2. Unfolding the workflow, we obtain a relaxed sound net. It is, however, obvious that there are data refinements that certainly prevent t_4 from being ever executed. Similar problems arise with all other soundness notions. These problems can also be addressed by using the may/must approach we are going to present in this paper.

3 Workflow Nets with Data

Given the motivation for incorporating data into workflow nets, this section formally defines workflow nets with data (WFD-nets). WFD-nets are based on Petri nets and workflow nets, so we define these two models first.

Definition 1 (Petri net). A Petri net $N = \langle P, T, F \rangle$ consists of two disjoint non-empty, finite sets P of places and T of transitions and of a flow relation $F \subseteq (P \times T) \cup (T \times P)$.

For a transition $t \in T$, we define the *pre-set of t* as $\bullet t = \{p \mid (p, t) \in F\}$, and the *post-set of t* as $t \bullet = \{p \mid (t, p) \in F\}$. Analogously we define the pre-set $\bullet p$ and the post-set $p \bullet$ for a place $p \in P$. A place p is called a *source* place if $\bullet p = \emptyset$, and a *sink* place if $p \bullet = \emptyset$.

At any time a place contains zero or more *tokens*, depicted as black dots. A state of a Petri net, called a *marking*, is a distribution of tokens over its places. Formally, a marking is defined as a mapping $m : P \rightarrow \mathbb{N}$, i.e., as a multiset over P . We use standard notation for multisets and write, e.g., $m = [2p + q]$ for a marking m with $m(p) = 2$, $m(q) = 1$, and $m(x) = 0$ for $x \in P \setminus \{p, q\}$. We define $+$ and $-$ for the sum and the difference of two markings and $=, <, >, \leq, \geq$ for comparison of markings in the standard way. For the marking m above we have, e.g., $m \leq [3p + 2q + r]$ and $m + [q + 3r] = [2p + 2q + 3r]$. A pair (N, m) , where N is a Petri net and m is a marking, is a *marked* Petri net.

A transition $t \in T$ is *enabled* at a marking m , denoted by $m \xrightarrow{t}$, if $m \geq \bullet t$. An enabled transition t may *fire*, which results in a new marking m' defined by $m' = m - \bullet t + t \bullet$. This firing is denoted as $m \xrightarrow{t} m'$.

Workflow nets [1] impose syntactic restrictions on Petri nets to comply to the workflow concept. The notion was triggered by the assumption that a typical workflow has a well-defined starting point and a well-defined ending point.

Definition 2 (WF-net). A Petri net $N = \langle P, T, F \rangle$ is a Workflow net (WF-net) if it has a single source place *start* and a single sink place *end*, and if every place and every transition is on a path from *start* to *end* (i.e., if $(\text{start}, n) \in F^*$ and $(n, \text{end}) \in F^*$, for all $n \in P \cup T$, where F^* is the reflexive-transitive closure of F).

Transitions in a WF-net are called *tasks*. A *case* is a workflow instance, i.e., a marked WF-net in which the place *start* is marked with one token and all other places are empty. Executing a workflow means to create a running instance of this workflow. Such an instance is called a *case*. Several cases of a workflow may coexist. Cases are assumed to be completely independent from each other (they only possibly share resources). Hence, each case is modeled as a copy of the corresponding workflow net N . We refer to the properties of $(N, [\text{start}])$ as the properties of N .

Example 1. Ignoring the transition guards and the read and write operations, Figure 1 depicts a WF-net. Places *start* and *end* are the source place and the sink place, respectively. Clearly, every place and transition is on a path from *start* to *end*. Each transition, such as *Inform by call*, models a task. The pre-set of *Inform by call* is the set $\{p5, p7\}$, and the post-set is the set $\{\text{end}\}$.

Adding data information. A workflow net with data elements is a workflow net in which tasks can read from, write to, or delete data elements. A task can

also have a (data dependent) guard that blocks its execution when it is evaluated to false.

We assume a finite set $\mathcal{D} = \{d_1, \dots, d_m\}$ of *data elements*, and we fix a set of predicates $\Pi = \{\pi_1, \dots, \pi_n\}$. We also assume a function $\ell : \Pi \rightarrow 2^{\mathcal{D}}$, called the *predicate labeling function*, that assigns to every predicate the set of data elements it depends on. When $\ell(\pi) = \{d_1, \dots, d_n\}$ for some predicate $\pi \in \Pi$, we sometimes write $\pi(d_1, \dots, d_n)$ to emphasize this fact. A *guard* is constructed from predicates by means of the standard Boolean operations; the set of all guards (over Π) is denoted by \mathcal{G}_{Π} . The function ℓ naturally extends to guards.

We now define a workflow net with data as a WF-net where every transition t is annotated with at most four sets: a set of data elements being read when firing t , a set of data elements being written when firing t , a set of data elements being deleted when firing t , and a transition guard. Note that we do not explicitly consider the update of data elements, because this is simply the combination of read and write at the same transition.

Definition 3 (WFD-net). A workflow net with data (*WFD-net*) $N = \langle P, T, F, \text{rd}, \text{wt}, \text{del}, \text{grd} \rangle$ consists of a WF-net $\langle P, T, F \rangle$, a reading data labeling function $\text{rd} : T \rightarrow 2^{\mathcal{D}}$, a writing data labeling function $\text{wt} : T \rightarrow 2^{\mathcal{D}}$, a deleting data labeling function $\text{del} : T \rightarrow 2^{\mathcal{D}}$, and a guard function $\text{grd} : T \rightarrow \mathcal{G}_{\Pi}$, assigning guards to transitions.

Example 2. An example of a WFD-net is the workflow of a shipper in Figure 1. Its data elements are $\mathcal{D} = \{\text{cl}, \text{gds}, \text{ads}, \text{price}, \text{bon}\}$. Consider transition `calculate bonus`, for instance. The labeling functions are $\text{rd}(\text{calculate bonus}) = \{\text{price}\}$, $\text{wt}(\text{calculate bonus}) = \{\text{bon}\}$, $\text{del}(\text{calculate bonus}) = \emptyset$, and $\text{grd}(\text{Ship normal}) = \neg \text{isHigh}(\text{price})$.

The next section assigns formal semantics to WFD-nets.

4 Semantics of WFD-nets

The model of WFD-nets is a conceptual model, a schema for characterizing several executable workflows. In this section we introduce a special semantics for WFD-nets that is based on hyper transition systems [14, 17] and allows us to capture all possible refinements of a WFD-net in one graph.

4.1 Behavior of WFD-nets

In a WFD-net data values are not specified, but we can distinguish non-created data values from created ones. In our semantics we choose to keep the exact value for the predicates in a state. Predicates and guards can be evaluated to true, false, or undefined (if some data element assigned to them does not have a value). This is formalized by the three abstraction functions assigning abstract values to the data elements, the predicates, and the guards, respectively. Function $\sigma_{\mathcal{D}} : \mathcal{D} \rightarrow \{\top, \perp\}$ assigns to each data element $d \in \mathcal{D}$ either \top (i.e., defined value) or

\perp (i.e., undefined value); Function $\sigma_{\Pi} : \Pi \rightarrow \{\mathsf{T}, \mathsf{F}, \perp\}$ assigns to each predicate one of the values true, false, and undefined. A consistency requirement that $\sigma(\pi) = \perp$ whenever $\sigma(d) = \perp$ and $d \in \ell(\pi)$ is imposed. A pair $\sigma = (\sigma_{\mathcal{D}}, \sigma_{\Pi})$ is called a *state*, and the set of all states is denoted by Σ . We use the following simplified notation: $\sigma(d) = \sigma_{\mathcal{D}}(d)$ for $d \in \mathcal{D}$, and $\sigma(\pi) = \sigma_{\Pi}(\pi)$ for $\pi \in \Pi$. As a guard $g \in \mathcal{G}_{\Pi}$ is built from Boolean operations, $\sigma(\text{grd}) \in \{\mathsf{T}, \mathsf{F}, \perp\}$ can be evaluated with the help of functions $\sigma_{\mathcal{D}}$ and σ_{Π} .

The next definition lifts the definition of a state of a WF-net to a WFD-net. We refer to a state of a WFD-net as a configuration¹. A configuration consists of a marking m of a WFD-net and a state.

Definition 4 (Configuration). *Let $N = \langle P, T, F, \text{rd}, \text{wt}, \text{del}, \text{grd} \rangle$ be a WFD-net. Let m be a marking of N , and let σ be as defined above. Then, $c = \langle m, \sigma \rangle$ is a configuration of N . The start configuration of N is defined by $\langle [\text{start}], (\{d_1 \mapsto \perp, \dots, d_n \mapsto \perp\}, \{\pi_1 \mapsto \perp, \dots, \pi_n \mapsto \perp\}) \rangle$. With Ξ we denote the set of all configurations, and $C_f = \{ \langle [\text{end}], \sigma \rangle \mid \sigma \in \Sigma \}$ defines the set of final configurations.*

In the initial configuration, only place `start` is marked, all data elements are undefined, and all predicates are evaluated to undefined. A configuration is a final configuration if it contains the final marking `[end]`.

Example 3. The initial configuration of the shipper in Figure 1 is defined to be $\langle [\text{start}], \sigma \rangle$, where σ assigns \perp to all data elements `cl`, `gds`, `ads`, `price`, `bon`. In addition, predicate `isHigh(price)` is \perp . Note that in Figure 1 no data element is deleted. However, we assume that upon reaching a final configuration (i.e., the case is completely executed), all data elements are deleted.

As Definition 4 lifts the notion of a state of a WF-net to a configuration of a WFD-net, we have to define the behavior of a WFD-net. For this purpose, we define when a transition t of a WFD-net N is enabled at a configuration $c = \langle m, \sigma \rangle$ of N .

The enabling of a transition t requires two conditions to be fulfilled. The first condition takes the control flow into account and requires that transition t must be enabled at marking m . The second condition considers the data values in configuration c . Any data element that is read by t or that is assigned to a predicate of t must be defined. In addition, the guard of t must evaluate to true.

An enabled transition t may fire. Firing of t changes the marking as well as the values of the data elements that have been written or deleted. As we do not know the concrete operations nor the values of the predicates, we have to consider any evaluation of the predicates. Hence, the firing of t yields a set of successor configurations $\langle m', \sigma' \rangle$. Each of these successor configurations has a marking m' , where firing t at marking m yields marking m' .

On the data level, we assign undefined (i.e., \perp) to each data element d that has been deleted when firing t , and we assign undefined to each predicate that

¹ The meaning of the term *configuration* here is “a state that includes data information”, and not the one related to configuring processes.

contains a data element that has been deleted. The reason is that reading always precedes writing, and writing always precedes deleting. Thus, no matter whether this data element has been also written, it is undefined after the firing of t . In addition, we assign defined (i.e., \top) to each data element d that has been written and not deleted when firing t , and evaluate each predicate that contains at least one data element that has been written and no data elements that have been deleted. The different evaluations of the predicates actually result in a set of successor configurations. This is formalized in the following definition.

Definition 5 (Firing rules for WFD-nets). *Let $N = \langle P, T, F, \text{rd}, \text{wt}, \text{del}, \text{grd} \rangle$ be a WFD-net. A transition $t \in T$ of N is enabled at a configuration $c = \langle m, \sigma \rangle$ of N if $m \xrightarrow{t}$, all data elements $d \in \text{rd}(t)$ are defined, all data elements assigned to any predicate occurring in the transition guard $\text{grd}(t)$ of t are defined, and $\sigma(\text{grd}(t)) = \top$. Firing t yields a set $C \subseteq \Sigma$ of configurations with $C = \{ \langle m', \sigma' \rangle \mid m \xrightarrow{t} m' \wedge (\forall d \in \text{del}(t) : \sigma'(d) = \perp \wedge \forall \pi \in \Pi : d \in \ell(\pi) \implies \sigma'(\pi) = \perp) \wedge (\forall d \in \text{wt}(t) \setminus \text{del}(t) : \sigma'(d) = \top \wedge (\forall \pi \in \Pi : \forall \bar{d} \in \ell(\pi) \setminus \{d\} : \sigma(\bar{d}) = \top) \implies \sigma'(\pi) \in \{\top, \text{F}\}) \}$ and is denoted by $c \xrightarrow{t} C$.*

Example 4. Consider transition `calculate price` in Figure 1. Suppose there is a token in place `p1`. Transition `calculate price` is enabled if data element `gds` is defined. Firing this transition means that the token in `p1` is removed, and a token in `p2` and in `p3` is produced. In addition, `calculate price` takes data element `gds` as its input and stores its output in data element `price`. We implicitly assume that inside a task reading always precedes writing, and writing always precedes deleting. As `price` did not have a value before the occurrence of `calculate price`, a new value of `price` is created (otherwise it would have been updated). Moreover, as `price` is assigned to predicate `isHigh(price)`, this predicate is evaluated to either true or false, yielding two configurations.

4.2 Reachability

Definition 5 defines the semantics of firing a single transition. Now we extend the firing of a single transition to sequences of transitions. In other words, we define the set of reachable configurations of N . To take into account that we do not know the concrete values of predicates a priori, we define may- and must-reachability. *May-reachability guarantees that the reachability holds in at least one data refinement, whereas must-reachability guarantees that the reachability holds in every data refinement.*

Given a configuration c , a *may-step* from c specifies the existence of a successor configuration c' of c . Accordingly, a *may-path* of length n specifies the existence of a sequence of n may-steps from c to a configuration c' . In this case, c' is *may-reachable* from c . A *must hyper-path* of length n from c defines the set C of all configurations c' such that a may-path of length n exists from c to c' . In case there exists a may-path of length $n - 1$ from c to a configuration c'' , and c'' has no successor configuration (i.e., c'' is a *deadlock*), then c'' is also contained in the set C of configurations being reachable via a may-path of length n . In

other words, a must hyper-path of length n contains both the configurations that are reachable from c via a may-path of length n and all the deadlocks that are may-reachable from c . We refer to the set C as the set of configurations that are *must-reachable* from c .

Whenever a configuration c has due to the data abstraction more than one successor configuration, may-reachability considers always one successor—that is, it considers only *one* data refinement. In contrast, a must hyper-path contains all successor configurations of c . Hence, it considers *all* possible data refinements.

Definition 6 (Reachability). *Let $N = \langle P, T, F, rd, wt, del, grd \rangle$ be a WFD-net, c, c' be configurations of N , and $C, C' \subseteq \Xi$ be sets of configurations of N .*

- *A set C of configurations is reachable from a configuration c , denoted by $c \rightarrow C$, if and only if there is a transition $t \in T$ being enabled at c and the firing of t yields C (i.e., $c \xrightarrow{t} C$).*
- *There is a may-step from a configuration c to a configuration c' , denoted by $c \rightarrow_{may} c'$, if and only if c' is an element of a set C of configurations that is reachable from c .*
- *A may-path (of length n) from a configuration c is a sequence of configurations c_1, \dots, c_n of N , $n \geq 0$, where $c_1 = c$ and $c_i \rightarrow_{may} c_{i+1}$ for every $i = 1, \dots, n - 1$; we denote the existence of a may-path c_1, \dots, c_n with $c_1 = c$ and $c_n = c'$ by $c \xrightarrow{*}_{may} c'$.*
- *A must hyper-path (of length n) from a configuration c is a set of may-paths from c inductively defined as follows: $\Omega_1 = \{c\}$ and $\Omega_{i+1} = \{\omega, c' \mid \omega \in \Omega_i \wedge \exists C : c \rightarrow C \wedge c' \in C\} \cup \{\omega \mid \omega \in \Omega_i \wedge c \not\rightarrow \omega\}$ for $i = 1, \dots, n - 1$.*
- *By $c \rightarrow_{must} C$ we denote the existence of a must hyper-path Ω_n such that, for every $c' \in C$, there is a may-path $c_1, \dots, c_n \in \Omega_n$ with $c_1 = c$ and $c_n = c'$.*

Example 5. The state space of the shipper is depicted in Figure 3. From the start configuration c_0 only the singleton set $\{c_1\}$ can be reached by firing transition **receive goods**. Configuration c_1 consists of a marking $[p_1]$, and the abstraction function σ assigns the value \top to data elements c_1 , gds , and ads . Predicate **isHigh(price)** is undefined in c_1 . Transition **calculate price** is enabled at c_1 . Firing this transition yields the set $\{c_2, c_3\}$ of successor configurations. The difference between both configurations is that in c_2 predicate **isHigh(price)** is evaluated to true (denoted **isHigh**), whereas in c_3 this predicate is evaluated to false (denoted \neg **isHigh**). From $c_0 \rightarrow c_1 \rightarrow \{c_2, c_3\}$, we conclude that there is a may-step from c_0 to c_1 , a mat-step from c_1 to c_2 as well as from c_1 to c_3 . So there is a may-path of length 2 from c_0 to c_2 and from c_0 to c_3 . Consequently, $\{c_2, c_3\}$ but also $\{c_1\}$ is must-reachable from c_0 , because a must hyper-path of length 2 and 1 exists, respectively. Observe that there is also a must hyper-path (of length 5) from c_0 to $\{c_{11}, c_{12}\}$. Only configuration c_{13} is may-reachable from c_{11} . Unlike c_{11} , configuration c_{12} does not have any successor. Hence, we conclude from the definition of a must hyper-path that there exists a must hyper-path of length 6 from c_0 to $\{c_{13}, c_{12}\}$.

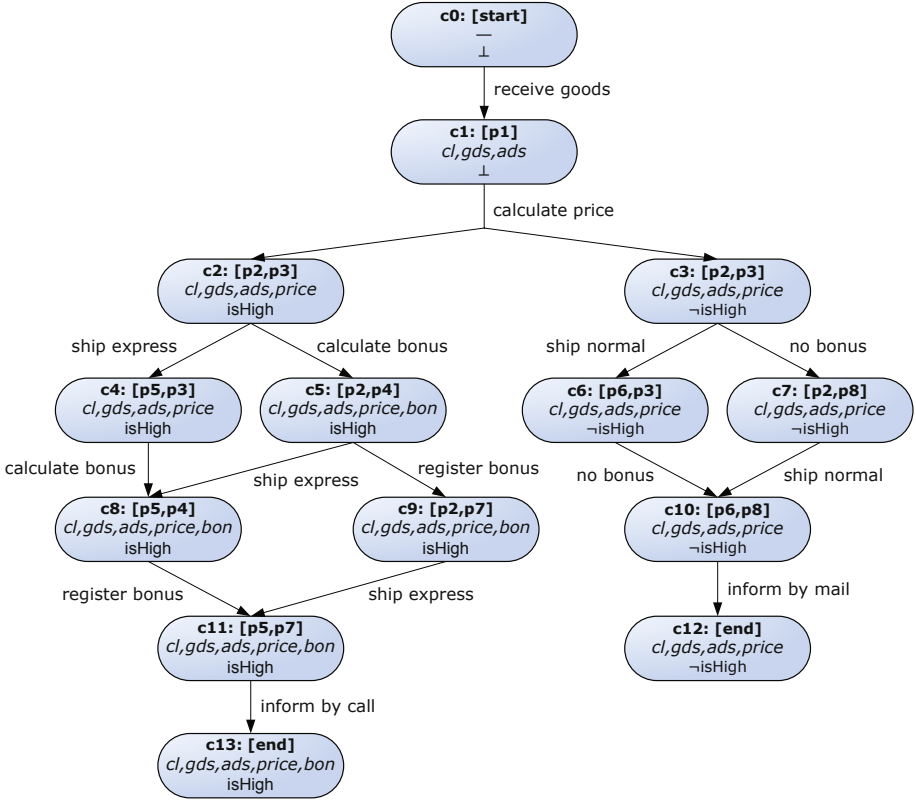


Fig. 3. State space of the shipper in Fig. 1. Each rounded rectangle specifies a configuration of the shipper. In the first line, the identifier of the configuration and the marking is depicted, the second line presents all defined data elements, and the third line evaluates the predicate $isHigh(price)$.

5 Soundness

With the help of may- and must-reachability we can formalize soundness for WFD-nets. The soundness property, originally defined for WF-nets, ensures that from any reachable marking the final marking can be reached, and every task can potentially be executed. In this section, we extend the definition of soundness to WFD-nets. We present two notions: may-soundness and must-soundness. A WFD-net is may-sound if and only if *there exists* a data refinement such that the concrete workflow model (that contains all data information) is sound. In contrast to may-soundness, the notion of must-soundness guarantees that the WFD-net is sound in *all* possible data refinements.

Definition 7 (May- and Must-soundness). Let $N = \langle P, T, F, rd, wt, del, grd \rangle$ be a WFD-net, let c_0 be the start configuration of N , and let $C_f \subseteq \Xi$ denote the set of final configurations of N . N is

- may-sound *if and only if* for every set C of configurations of N being must-reachable from the start configuration c_0 (i.e., $c_0 \rightarrow_{\text{must}} C$), there exists a configuration $c \in C$ such that a configuration $c_f \in C_f$ is may-reachable from c (i.e., $c \rightarrow_{\text{may}}^* c_f$).
- must-sound *if and only if* for every configuration c being may-reachable from the start configuration c_0 of N (i.e., $c_0 \rightarrow_{\text{may}}^* c$), there exists a set $C \subseteq C_f$ of final configurations that is must-reachable from c (i.e., $c \rightarrow_{\text{must}} C$).

May-soundness ensures that for any set C of configurations that are must-reachable from the start configuration, there exists a configuration $c \in C$ from which a final configuration is may-reachable. The set C contains all configurations that are reachable from the initial configuration in any data refinement (because must-reachability considers all may-paths). The existence of a configuration $c \in C$ from which a final configuration is may-reachable guarantees that there exists at least one data refinement of N (i.e., one may-path) in which a final marking can be reached.

Must-soundness ensures that from any configuration c that is may-reachable from the start configuration, a subset of the final configurations is must-reachable. That means, from every marking that is reachable in the WFD-net N , a final configuration can be reached in any data refinement of N (because must-reachability considers all may-paths).

Example 6. Consider again the state space of the shipper in Figure 3. As previously mentioned, there exists a must hyper-path from the start configuration c_0 to the set $\{c_{13}, c_{12}\}$ of configurations. Both configurations, c_{13} and c_{12} , are final configurations (and clearly c_{13} is may-reachable from c_{13} and c_{12} is may-reachable from c_{12}). As this is the longest must hyper-path in the state space, we conclude that from any other must hyper-path, there always exists a state from which either c_{13} or c_{12} is may-reachable. Thus, the shipper is may-sound. It can also be easily seen that from each state being may-reachable from c_0 , there exists a must hyper-path to a final configuration. Hence, we conclude that the WFD-net of the shipper is also must-sound. In other words, the soundness property holds in any data refinement of the WFD-net in Figure 1.

Let us now come back to the modification of the shipper (cf. Section 2) where different predicates (`isHighLeft(price)` and `isHighRight(price)`) are used in the left and the right part of the shipper. In this case, c_2 corresponds to a configuration where both predicates are true, and c_3 corresponds to a configuration where both predicates are false. In addition, c_1 has two more successors, say $c_{2'}$ and $c_{3'}$, corresponding to configurations where `isHighLeft(price)` is true and `isHighRight(price)` is false and vice versa. In configuration $c_{2'}$ transitions `ship express` and `no bonus` are enabled. Firing these transitions yields a configuration, say c , where the shipper is in the marking $[p_5, p_8]$. Configuration c is a deadlock, and it is may-reachable from the start configuration. Hence, there does not exist a must hyper-path from c to a final configuration, and thus the modified shipper is not must-sound. However, the modified shipper is may-sound: there exists a data refinement in which a final configuration can be reached, namely, if

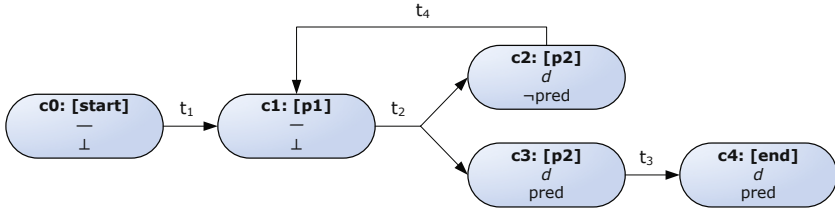


Fig. 4. State space of the example from Figure 2

the two predicates `isHighLeft(price)` and `isHighRight(price)` always evaluate to the same value.

Figure 4 shows the state space of the WFD-net from Figure 2. As there is no guarantee that both branches of t_2 will lead to proper completion (there exists an infinite sequence $c_1 \rightarrow c_2 \rightarrow c_1 \rightarrow c_2 \dots$), we conclude that this WFD-net is not must sound. However, we see that one branch of t_2 , namely the one going to c_3 , always leads to completion. Therefore, the workflow is may-sound.

6 Conclusion

In this paper we showed how to obtain reliable verification results when verifying conceptual workflow nets with data information. Our work is in fact a cross-fertilization of design and modeling frameworks coming from the field of Process-Aware Information Systems (PAIS), and verification and abstraction approaches developed in the field of Formal Methods.

The final target of researchers working in the area of formal methods is usually the verification of programs/systems which may contain complex data coming from large or infinite data domains, consist of a large number of distributed components, etc. To cope with the complexity of the objects to be verified, many abstraction techniques [4,5,6,7,10,15] such as predicate abstractions, and abstraction methodologies, such as CEGAR (counter-example guided abstraction refinement) [3,13], are proposed. The concept of may/must transition systems was defined in this area [14] and then found multiple applications there.

In our work, the verification target is not a refined system but a conceptual model, which may later on be refined in different ways. We do not need to apply abstractions to cope with the complexity of data, as it is done in [16,11,12] for WS-BPEL processes—the data is still underdefined, abstract, in the models we consider.

We use WFD-nets to specify conceptual workflows. As we showed in [19], WFD-nets can be seen as an abstraction from notations deployed by popular modeling tools, like Protos of Pallas Athena, which uses a Petri-net-based modeling notation and is a widely-used business process modeling tool. (It is used by more than 1500 organizations in more than 20 countries and is the leading

business process modeling tool in the Netherlands.) By building on the classical formalism of Petri nets, we keep our framework easily adaptable to many industrial and academic languages.

Future work. For the future work we plan to integrate our implementation of the may/must-based soundness verification of WFD-nets in the process analysis/discovery framework ProM [2]. As a basis, we use the generic CTL model-checking algorithm presented in [17], and restrict it to the soundness property. This algorithm shall then replace the standard algorithm [19] for checking soundness. As ProM provides import functionality for many industrial process modeling languages, by integrating our implementation in ProM we will achieve direct applicability of our framework to real-world conceptual workflows.

Another item for the future work concerns the diagnostic methods for the identification of possible causes of incorrectness in the workflow, which would be fit to work within the may/must framework.

References

1. van der Aalst, W.M.P.: The Application of Petri Nets to Workflow Management. *The Journal of Circuits, Systems and Computers* 8(1), 21–66 (1998)
2. van der Aalst, W.M.P., van Dongen, B.F., Günther, C.W., Mans, R.S., Alves de Medeiros, A.K., Rozinat, A., Rubin, V., Song, M., Verbeek, H.M.W., Weijters, A.J.M.M.: ProM 4.0: Comprehensive Support for Real Process Analysis. In: Kleijn, J., Yakovlev, A. (eds.) ICATPN 2007. LNCS, vol. 4546, pp. 484–494. Springer, Heidelberg (2007)
3. Clarke, E.M., Grumberg, O., Jha, S., Lu, Y., Veith, H.: Counterexample-guided Abstraction Refinement for Symbolic Model Checking. *Journ. of the ACM* 50(5), 752–794 (2003)
4. Clarke, E.M., Grumberg, O., Long, D.E.: Model Checking and Abstraction. *ACM Transactions on Programming Languages and Systems* 16(5), 1512–1542 (1994); A preliminary version appeared in the Proc. of the POPL 1992
5. Cousot, P., Cousot, R.: Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints. In: Proc. of the 4th ACM SIGACT-SIGPLAN Symp. on Principles of programming languages (POPL 1977), pp. 238–252. ACM Press, New York (1977)
6. Dams, D.: Abstract Interpretation and Partition Refinement for Model Checking. PhD dissertation, Eindhoven University of Technology (July 1996)
7. Dams, D., Gerth, R., Grumberg, O.: Abstract Interpretation of Reactive Systems. *ACM Transactions on Programming Languages and Systems (TOPLAS)* 19(2), 253–291 (1997)
8. Dehnert, J., Rittgen, P.: Relaxed soundness of business processes. In: Dittrich, K.R., Geppert, A., Norrie, M.C. (eds.) CAiSE 2001. LNCS, vol. 2068, pp. 157–170. Springer, Heidelberg (2001)
9. Dumas, M., van der Aalst, W.M.P., ter Hofstede, A.H.M.: *Process-Aware Information Systems: Bridging People and Software through Process Technology*. Wiley & Sons, Chichester (2005)
10. Graf, S., Saïdi, H.: Construction of Abstract State Graphs with PVS. In: Grumberg, O. (ed.) CAV 1997. LNCS, vol. 1254, pp. 72–83. Springer, Heidelberg (1997)

11. Heinze, T.S., Amme, W., Moser, S.: Generic CSSA-based pattern over boolean data for an improved WS-BPEL to petri net mapping. In: Mellouk, A., Bi, J., Ortiz, G., Chiu, D.K.W., Popescu, M. (eds.) Third International Conference on Internet and Web Applications and Services, ICIW 2008, Athens, Greece, June 8-13, pp. 590–595. IEEE Computer Society, Los Alamitos (2008)
12. Heinze, T.S., Amme, W., Moser, S.: A restructuring method for WS-BPEL business processes based on extended workflow graphs. In: Dayal, U., Eder, J., Koehler, J., Reijers, H.A. (eds.) BPM 2009. LNCS, vol. 5701, pp. 211–228. Springer, Heidelberg (2009)
13. Lakhnech, Y., Bensalem, S., Berezin, S., Owre, S.: Incremental Verification by Abstraction. In: Margaria, T., Yi, W. (eds.) TACAS 2001. LNCS, vol. 2031, pp. 98–112. Springer, Heidelberg (2001)
14. Larsen, K.G.: Modal specifications. In: Sifakis, J. (ed.) CAV 1989. LNCS, vol. 407, pp. 232–246. Springer, Heidelberg (1990)
15. Loiseaux, C., Graf, S., Sifakis, J., Bouajjani, A., Bensalem, S.: Property Preserving Abstractions for the Verification of Concurrent Systems. *Formal Methods in System Design* 6(1), 11–44 (1995)
16. Moser, S., Martens, A., Görlach, K., Amme, W., Godlinski, A.: Advanced verification of distributed WS-BPEL business processes incorporating CSSA-based data flow analysis. In: 2007 IEEE International Conference on Services Computing (SCC 2007), Salt Lake City, Utah, USA, July 9-13, pp. 98–105. IEEE Computer Society, Los Alamitos (2007)
17. Shoham, S., Grumberg, O.: Monotonic abstraction-refinement for ctl. In: Jensen, K., Podelski, A. (eds.) TACAS 2004. LNCS, vol. 2988, pp. 546–560. Springer, Heidelberg (2004)
18. Trčka, N.: Workflow Soundness and Data Abstraction: Some negative results and some open issues. In: Workshop on Abstractions for Petri Nets and Other Models of Concurrency (APNOC), pp. 19–25 (2009)
19. Trčka, N., van der Aalst, W.M.P., Sidorova, N.: Data-Flow Anti-Patterns: Discovering Data-Flow Errors in Workflows. In: van Eck, P., Gordijn, J., Wieringa, R. (eds.) CAiSE 2009. LNCS, vol. 5565, pp. 425–439. Springer, Heidelberg (2009)

Octavian Panel on Intentional Perspectives on Information Systems Engineering

Arne Sølvsberg

NTNU — Norwegian University of Science and Technology, Trondheim, Norway

Octavian panel is a round table-table discussion with 8 (Octavian) people on the scene and an audience. Only those people can talk who are on the scene. The talk time is restricted to 3 minutes. Panelists may leave the scene at any time to join the audience, thus opening for a new panelist on the scene. Questions from outside can be forwarded through the moderator. The moderator cannot leave the scene. The moderator's contribution is limited to 1 minute. The first two rounds of the panelists are pre-selected what does not mean that this will be the order of making contributions.

Themes for Discussion

The panel is challenged to discuss the relationships among intention oriented information system development and methods for

- goal specification
- requirements engineering
- implementation methods, e.g., agile approaches

The panel is challenged to discuss intention oriented approaches with respect to

- practical use, state-of-the-art
- cross-disciplinary systems development
- quality assurance, security
- other relevant themes that come up during the discussions

Author Index

- Aalst, Wil M.P. van der 38, 515
Almeida da Silva, Marcos Aurélio 348
- Barone, Daniele 53
Batini, Carlo 53
Ben-Abdallah, Hanène 150
Bendraou, Reda 348
Bennacer, Nacéra 224
Blanco, Lorenzo 83
Blanc, Xavier 348
Bonvin, Nicolas 302
Bouassida, Nadia 150
Bouquet, Paolo 302
Brambilla, Marco 1
Butler, Tom 281
- Cabot, Jordi 424
Castano, Silvana 194
Catasta, Michele 302
Ceri, Stefano 1
Charoy, François 135
Chiniforooshan Esfahani, Hesam 424
Chopra, Amit K. 113
Cleve, Anthony 332
Cordioli, Daniele 302
Costache, Stefania 230
Crescenzi, Valter 83
Crespo, Yania 409
- Dadam, Peter 9
Dalpiaz, Fabiano 113
Daniel, Florian 98
Davidsen, Magne 296
Desai, Nirmal 499
de Spindler, Alexandre 363
Dijkman, Remco 483
Dimov, Alexandar 266
Dorn, Christoph 393
du Bousquet, Lydie 165
Dustdar, Schahram 98, 393
- Fankhauser, Peter 302
Ferrara, Alfio 194
Fielt, Erwin 129
- Finnegan, Patrick 281
Franch, Xavier 439
- Gaaloul, Khaled 135
Gaugaz, Julien 302
Giorgini, Paolo 113
Godart, Claude 135
Grossniklaus, Michael 363
- Habra, Naji 236
Hainaut, Jean-Luc 317, 332
Holmes, Ta'id 98
Hordijk, Wiebe 68
- Ilieva, Sylvia 266
Indulska, Marta 251
Ioannou, Ekaterini 302
- Jarke, Matthias 470
Jian, Jingxian 515
- Kamseu, Flora 236
Kiely, Gaye 281
Kohlborn, Thomas 129
Korthaus, Axel 129
Koshutanski, Hristo 302
Krusteva, Iva 266
Krcmar, Helmut 129
Krogstie, John 296
- Laguna, Miguel A. 409
Lemaitre, Jonathan 317
Leonardi, Chiara 455
Leone, Stefania 363
Loucopoulos, Pericles 470
Luebeck, Christian 129
Ly, Linh Thao 9
Lyytinen, Kalle 470
- Maña, Antonio 302
Marqués, José M. 409
Matulevičius, Raimundas 236
Mending, Jan 483, 499
Merialdo, Paolo 83
Miklós, Zoltán 302

- Montanelli, Stefano 194
Mougenot, Alix 348
Mrabet, Yassine 224
Mylopoulos, John 113, 470
- Nejdl, Wolfgang 230
Niederée, Claudia 230, 302
Norrie, Moira C. 363
- Palpanas, Themis 302
Papotti, Paolo 83
Paraszcak, Jurij 7
Pelechano, Vicente 378
Pernelle, Nathalie 224
Pesic, Maja 38
Polyvyanyy, Artem 499
- Ramdoyal, Ravi 332
Rassadko, Nataliya 180
Riedl, Christoph 129
Rinderle-Ma, Stefanie 9
Rittgen, Peter 24
Robinson, William 470
Rosemann, Michael 129
- Sabatucci, Luca 455
Sadiq, Shazia 251
Schonenberg, Helen 515
- Sen, Sinan 209
Serral, Estefanía 378
Sidorova, Natalia 515, 530
Sindre, Guttorm 165
Sølvberg, Arne 545
Song, Minseok 38
Stålhane, Tor 165
Stahl, Christian 530
Stecher, Rodolfo 230
Stella, Fabio 53
Stoermer, Heiko 180, 302
Stojanovic, Nenad 209
Susi, Angelo 455
Syed Abdullah, Norris 251
- Thiam, Mouhamadou 224
Trčka, Nikola 530
- Vaidya, Nachiket 180
Valderas, Pedro 378
- Weidlich, Matthias 483, 499
Wieringa, Roel 68
- Yu, Eric 424
- Zahoor, Ehtesham 135
Zancanaro, Massimo 455
Zdun, Uwe 98