

Pascal van Eck  
Jaap Gordijn  
Roel Wieringa (Eds.)

LNCS 5565

# Advanced Information Systems Engineering

21st International Conference, CAiSE 2009  
Amsterdam, The Netherlands, June 2009  
Proceedings

**CAiSE'09**  
Amsterdam

 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

*University of Dortmund, Germany*

Madhu Sudan

*Massachusetts Institute of Technology, 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*

Pascal van Eck Jaap Gordijn  
Roel Wieringa (Eds.)

# Advanced Information Systems Engineering

21st International Conference, CAiSE 2009  
Amsterdam, The Netherlands, June 8-12, 2009  
Proceedings

Volume Editors

Pascal van Eck  
Roel Wieringa  
University of Twente  
Department of Computer Science  
P.O. Box 217, 7500 AE Enschede, The Netherlands  
E-mail: {p.vaneck,r.j.wieringa}@utwente.nl

Jaap Gordijn  
VU University  
Department of Computer Science  
De Boelelaan 1081, 1081 HV Amsterdam, The Netherlands  
E-mail: gordijn@cs.vu.nl

Library of Congress Control Number: Applied for

CR Subject Classification (1998): H.2, H.3-5, J.1, K.4.3-4, K.6, D.2, I.2.11

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

ISSN 0302-9743  
ISBN-10 3-642-02143-3 Springer Berlin Heidelberg New York  
ISBN-13 978-3-642-02143-5 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 2009  
Printed in Germany

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India  
Printed on acid-free paper SPIN: 12690534 06/3180 5 4 3 2 1 0



# Preface

Starting in the late 1980s, the CAiSE series of conferences has established a platform for presenting and exchanging results of design-oriented research in information systems. In addition to the presentation of new information systems techniques, recent years have seen the rise of empirical validation of such techniques. There is also increasing attention for industry participation. The 21st CAiSE conference, held in Amsterdam, The Netherlands, during June 8–12, 2009, continued this tradition.

The theme of CAiSE 2009 was “Information Systems for Business Innovation.” Due to the widespread use of the Web, businesses innovate their propositions to customers and come up with new IT-enabled services. Such innovation requires understanding of business and technology in an integrated way. Multi-disciplinary research areas such as service science, networked enterprises, and social networking are paying attention to IT and business innovation. This theme was evident both in the pre-conference workshops and in the invited speakers of the conference.

The first two days consisted of pre-conference workshops on business process modelling, modelling methods, requirements engineering, organizational modelling, interoperability and cooperation, the knowledge industry, ontologies, governance, Web information systems, business-IT alignment, legal aspects, systems of things and domain engineering. The conference proper was combined with a doctoral consortium where PhD students could present and discuss their research plans and with an industrial event with presentations and exhibitions.

Four invited speakers shed light on the role of ontologies in business, process mining, business networking and IT entrepreneurship. Highlights of the conference included a concert and dinner in the world-famous *Concertgebouw* building and a reception in the *Muziekgebouw aan het IJ* in Amsterdam harbor.

We thank all Program Committee members and all additional reviewers who put in their time and effort to make this once again an excellent conference. Each submission was reviewed by at least three reviewers. In addition, there was a program board whose members acted as assistant Program Committee Chairs and who coordinated on-line discussion among the reviewers of each paper. The program board met in January 2009 in Amsterdam to select papers based on the reviews and on-line discussions. Out of 230 submitted papers, 36 (16%) were accepted for the main conference. An additional 23 (10%) were accepted for the CAiSE Forum. We extend our thanks to everyone involved in this process.

We are also grateful to all local organizers for managing the complex coordination involved in organizing a conference and extend our thanks to our sponsors

who made the event financially feasible. Finally, we thank the participants and hope that they look back on another rewarding and inspiring CAiSE conference.

April 2009

Pascal van Eck  
Jaap Gordijn  
Roel Wieringa

# Organization

Advisory Committee	Janis Bubenko Jr. Royal Institute of Technology, Sweden Colette Rolland Université Paris 1 Panthéon Sorbonne, France Arne Sølvsberg Norwegian University of Science and Technology, Norway
General Chair	Roel Wieringa University of Twente, The Netherlands
Program Chair	Jaap Gordijn VU University Amsterdam, The Netherlands
Doctoral Consortium Chairs	Hans Weigand University of Tilburg, The Netherlands Sjaak Brinkkemper University of Utrecht, The Netherlands
Forum Chair	Eric Yu University of Toronto, Canada
Workshop and Tutorial Chairs	Paul Johannesson KTH Stockholm, Sweden Eric Dubois CRP Henri Tudor, Luxembourg
Industrial Event and Exhibition Chairs	Erik Proper Capgemini and Radboud University Nijmegen, The Netherlands Bas van der Raadt Capgemini, The Netherlands Nico Lassing Accenture, The Netherlands
Sponsorship Chair	Ellen Schulten VU University Amsterdam, The Netherlands
Publications Chair	Pascal van Eck University of Twente, The Netherlands



## Pre-conference Workshops

10th International Workshop on Business Process Modeling, Development and Support (BPMDS 2009)

*Selmin Nurcan, Rainer Schmidt, Pnina Soffer, Roland Ukor*

14th International Conference on Exploring Modeling Methods for Systems Analysis and Design (EMMSAD 2009)

*John Krogstie, Erik Proper, Terry Halpin*

15th International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ 2009)

*Martin Glinz, Patrick Heymans*

5th International Workshop on Enterprise and Organizational Modeling and Simulation (EOMAS 2009)

*Johann Kinghorn, Srinu Ramaswamy*

5th International Workshop on Cooperation and Interoperability - Architecture and Ontology (CIAO! 2009)

*Antonia Albani, Jan Dietz*

International Workshop on Knowledge Industry Survival Strategy (KISS 2009)

*Jorn Bettin, Tony Clark, Keith Duddy, Derek Roos*

Third International Workshop on Ontology, Conceptualization and Epistemology for Information Systems, Software Engineering and Service Science (ONTOSE 2009)

*Christian Kop, Miguel-Angel Sicilia, Fabio Sartori*

Second International Workshop on Governance, Risk and Compliance in Information Systems (GRCIS 2009)

*Shazia Sadiq, Marta Indulska, Michael zur Muehlen*

6th International Workshop on Web Information Systems Modeling (WISM 2009)

*Flavius Frasincar, Geert-Jan Houben, Philippe Thiran*

4th International Workshop on Business IT Alignment and Interoperability (BUSITAL 2009)

*Hans Weigand, Hannes Werthner, Graham Gal*

International Workshop on Legal Aspects of Information Systems (LAoIS 2009)

*Kamalakar Karlapalem, Eleanna Kafeza, Irene Kafeza*

International Workshop on Value-Driven Engineering of Systems of Things (VEST 2009)

*Camille Salinesi, Gianluigi Viscusi*

First International Workshop on Domain Engineering (DE@CAiSE 2009)

*Iris Reinhartz-Berger, Arnon Sturm, Yair Wand*

## Program Committee Board

Hans Akkermans, The Netherlands  
Sjaak Brinkkemper, The Netherlands  
Eric Dubois, Luxembourg  
Johann Eder, Austria  
Pericles Loucopoulos, UK  
Andreas Opdahl, Norway  
Oscar Lopez Pastor, Spain

Barbara Pernici, Italy  
Anne Persson, Sweden  
Klaus Pohl, Germany  
Colette Rolland, France  
Camille Salinesi, France  
Pnina Soffer, Israel

## Program Committee

Wil van der Aalst, The Netherlands  
Pär Ågerfalk, Sweden  
Jacky Akoka, France  
Marco Bajec, Slovenia  
Luciano Baresi, Italy  
Zorah Bellahsene, France  
Boalem Benatallah, Australia  
Giuseppe Berio, Italy  
Claudio Bettini, Italy  
Nacer Boudjlida, France  
Mokrane Bouzeghoub, France  
Fabio Casati, Italy  
Silvana Castano, Italy  
Jaelson Castro, Brazil  
Corinne Cauvet, France  
João Falcão Cunha, Portugal  
Marlon Dumas, Estonia  
Joerg Evermann, Canada  
Xavier Franch, Spain  
Paolo Giorgini, Italy  
Claude Godart, France  
Mohand-Said Hacid, France  
Terry Halpin, USA  
Brian Henderson-Sellers, Australia  
Patrick Heymans, Belgium  
Matthias Jarke, Germany  
Manfred Jeusfeld, The Netherlands  
Paul Johannesson, Sweden  
Henk Jonkers, The Netherlands  
Havard Jorgensen, Norway  
Roland Kaschek, New Zealand  
Marite Kirkova, Latvia  
John Krogstie, Norway

Patricia Lago, The Netherlands  
Régina Laleau, France  
Marc Lankhorst, The Netherlands  
Wilfried Lemahieu, Belgium  
Michel Leonard, Switzerland  
Kalle Lyytinen, USA  
Isabelle Mirbel, France  
Haris Mouratidis, UK  
John Mylopoulos, Canada  
Moirra Norrie, Switzerland  
Andreas Oberweis, Germany  
Antoni Olivé, Spain  
Barbara Paech, Germany  
Herve Panetto, France  
Jeffrey Parsons, Canada  
Michael Petit, Belgium  
Yves Pigneur, Switzerland  
Geert Poels, Belgium  
Erik Proper, The Netherlands  
Jolita Ralyté, Switzerland  
Björn Regnell, Sweden  
Manfred Reichert, Germany  
Mart Roantree, Ireland  
Michael Rosemann, Australia  
Gustavo Rossi, Argentina  
Matti Rossi, Finland  
Motoshi Saeki, Japan  
Camille Salinesi, France  
Tony C. Shan, USA  
Keng Siau, USA  
Guttorm Sindre, Norway  
Monique Snoeck, Belgium  
Janis Stirna, Sweden

Arnon Sturm, Israel  
 Alistair Sutcliffe, UK  
 Stefan Tai, USA  
 David Taniar, Australia  
 Bernhard Thalheim, Germany  
 Farouk Toumani, France  
 Olga de Troyer, Belgium  
 Aphrodite Tsalgatidou, Greece  
 Jean Vanderdonckt, Belgium

Olegas Vasilecas, Lithuania  
 Yair Wand, Canada  
 Mathias Weske, Germany  
 Hans Weigand, The Netherlands  
 Roel Wieringa, The Netherlands  
 Carson Woo, Canada  
 Eric Yu, Canada  
 Konstantinos Zachos, UK  
 Didar Zowghi, Australia

## Additional Referees

Sudhir Agarwal  
 Fernanda Alencar  
 Nicolas Arni-Bloch  
 George Athanasopoulos  
 Ahmed Awad  
 Ladjel Bellatreche  
 Fredrik Bengtsson  
 Nicholas Berente  
 Maria Bergholtz  
 Richard Berntsson-Svensson  
 Lianne Bodenstaff  
 Remco de Boer  
 Lars Borner  
 Quentin Boucher  
 Jordi Cabot  
 Sven Casteleyn  
 Andreas Classen  
 Andre van Cleeff  
 Chad Coulin  
 Maya Daneva  
 Wilco Engelsman  
 Alfio Ferrara  
 Benoît Fraikin  
 Virginia Franqueira  
 Dario Freni  
 Štefan Furlan  
 Frederic Gervais  
 Françoise Gire  
 Christophe Gnaho  
 Daniela Grigori  
 Alexander Grosskopf  
 Qing Gu  
 Adnene Guabtni

Martin Henkel  
 Marcel Hiel  
 Arnaud Hubaux  
 Ela Hunt  
 Helene Jaudoin  
 Zoubida Kedad  
 Oleg Koffmane  
 Woralak Kongdenfha  
 Stefan Lamparter  
 Algirdas Laukaitis  
 Dejan Lavbič  
 Juho Lindman  
 Annabella Loconsole  
 Kajsa Lorentzon  
 Dewi Mairiza  
 Amel Mammar  
 Sergio Mascetti  
 Zafar Mehboob  
 Stefano Montanelli  
 Dr. Mahmood Niazi  
 Nurie Nurmuliani  
 Martin F. O'Connor  
 Horst Pichler  
 Michael Pantazoglou  
 Linda Pareschi  
 Emilian Pascalau  
 Bram Pellens  
 Artem Polyvyanyy  
 Rahul Premraj  
 Ricardo Argenton Ramos  
 Maryam Razavian  
 Daniele Riboni  
 Mohsen Rouached

Seung Ryu  
Jürgen Rückert  
Ove Sörensen  
Khalid Saleem  
Samiaji Sarosa  
Farida Semmak  
Patricia Silveira  
Marten van Sinderen  
Jonas Sjöström  
Sergey Smirnov  
Sergejus Sosunovas

Lovro Šubelj  
Evi Syukur  
Christer Thörn  
Christina Tsagkani  
Gaia Varese  
Damjan Vavpotič  
Hans van Vliet  
Krzysztof Wnuk  
William Van Woensel  
Andreas Wombacher



## Sponsors



*vrije Universiteit amsterdam*



**University of Twente**  
*The Netherlands*



The Network Institute



# Table of Contents

## Keynotes

The Science of the Web . . . . .	1
<i>Nigel Shadbolt</i>	
TomTom for Business Process Management (TomTom4BPM) . . . . .	2
<i>Wil M.P. van der Aalst</i>	
Computer-Centric Business Operating Models vs. Network-Centric Ones . . . . .	6
<i>Mark de Simone</i>	
The IT Dilemma and the Unified Computing Framework . . . . .	8
<i>Edwin Paalvast</i>	
Tutorial: How to Value Software in a Business, and Where Might the Value Go? . . . . .	9
<i>Gio Wiederhold</i>	
Towards the Next Generation of Service-Based Systems: The S-Cube Research Framework . . . . .	11
<i>Andreas Metzger and Klaus Pohl</i>	

## Model Driven Engineering

An Extensible Aspect-Oriented Modeling Environment . . . . .	17
<i>Naoyasu Ubayashi, Genya Otsubo, Kazuhide Noda, and Jun Yoshida</i>	
Incremental Detection of Model Inconsistencies Based on Model Operations . . . . .	32
<i>Xavier Blanc, Alix Mougenot, Isabelle Mounier, and Tom Mens</i>	
Reasoning on UML Conceptual Schemas with Operations . . . . .	47
<i>Anna Queralt and Ernest Teniente</i>	

## Conceptual Modelling 1

Towards the Industrialization of Data Migration: Concepts and Patterns for Standard Software Implementation Projects . . . . .	63
<i>Klaus Haller</i>	

Defining and Using Schematic Correspondences for Automatically  
 Generating Schema Mappings ..... 79  
*Lu Mao, Khalid Belhajjame, Norman W. Paton, and  
 Alvaro A.A. Fernandes*

The Problem of Transitivity of Part-Whole Relations in Conceptual  
 Modeling Revisited ..... 94  
*Giancarlo Guizzardi*

**Conceptual Modelling 2**

Using UML as a Domain-Specific Modeling Language: A Proposal for  
 Automatic Generation of UML Profiles ..... 110  
*Giovanni Giachetti, Beatriz Marín, and Oscar Pastor*

Verifying Action Semantics Specifications in UML Behavioral Models ... 125  
*Elena Planas, Jordi Cabot, and Cristina Gómez*

Using Macromodels to Manage Collections of Related Models ..... 141  
*Rick Salay, John Mylopoulos, and Steve Easterbrook*

**Quality and Data Integration**

A Case Study of Defect Introduction Mechanisms ..... 156  
*Arbi Ghazarian*

Measuring and Comparing Effectiveness of Data Quality Techniques .... 171  
*Lei Jiang, Daniele Barone, Alex Borgida, and John Mylopoulos*

Improving Model Quality Using Diagram Coverage Criteria ..... 186  
*Rick Salay and John Mylopoulos*

**Goal-Oriented Requirements Engineering**

A Method for the Definition of Metrics over  $i^*$  Models ..... 201  
*Xavier Franch*

Preference Model Driven Services Selection ..... 216  
*Wenting Ma, Lin Liu, Haihua Xie, Hongyu Zhang, and Jinglei Yin*

Secure Information Systems Engineering: Experiences and Lessons  
 Learned from Two Health Care Projects ..... 231  
*Haralambos Mouratidis, Ali Sunyaev, and Jan Jurjens*

**Requirements and Architecture**

An Architecture for Requirements-Driven Self-reconfiguration ..... 246  
*Fabiano Dalpiaz, Paolo Giorgini, and John Mylopoulos*

Automated Context-Aware Service Selection for Collaborative Systems .....	261
<i>Hong Qing Yu and Stephan Reiff-Marganiec</i>	
Development Framework for Mobile Social Applications .....	275
<i>Alexandre de Spindler, Michael Grossniklaus, and Moira C. Norrie</i>	
<b>Service Orientation</b>	
Evolving Services from a Contractual Perspective .....	290
<i>Vasilios Andrikopoulos, Salima Benbernou, and Mike P. Papazoglou</i>	
Efficient IR-Style Search over Web Services .....	305
<i>Yanan Hao, Jinli Cao, and Yanchun Zhang</i>	
Towards a Sustainable Services Innovation in the Construction Sector .....	319
<i>Sylvain Kubicki, Eric Dubois, Gilles Halin, and Annie Guerriero</i>	
<b>Web Service Orchestration</b>	
P2S: A Methodology to Enable Inter-organizational Process Design through Web Services .....	334
<i>Devis Bianchini, Cinzia Cappiello, Valeria De Antonellis, and Barbara Pernici</i>	
Composing Time-Aware Web Service Orchestration .....	349
<i>Horst Pichler, Michaela Wenger, and Johann Eder</i>	
Asynchronous Timed Web Service-Aware Choreography Analysis .....	364
<i>Nawal Guermouche and Claude Godart</i>	
<b>Value-Driven Modelling</b>	
Evaluation Patterns for Analyzing the Costs of Enterprise Information Systems .....	379
<i>Bela Mutschler and Manfred Reichert</i>	
Using the REA Ontology to Create Interoperability between E-Collaboration Modeling Standards .....	395
<i>Frederik Gailly and Geert Poels</i>	
Value-Based Service Modeling and Design: Toward a Unified View of Services .....	410
<i>Hans Weigand, Paul Johannesson, Birger Andersson, and Maria Bergholtz</i>	

**Workflow**

Data-Flow Anti-patterns: Discovering Data-Flow Errors in Workflows ..... 425  
*Nikola Trčka, Wil M.P. van der Aalst, and Natalia Sidorova*

Process Algebra-Based Query Workflows ..... 440  
*Thomas Hornung, Wolfgang May, and Georg Lausen*

ETL Workflow Analysis and Verification Using Backwards Constraint Propagation ..... 455  
*Jie Liu, Senlin Liang, Dan Ye, Jun Wei, and Tao Huang*

**Business Process Modelling**

The Declarative Approach to Business Process Execution: An Empirical Test ..... 470  
*Barbara Weber, Hajo A. Reijers, Stefan Zugal, and Werner Wild*

Configurable Process Models: Experiences from a Municipality Case Study ..... 486  
*Florian Gottschalk, Teun A.C. Wagemakers, Monique H. Jansen-Vullers, Wil M.P. van der Aalst, and Marcello La Rosa*

Business Process Modeling: Current Issues and Future Challenges ..... 501  
*Marta Indulska, Jan Recker, Michael Rosemann, and Peter Green*

**Requirements Engineering**

Deriving Information Requirements from Responsibility Models ..... 515  
*Ian Sommerville, Russell Lock, Tim Storer, and John Dobson*

Communication Analysis: A Requirements Engineering Method for Information Systems ..... 530  
*Sergio España, Arturo González, and Óscar Pastor*

Spectrum Analysis for Quality Requirements by Using a Term-Characteristics Map ..... 546  
*Haruhiko Kaiya, Masaaki Tanigawa, Shunichi Suzuki, Tomonori Sato, and Kenji Kaijiri*

**Author Index** ..... 561

# The Science of the Web

Nigel Shadbolt

School of Electronics and Computer Science  
University of Southampton  
Southampton SO17 1BJ  
United Kingdom

**Abstract.** Since its inception, the World Wide Web has changed the ways people communicate, collaborate, and educate. There is, however, a growing realization among many researchers that a clear research agenda aimed at understanding the current, evolving, and potential Web is needed. A comprehensive set of research questions is outlined, together with a sub-disciplinary breakdown, emphasising the multi-faceted nature of the Web, and the multi-disciplinary nature of its study and development. These questions and approaches together set out an agenda for Web Science — a science that seeks to develop, deploy, and understand distributed information systems, systems of humans and machines, operating on a global scale.

When we discuss an agenda for a science of the Web, we use the term “science” in two ways. Physical and biological science analyzes the natural world, and tries to find microscopic laws that, extrapolated to the macroscopic realm, would generate the behaviour observed. Computer science, by contrast, though partly analytic, is principally synthetic: it is concerned with the construction of new languages and algorithms in order to produce novel desired computer behaviours. Web science is a combination of these two features. The Web is an engineered space created through formally specified languages and protocols. However, because humans are the creators of Web pages and links between them, their interactions form emergent patterns in the Web at a macroscopic scale. These human interactions are, in turn, governed by social conventions and laws. Web science, therefore, must be inherently interdisciplinary; its goal is to both understand the growth of the Web and to create approaches that allow new powerful and more beneficial patterns to occur. Finally, the Web as a technology is essentially socially embedded; therefore various issues and requirements for Web use and governance are also reviewed.

# TomTom for Business Process Management (TomTom4BPM)

Wil M.P. van der Aalst

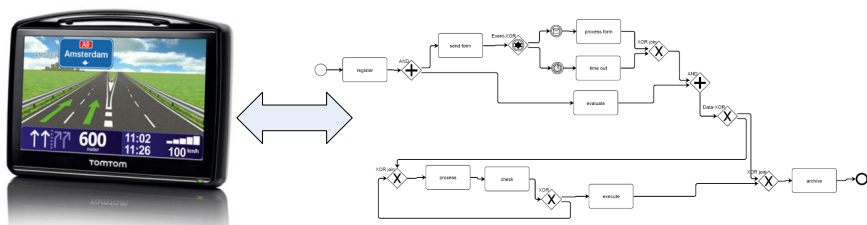
Eindhoven University of Technology  
P.O. Box 513, NL-5600 MB, Eindhoven, The Netherlands  
w.m.p.v.d.aalst@tue.nl

**Abstract.** Navigation systems have proven to be quite useful for many drivers. People increasingly rely on the devices of TomTom and other vendors and find it useful to get directions to go from A to B, know the expected arrival time, learn about traffic jams on the planned route, and be able to view maps that can be customized in various ways (zoom-in/zoom-out, show fuel stations, speed limits, etc.). However, when looking at business processes, such information is typically lacking. Good and accurate “maps” of business process are often missing and, if they exist, they tend to be restrictive and provide little information. For example, very few business process management systems are able to predict when a case will complete. Therefore, we advocate more TomTom-like functionality for business process management (TomTom4BPM). Process mining will play an essential role in providing TomTom4BPM as it allows for process discovery (generating accurate maps), conformance checking (comparing the real processes with the modeled processes), and extension (augmenting process models with additional/dynamic information).

## 1 The Need for Process Navigation

Business Process Management Systems (BPMSs) [15, 8] are used to manage and execute operational processes involving people, applications, and/or information sources on the basis of process models. These systems can be seen as the next generation of workflow technology offering more support for analysis. Despite significant advances in the last decade, the functionality of today’s BPMSs leaves much to be desired. This becomes evident when comparing such systems with the latest car navigation systems of TomTom that provide detailed maps, real-time traffic information, re-routing, customized points of interest, estimated arrival times, etc. (cf. Figure 1). Some examples of TomTom-like functionality that is generally missing are listed below:

- In today’s organizations often *a good process map is missing*. Process models are not present, incorrect, or outdated. Sometimes process models are used to directly configure the BPMS. However, in most situations there is not an explicit process model as the process is fragmented and hidden inside legacy code, the configuration of ERP systems, and in the minds of people.



**Fig. 1.** Comparing maps in a navigation system with maps in a BPMS

- If process models exist in an explicit form, *their quality typically leaves much to be desired*. Especially when a process model is not used for enactment and is only used for documentation and communication, it tends to present a “PowerPoint reality”. Road maps are typically of much higher quality and use intuitive colors and shapes of varying sizes, e.g., highways are emphasized by thick colorful lines and dirt roads are not shown or shown using thin dark lines. In process models, *all activities tend to have the same size and color and it is difficult to distinguish the main process flow from the less traveled process paths*.
- Most process modeling languages have a static decomposition mechanism (e.g., nested subprocesses). However, what is needed are controls allowing users to *zoom in or zoom out seamlessly like in a navigation system or Google maps*. Note that, while zooming out, insignificant things are either left out or dynamically clustered into aggregate shapes (e.g., streets and suburbs amalgamate into cities). Process models should not be static but allow for various views.
- Sometimes process models are used for enactment. However, such “process maps” are controlling the users. When using a car navigation system, the driver is always in control, i.e., the road map (or TomTom) is not trying to “control” the user. The goal of a BPMS should be to *provide directions and guidance rather than enforcing a particular route*.
- A navigation system continuously shows a clear *overview of the current situation* (i.e., location and speed). Moreover, traffic information is given, showing potential problems and delays. This information is typically missing in a BPMS. Even if the BPMS provides a management dashboard, TomTom-like features such as traffic information and current location are typically not shown in an intuitive manner.
- A TomTom system *continuously recalculates* the route, i.e., the recommended route is not fixed and changed based on the actions of the driver and contextual information (e.g. traffic jams). Moreover, at any point in time the navigation system is showing the *estimated arrival time*. Existing BPMSs are not showing this information and do not recalculate the optimal process based on new information.



The above list of examples illustrates desirable functionality that is currently missing in commercial BPMSs. Fortunately, recent breakthroughs in *process mining* may assist in realizing TomTom-like functionality for business process management (TomTom4BPM).

## 2 Process Mining

Process mining techniques attempt to extract non-trivial and useful information from *event logs* [2,3]. Many of today's information systems are recording an abundance of events in such logs. Various process mining approaches make it possible to uncover information about the processes they support. Typically, these approaches assume that it is possible to sequentially record events such that each event refers to an activity (i.e., a well-defined step in the process) and is related to a particular case (i.e., a process instance). Furthermore, some mining techniques use additional information such as the performer or originator of the event (i.e., the person/resource executing or initiating the activity), the timestamp of the event, or data elements recorded with the event (e.g., the size of an order).

Process mining addresses the problem that most people have very limited information about what is actually happening in their organization. In practice, there is often a significant gap between what is prescribed or supposed to happen, and what *actually* happens. Only a concise assessment of the organizational reality, which process mining strives to deliver, can help in verifying process models, and ultimately be used in a process redesign effort or BPMS implementation.

Some examples of questions addressed by process mining:

- *Process discovery*: “What is really happening?”
- *Conformance checking*: “Do we do what was agreed upon?”
- *Performance analysis*: “Where are the bottlenecks?”
- *Process prediction*: “Will this case be late?”
- *Process improvement*: “How to redesign this process?”

These examples show that process mining is an important enabler for TomTom4BPM, i.e., TomTom-like functionality for business processes. This can be demonstrated by looking at the functionality of *ProM* [2].

- ProM's *Fuzzy Miner* [6] can discover processes from event logs and offers a seamless zoom similar to TomTom or Google Maps.
- ProM's *Recommendation Engine* [7] learns from historic data and uses this to provide recommendations to the user. This way the workflow system can provide more flexibility while still supporting the user. This is comparable to the directions given by a navigation system.
- ProM's *Prediction Engine* [4] also learns from historic data but now uses this information to make predictions, e.g., the estimated completion time of a case or the likelihood the occurrence of a particular activity.

The interested reader is referred to [www.processmining.org](http://www.processmining.org) for more information about these ideas and for downloading the *ProM* software.

## References

1. van der Aalst, W.M.P.: Business Process Management Demystified: A Tutorial on Models, Systems and Standards for Workflow Management. In: Desel, J., Reisig, W., Rozenberg, G. (eds.) Lectures on Concurrency and Petri Nets. LNCS, vol. 3098, pp. 1–65. Springer, Heidelberg (2004)
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., 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)
4. 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.) CoopIS 2008, OTM 2008, Part I. LNCS, vol. 5331, pp. 319–336. Springer, Heidelberg (2008)
5. 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)
6. Günther, C.W., van der Aalst, W.M.P.: Fuzzy Mining: Adaptive Process Simplification Based on Multi-perspective Metrics. In: Alonso, G., Dadam, P., Rosemann, M. (eds.) BPM 2007. LNCS, vol. 4714, pp. 328–343. Springer, Heidelberg (2007)
7. 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)
8. Weske, M.: Business Process Management: Concepts, Languages, Architectures. Springer, Berlin (2007)

# Computer-Centric Business Operating Models vs. Network-Centric Ones

Mark de Simone

Chief Sales and Business Development Officer  
CORDYS  
Abbey House, Wellington Way – Brooklands Business Park  
Surrey KT13 0TT, Weybridge  
United Kingdom

## The Rise of the Cloud Business Operation Platform

For the first time ever, the centrality of the computer based application model is being challenged by a fast emerging new model fueled by 6 important changes which have now reached an intensity which is impossible to deny.

### 1. Suppliers Declare New Battle Fronts

Cisco has launched a vision which builds on the network centrality of future ICT models aggregating a value chain around a network and collaborative web application architecture. This is clearly in sharp contrast with the legacy approach of the computing centric model of HP and IBM. With VMware and Cordys clearly positioned to enable this acceleration of virtualization of both hardware infrastructure as well as software infrastructure, Cisco is focusing all its considerable resources to launch what it describes as “Unified Computing”. In what amounts to an apparent declaration that the network centric application model is far more efficient, effective and more responsive to today’s business needs than yesterday’s computer centric application models, Cisco has now challenged the traditional application development and deployment framework so common to the IBM and HP ecosystems. From Oracle to SAP to Microsoft, the virtualized application utility heralded by WebEx and Salesforce.com is far more at home with the Cisco vision of ICT than either IBM’s or HP’s. And the software industry has been served a formidable challenge greater than the one which the Web served Microsoft in the 1990s.

### 2. Time is the Ultimate Challenge

The new race to the post credit-crunch / economic meltdown world requires companies of all sizes to reinvent themselves, merge, reorganize, acquire, divest, change business and regulatory operating mechanisms on the fly and with full involvement of business leaders’ decision making authorities at all levels. The legacy model of designing in detail the specification requirements of the business process and having specialists with application platform proprietary languages know-how (4GL) to translate the requirements into a

technical customization of the traditional platform is now simply too onerous and long winded.

### 3. **Integration Becomes Innovation. The Rise of Cloudsourcing**

No one can afford to spend any money on integration services to just enable the association of data and processes from one application to another. The speed and degree of integration needs is just too large to be addressed through a service model with variable costs associated to the number of people-days required in traditional application infrastructures. The business operating platform used must be able to integrate any structured and non structured data and application to any defined process in real time. The Cordys Enterprise Cloud Orchestration System has created a new benchmark in the execution of the people intensive services that need to be concentrated on innovation of businesses processes, not integration. This is changing the profile and business models of Professional Services companies and outsourcing companies which now need to create more value for companies than just cost arbitrage on system integration or skills availability.

### 4. **Lower Investments and Lower Operating Expenses**

Every senior executive is now asking for less expense at both Capex and Opex levels. Furthermore whatever project is selected, the need for the payback to be within a quarter so as to be able to have a maximum deployment time of a quarter, a payback of a quarter and two quarters of improved operating performance. This new framework cannot be realized with the traditional models. The new model is a combination of Cloud Services and seamless business operation platform orchestration of processes across legacy environments.

### 5. **Today's Titans or Tomorrow's?**

History is being written very fast. The safety net and safety blanket of large companies whose cost structures are huge and whose revenue streams are stalling may no longer provide the same kind of security as to prior to this inflection point. Companies like Cordys whose revenues are on a 100% growth curve can provide the needed jolt for questioning the viability of betting on the past alone versus a risk-adjusted model of investing with an eye to what new model will eventually become predominant in the new equilibrium.

### 6. **A Small Matter of Leadership**

Christensen's innovation dilemma is being put to the test at an even more radical pace by the hypercompetitive world we are about to be part of. Leaders who thrive in extreme competition are those who can intercept the changing models while avoiding to destroy the ones which currently run their companies. They understand what needs to be done, pick some important battles, put their best talent behind it and give them the resources required to drive the changes. 2009 will require a lot of leadership from companies' CEOs, CIOs, COOs, CFOs and their board of directors. The Cloud model is one which leaders can no longer afford not to experience.

# The IT Dilemma and the Unified Computing Framework

Edwin Paalvast

Vice President Service Sales Europe  
Cisco Systems  
Haarlerbergweg 13-19, 1101 CH Amsterdam  
The Netherlands

**Abstract.** One of the main challenges of the IT industry is the portion of the IT budget spent on maintaining the IT systems. For most companies this is around 80-85% of the total budget. This leaves very little room for new functionality and innovation. One of the ways to save money is to make more effective use of the underlying hardware like disks and processors. With the freed up budget the real IT issue can be addressed: the cost of application maintenance. With the use of publish-subscribe and agent models the changes in policies and business models can be supported more quickly, but it requires the right underlying infrastructure. I will discuss a Unified Computing framework that will enable these savings and will have the required capabilities to support model based programming.

# Tutorial: How to Value Software in a Business, and Where Might the Value Go?

Gio Wiederhold

Professor Emeritus  
Stanford University and MITRE Corporation  
Stanford CA 94305-9040  
USA

This tutorial consists of two parts. In the first part we motivate why businesses should determine the value of the Intellectual Property (IP) inherent in software and present a method to determine that value in a simple business model: Enterprise software for sale. In the second part we consider the same software as marketed in a service model. A service model imposes a maintenance obligation. Maintenance is often outsourced so we also consider the risk to IP, especially when work is performed offshore. Finally we present the consequences to the original creators when IP is segregated into a tax haven.

There exists a voluminous literature on estimation of the cost of producing software, but that literature largely ignores the benefits of using that software. While software creators believe that their products are valuable, they are rarely called upon to quantify its benefits. Evaluation of software and its benefits in commerce is left to lawyers, economists, software vendors, or promoters. The results are often inconsistent. For modern enterprises intellectual capital greatly exceeds the tangible capital on the books, but is not shown in their annual reports. The problem is that the value of software is essentially independent of the cost and effort spent to create it. A few brilliant lines of code can have a very high value, whereas a million lines of code that generate a report that nobody uses have little value. The presentation will present a method for valuing software based on the income that sales of a software product are expected to generate, following accepted economic principles. An issue specific to software is that software is continually being upgraded, and is hence more squishy than say books or music. The steps of the valuation process are integrated via a simple quantitative example.

Having a quantitative model on a spreadsheet allows exploration of alternatives. That capability is shown by evaluating also a service business model alternative. In that setting we can also determine why software has a finite life, although it does not wear out as tangibles do. The model is being used now for international IP transfer pricing. When companies outsource part of their operations they must also give access to the required IP. If that IP is not properly valued distortions occur in decision-making processes, and risks cannot be quantified. These distortions can have very large effects on the income and capital accumulation in the home and destination country. If a taxhaven is interposed as a holder of the IP the consequences for creators, the stockholders, and their countries can be amazing, and are rarely understood by the participants.

Awareness of the value of the product of a creator's knowledge and effort can help in making decisions on the design and the implementation focus. Some further conclusions are drawn from the modeling results that reflect on academic and business practice. A paper on the method used has appeared in the Comm. of the ACM, September 2006, but could not cover all of the issues. Links to further material are available at <http://infolab.stanford.edu/pub/gio/inprogress.html#worth>.

# Towards the Next Generation of Service-Based Systems: The S-Cube Research Framework\*

Andreas Metzger and Klaus Pohl

Software Systems Engineering, University of Duisburg-Essen  
45117 Essen, Germany  
{andreas.metzger,klaus.pohl}@sse.uni-due.de

**Abstract.** Research challenges for the next generation of service-based systems often cut across the functional layers of the Service-Oriented Architecture (SOA). Providing solutions to those challenges requires coordinated research efforts from various disciplines, including service engineering, service composition, software engineering, business process management, and service infrastructure. The FP7 Network of Excellence on Software Services and Systems (S-Cube) performs cross-discipline research to develop solutions for those challenges. Research in S-Cube is organised around the S-Cube research framework, which we briefly introduce in this paper. Moreover, we outline the envisioned types of interactions between the key elements of the S-Cube research framework, which facilitate the specification and design as well as the operation and adaptation of future service-based systems.

**Keywords:** Service-based Systems, Service Oriented Architecture, Service Engineering, Software Services.

## 1 Motivation

Service-orientation is increasingly adopted as a paradigm for building highly dynamic, distributed and adaptive software systems, called service-based systems. A service-based system is realized by composing software services. For the service composer, a software service is not an individual piece of software. Rather, it represents some functionality that can be invoked through the service's interface, where the actual software that implements this functionality is executed, maintained and owned by the provider of that service [1].

Currently, the common practice for developing service-based systems is to employ the Service-Oriented Architecture (SOA) paradigm, which distinguishes between three functional layers [2][3][4]: The *service infrastructure layer (SI)* supports describing, publishing and discovering services and provides the run-time environment for the execution of service-based systems. It provides primitives for service communication, facilities for service description, as well as capabilities for service discovery.

---

\* The research leading to these results has received funding from the European Community's Seventh Framework Programme FP7/2007-2013 under grant agreement 215483 (S-Cube). For further information please visit <http://www.s-cube-network.eu/>



The *service composition and coordination layer (SCC)* supports the (hierarchical) composition of multiple services. Such service compositions can in turn be offered to service clients, used in further service compositions and eventually be composed to service-based systems. The *business process management layer (BPM)* provides mechanisms for modelling, analysing and managing business processes that can span the administrative boundaries of multiple organizations.

When building service-based systems one typically faces challenges which cut across the functional SOA layers, such as:

- *Incomplete knowledge*: A service-based system cannot be specified completely in advance due to the incomplete knowledge about the interacting parties (e.g., service providers) as well as the system's context. Thus, compared to traditional software engineering, much more decisions need to be taken during the run-time (operation) phase. As a consequence, new life cycle models and methods will have to support the agility, flexibility and dynamism required to leverage the potential of service-based systems.
- *Adaptations across the functional layers*: Adaptations at the functional layers can be conflicting. For example, the service composition layer might perform a fragmentation of the business process while at the same time the infrastructure layer might initiate a data fragmentation. These two adaptations can be in conflict if, for example, the data required by a process fragment is moved to another location than the process fragment, or if the data fragmentation leads to a violation of privacy policies stipulated by the process fragments. Thus, an adaptation strategy is required which coordinates adaptations across the functional layers and thereby avoids such conflicts.
- *End-to-end-quality*: Each functional layer contributes to the end-to-end quality of a service-based system. Thus, to assure end-to-end quality, the different quality characteristics (like reliability or performance) and their dependencies must be understood and the different quality levels, as stipulated in individual quality contracts (e.g., as part of SLAs), need to be aggregated across the functional layers.

Addressing such cross-cutting challenges (see [5][6] for a more detailed list) requires an integration of knowledge and competencies of various disciplines, including service engineering, service composition and orchestration, software engineering, business process management, and service infrastructure. In Section 2 we sketch the S-Cube research framework, which provides a clear separation of concerns but also systematically addresses the cross-cutting challenges. In Section 3 we outline the envisioned types of interactions between the key elements of the S-Cube framework, which facilitate the specification and design as well as the operation and adaptation of future service-based systems.

## 2 The S-Cube Research Framework

Figure 1 provides an overview of the S-Cube research framework and its key elements. In addition to the functional SOA layers (BPM, SCC and SI – in the S-Cube framework we call those layers technology layers), the S-Cube research framework introduces the following cross-cutting elements [7]:

- *Service engineering and design (SED)*: SED provides principles, techniques, methods and life-cycle models which facilitate the creation of innovative service-based systems, which include requirements engineering and design principles and techniques. SED takes a holistic view on engineering, operating and evolving service-based systems.
- *Service adaptation and monitoring (SAM)*: SAM facilitates the cross-layer monitoring of service-based systems as well as their continuous adaptation in response to, e.g., context changes, system failures or underperformance. In addition, SAM supports the pro-active adaptation of service-based systems across the three layers.
- *Service quality definition, negotiation and assurance (SQDNA)*: SQDNA provides principles, techniques and methods for defining, negotiating and assuring end-to-end quality of service-based systems as well as conformance to SLAs. In addition, it facilitates pro-active adaptations by providing novel techniques for predicting the future quality of service-based systems.

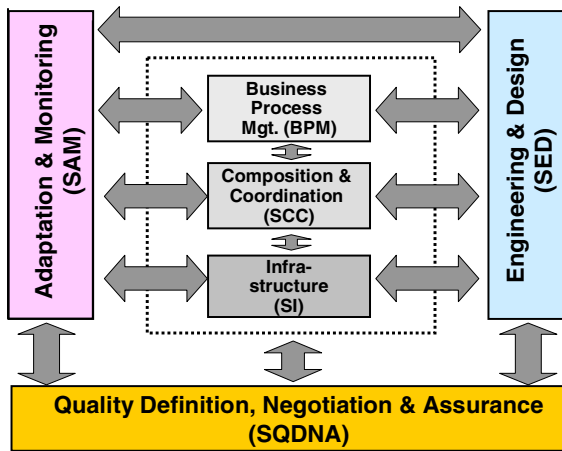


Fig. 1. Overview on the S-Cube research framework

### 3 Envisioned Interactions between the Framework Elements

For each element of the S-Cube research framework, its interactions and interfaces with the other framework elements are defined. We distinguish between two principle kinds of interactions:

- *Design and Specification Interactions*: These interactions expose the capabilities and features of one framework element to another framework element. The exposed capabilities and features are taken into account when engineering, designing, monitoring, adapting or assuring the quality of a service-based system. Moreover, the cross-cutting elements (SED, SAM and SQDNA) can constrain the capabilities offered by the three technology layers by specifying how to use those capabilities in a concrete service-based system.

- *Operation and Run-time Interactions:* These interactions reflect the information which is exchanged between the framework elements during the operation, execution and adaptation of the service-based system as well as its instances.

We illustrate the two types of interactions by describing the envisioned interactions between the SED, SAM and SCC elements depicted in Figure 2.

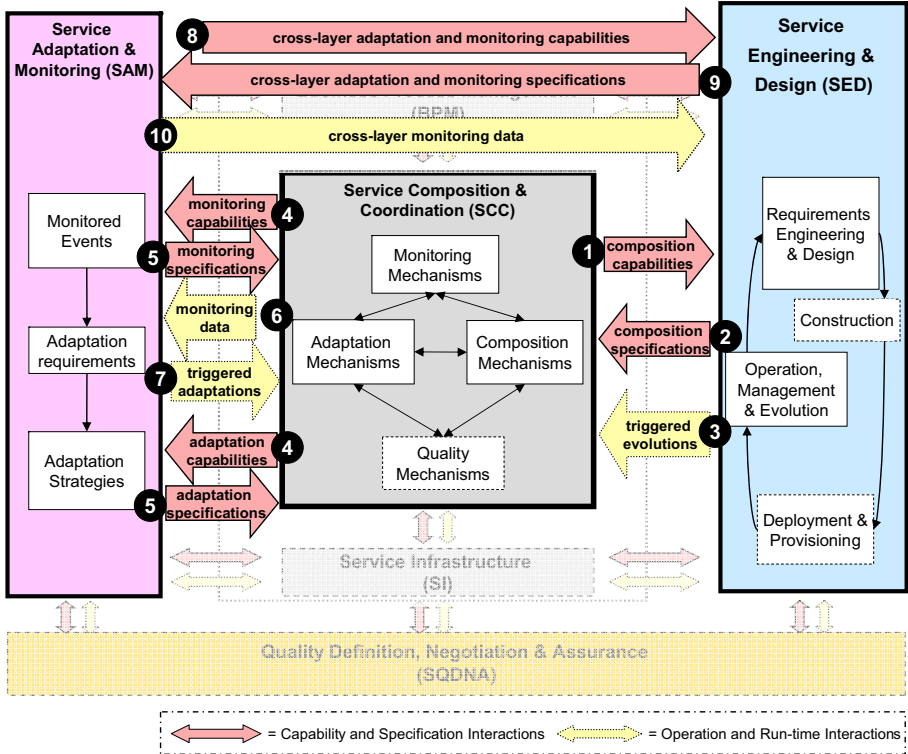


Fig. 2. Envisioned interactions between the SAM, SED and SCC elements

### Envisioned interactions between SCC and SED

As depicted in Figure 2 we envision three principle interactions between SCC and SED:

- (1) The SCC layer communicates its composition capabilities to the SED element. The SED element exploits those capabilities during requirements engineering and design of the service-based system together with the capabilities of the SI, BPM, SAM and SQDNA elements.
- (2) As a result of the engineering and design activities, the SED element specifies which SCC capabilities should be used for the service-based system at hand. The SED element communicates those specifications to the SCC layer. It thereby can restrict the capabilities offered by the SCC layer. For example, the SED element can forbid the use of a certain composition mechanism or define the order in

which the mechanisms have to be used. In addition, the SED element also communicates the specifications for the quality, the monitoring and the adaptation capabilities to the SCC layer (not depicted in Figure 2).

- (3) In the case of an evolution of the service-based system, the SED element communicates the evolution triggers (i.e., actions to be executed to implement the evolution) to the SCC layer. This includes updates of the specifications for the SCC capabilities.

### **Envisioned interactions between SAM and SCC**

Between SCC and SAM the following four key interactions are envisioned:

- (4) The SCC layer communicates its capabilities to collect monitoring data from the service compositions to the SAM element. In addition, the SCC layer exposes its capabilities for adapting service compositions to the SAM element. The SAM element uses those capabilities within its cross-layer adaptation techniques.
- (5) Based on the monitoring and adaptation strategies designed for the service-based system, the SAM element communicates the monitoring and adaptation specification for the service-based system at hand to the SCC layer and thereby defines which capabilities are valid for that service-based system.
- (6) During the operation of the service-based system, the collected monitoring data is communicated from the SCC layer to the SAM element. The SAM element analyzes the monitoring data received – under consideration of the monitoring data received from the SI and BPM layers – and determines required adaptations.
- (7) The adaptation strategies of the SAM element ensure conflict-free, cross-layer adaptations. If adaptations of the service compositions are required, the SAM element communicates the required adaptation specifications to the SCC layer.

### **Envisioned interactions between SAM and SED**

As shown in Figure 2, three principle interactions between SAM and SED are envisioned:

- (8) The SAM element communicates its cross-layer monitoring and adaptation capabilities to the SED element, which considers those capabilities during the engineering and design of the service-based system. For example, the SED element could decide whether to use intrusive monitoring or non-intrusive monitoring in a certain system and how the monitoring capabilities of all three technology layers should be correlated.
- (9) The SED element communicates the cross-layer adaptation and monitoring specification for the service-based system at hand to the SAM element. For example, the specification can define that only non-intrusive monitoring capabilities are to be used by all three technology layers. The SAM element follows the specification and adjusts his monitoring and adaptation capabilities for the service-based system accordingly.
- (10) During the operation of the service-based system, the SAM element correlates the monitoring data from the different layers and provides correlated, cross-layer monitoring data to the SED element. This data is analyzed to determine if an evolution of the service-based system is required (e.g., in the case that adaptations of individual instances of the service-based system do not suffice).

The small arrows in Figure 2 indicate interactions between framework elements which are not described above. The S-Cube research vision white paper describes all envisioned interactions of the S-Cube research framework. The white paper is available from the S-Cube web portal at <http://www.s-cube-network.eu/>.

## 4 Conclusions

The FP7 Network of Excellence S-Cube addresses the cross-cutting research challenges faced when engineering, designing, adapting, operating and evolving the next generation of service-based systems. S-Cube's research is guided by the research framework and its envisioned interactions sketched in this paper. The S-Cube research framework clearly distinguishes between principles and methods for engineering and adapting service-based systems and the technology and mechanisms which are used to realize those systems, while taking into account cross-cutting issues like Quality of Service (QoS) and SLA compliance. By synthesizing and integrating diversified knowledge across different research disciplines, S-Cube develops the next generation of principles, techniques and methods for the service-based systems of the future.

**Acknowledgments.** We cordially thank all S-Cube members for the fruitful discussions and their contributions to the S-Cube research framework.

## References

- [1] Di Nitto, E., Ghezzi, C., Metzger, A., Papazoglou, M., Pohl, K.: A Journey to Highly Dynamic, Self-adaptive Service-based Applications. *Automated Software Engineering* 15(3-4) (December 2008)
- [2] Erl, T.: *Service-oriented Architecture*. Prentice-Hall, Englewood Cliffs (2004)
- [3] Josuttis, N.: *SOA in Practice: The Art of Distributed System Design*. O'Reilly, Sebastopol (2007)
- [4] Kaye, D.: *Loosely Coupled: The Missing Pieces of Web Services*. RDS Press (2003)
- [5] Papazoglou, M., Pohl, K.: S-Cube: The Network of Excellence on Software Services and Systems. In: Di Nitto, E., Traverso, P., Sassen, A., Zwegers, A. (eds.) *At Your Service: An Overview of Results of Projects in the Field of Service Engineering of the IST Programme*. MIT Press Series on Information Systems (2009)
- [6] Papazoglou, M., Pohl, K.: Report on Longer Term Research Challenges in Software and Services. In: Boniface, M., Ceri, S., Hermenegildo, M., Inverardi, P., Leymann, F., Maiden, N., Metzger, A., Priol, T. (eds.) *Results from two workshops held at the European Commission premises at 8th of November 2007 and 28th and 29th of January 2008*, European Commission (2008), <http://www.cordis.lu>
- [7] Metzger, A., Pohl, K.: S-Cube: Enabling the Next Generation of Software Services. In: Filipe, J., Cordeiro, J., Cardoso, J. (eds.) *Proceedings of the 5th Intl. Conference on Web Information Systems and Technologies (WEBIST 2008)*. LNBIP, vol. 18, pp. 40–47. Springer, Heidelberg (2009)

# An Extensible Aspect-Oriented Modeling Environment

Naoyasu Ubayashi, Genya Otsubo, Kazuhide Noda, and Jun Yoshida

Kyushu Institute of Technology, Japan

{ubayashi,otsubo,noda,yoshida}@minnie.ai.kyutech.ac.jp

**Abstract.** AspectM is an aspect-oriented modeling language for providing not only basic modeling constructs but also an extension mechanism called metamodel access protocol (MMAP) that allows a modeler to modify the metamodel. This paper proposes a concrete implementation for constructing an aspect-oriented modeling environment in terms of extensibility. The notions of edit-time structural reflection and extensible model weaving are introduced.

## 1 Introduction

Aspect-oriented programming (AOP) [9] can separate crosscutting concerns from primary concerns. In major AOP languages such as AspectJ [10], crosscutting concerns including logging, error handling, and transactions are modularized as aspects and they are woven into primary concerns. AOP is based on join point mechanisms (JPM) consisting of join points, a means of identifying join points (pointcut), and a means of semantic effect at join points (advice). In AspectJ, program points such as method execution are detected as join points, and a pointcut designator extracts a set of join points related to a specific crosscutting concern from all join points. A weaver inserts advice code at the join points selected by pointcut designators. Aspect orientation has been proposed for coping with concerns not only at the programming stage but also at the early stages of the development such as requirements analysis and architecture design.

We previously proposed a UML-based aspect-oriented modeling (AOM) language called AspectM that provides not only major JPMs but also a mechanism called metamodel access protocol (MMAP) for allowing a modeler to modify the AspectM metamodel, an extension of the UML metamodel [15]. The mechanism enables a modeler to define a new JPM that includes domain-specific join points, pointcut designators, and advice. Although the notion of extensible AOM is useful, its construction has not yet been established.

This paper proposes a concrete implementation for constructing an AOM environment in terms of extensibility. In our AspectM support tool consisting of a model editor and a model weaver, the notions of edit-time structural reflection and extensible model weaving are introduced. The model editor supporting edit-time structural reflection enables a modeler to define a domain-specific JPM. A newly introduced JPM is dealt with by the extensible model weaver. Although

the AspectM language features can be extended by MMAP, it is not necessarily easy to confirm the correctness of model weaving. Verification concerning model consistency and aspect interference becomes difficult because a weaver must be validated whenever AspectM is extended. If a verification mechanism is not provided, it is difficult to judge whether an extended part does not interfere with other existing parts. If the extension includes defects, models are not woven properly. It is not easy for a modeler to know whether the original model or the extension is incorrect. To deal with this problem, the model weaver provides a set of verifiers consisting of a metamodel checker for verifying whether a base model conforms to the metamodel, a model structure checker for verifying well-formness, and an assertion checker for validating the intention of a modeler.

The remainder of the paper is structured as follows. Section 2 explains AspectM briefly, and claims why extension mechanisms are needed in AOM. The mechanisms of edit-time reflection and verifying model weaving are shown in Section 3 and 4, respectively. In Section 5, we show a case study that adopts AspectM and evaluates the effectiveness of the extension mechanism. Section 6 introduces related work. Concluding remarks are provided in Section 7.

## 2 Motivation

In this section, we briefly excerpt the overview of AspectM from our previous work [15]. AspectM without extension mechanisms is called *Core AspectM*. The necessity of extension mechanisms is pointed out.

### 2.1 Core AspectM

The notion of JPM can be applied to not only programming but also modeling as illustrated in Fig. 1: a class is regarded as a join point; a pointcut definition ‘`classA || classB`’ extracts the corresponding two classes; and a model transformation *add-attribute* is regarded as advice. Core AspectM provides seven basic JPMs as shown in Table 1: PA (pointcut & advice for operation bodies), CM (class composition), EL (element), OC (open class), RN (rename), RL (relation), and IH (inheritance). Although current Core AspectM only supports class diagrams, the dynamic aspect of system behavior can be described as a protocol state machine because a modeler can specify preconditions and postconditions in an operation by using OCL (Object Constraint Language).

An aspect in Core AspectM is separated into three compartments: aspect name, pointcut definitions, and advice definitions. An aspect name and a JPM type are described in the first compartment. Pointcut definitions are described in the second compartment. Each definition consists of a pointcut name, a join point type, and a pointcut body. In pointcut definitions, we can use designators including **cname** (class name matching), **aname** (attribute name matching), and **oname** (operation name matching). We can also use three logical operators: **&&** (and), **||** (or), and **!** (not). Advice definitions are described in the third compartment. Each of them consists of an advice name, a pointcut name, an advice

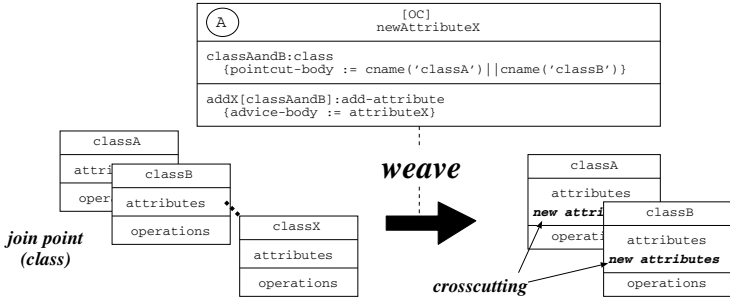


Fig. 1. Aspect-oriented modeling

Table 1. AspectM basic JPM

JPM	Join point	Advice
PA	operation	before, after, around add/delete/modify-precondition,postcondition
CM	class	merge-by-name
EL	class diagram	add/delete/modify-class
OC	class	add/delete/modify-operation, attribute, invariant
RN	class, attribute, operation	rename
RL	class	add/delete/modify-aggregation, relationship
IH	class	add/delete/modify-inheritance

type, and an advice body. A pointcut name is a pointer to a pointcut definition in the second compartment. The advice is applied at join points selected by a pointcut.

## 2.2 Problems in Core AspectM

Although Core AspectM provides basic JPMs, a modeler cannot define domain-specific JPMs. It would be better for a modeler to describe a model as shown in Fig. 2 (the class diagram is cited from [4]). Domain-specific model elements are denoted by stereotypes that are not merely annotations but elements introduced by metamodel extension.

The model in Fig. 2 describes an invoice processing system comprised of two kinds of domain-specific distributed components: `DCEntityContract` for defining the contract of a distributed entity component and `DCControllerContract` for defining the contract of a distributed controller component. The model also includes a domain-specific JPM `DCLogger` that adds log operations to `DCEntityContracts` whose `UniqueId` is not assigned by users. `DCLogger` consists of domain-specific pointcut designators and advice. `DCEntityContract` can be regarded as a domain-specific join point. The `!DCEntityContractUniqueIdIsUserAssigned` pointcut selects two classes `Customer` and `Invoice`. If only primitive predicates can be used, it is necessary to specify as follows: `cname`



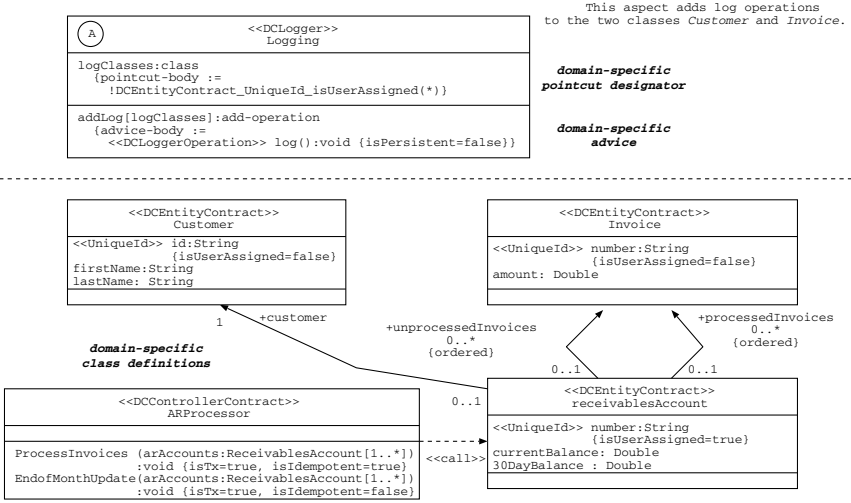


Fig. 2. Example of a domain-specific JPM

(‘Customer’) || cname(‘Invoice’). This definition must be modified whenever the `isUserAssigned` tag value is changed. This pointcut definition is fragile in terms of software evolution. On the other hand, an expressive pointcut such as `!DCEntityContract.UniqueId.isUserAssigned` is robust because this pointcut does not have to be modified even if the `isUserAssigned` tag value is changed.

Although expressive and domain-specific JPMs are effective, it is not necessarily easy to describe aspects such as *DCLogger* by merely using stereotypes as annotations because associations among stereotypes cannot be specified. Without extending a metamodel, the following fact cannot be specified: *DCEntityContract* must have a *UniqueId* whose tag is *isUserAssigned*. If an aspect is defined based on fragile stereotypes that lack consistency, the aspect might introduce unexpected faults because the aspect affects many model elements.

There are many situations that need domain-specific JPMs—for example, domain-specific logging, resource management, and transaction.

### 2.3 MMAP

There are two approaches to extending UML: a lightweight approach using stereotypes and a heavyweight approach that extends the UML metamodel by using MOF (Meta Object Facility). While it is easy to use stereotypes, there are limitations as mentioned above: the typing of tags is weak; and new associations among UML metamodel elements cannot be declared. On the other hand, MOF is very strong because all of the metamodel elements can be extended. MMAP aims at a middleweight approach that restricts available extension by MOF. Adopting this approach, domain-specific JPMs can be introduced at low cost.

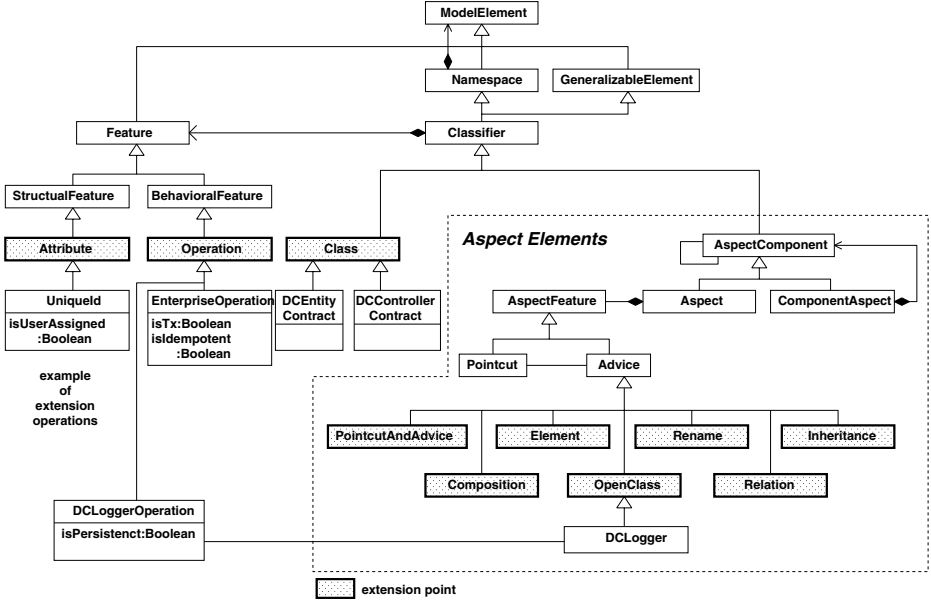


Fig. 3. AspectM metamodel

Table 2. MMAP primitive predicates

Predicate	Explanation
meta-class-of( $mc, c$ )	$mc$ is a metaclass of $c$
member-of( $m, c$ )	$m$ is a member of a class $c$
value-of( $v, a$ )	$v$ is value of an attribute $a$
super-class-of( $c1, c2$ )	$c1$ is a superclass of $c2$
related-to( $c1, c2$ )	$c1$ is related to $c2$

Fig. 3 shows a part of the AspectM metamodel defined as an extension of the UML metamodel. The aspect (`AspectComponent`) class inherits `Classifier`. Pointcuts and advice are represented by `Pointcut` and `Advice`, respectively. Concrete advice corresponding to the seven JPMs is defined as a subclass of `Advice`. The constraints among metamodel elements can be specified in OCL.

MMAP, a set of protocols exposed for a modeler to access the AspectM metamodel, is comprised of extension points, extension operations, and primitive predicates for navigating the AspectM metamodel. An extension point is an AspectM metamodel element that can be extended by inheritance. The extension points includes `Class`, `Attribute`, `Operation`, `Association` (`Association` is omitted in Fig. 3), and a set of JPM metaclasses. In Fig. 3, a class represented by a gray box is an extension point. An extension operation is a modeling activity allowed at the exposed extension points. There are four operations including *define subclasses*, *add attributes to subclasses*, *create associations among subclasses*, and *add/delete/replace constraints*. Table 2 is a list of primitive predicates for

navigating the metamodel. Using these predicates, pointcut designators can be defined as below. The defined pointcut designator represents all elements that satisfy the right-hand side predicates.

```
define pointcut cname(c):
  meta-class-of('Class', c) && member-of('Name', 'Class')
  && value-of(c, 'Name')
```

The idea of MMAP originates in the mechanisms of extensible programming languages, such as metaobject protocol (MOP) [8] and computational reflection in which interactions between the base-level (the level to execute applications) and the meta-level (the level to control meta information) are described in the same program. There are two kinds of reflection: behavioral reflection and structural reflection. MMAP corresponds to the latter. That is, MMAP focuses on the reflection whose target is a model structure.

## 2.4 Challenges in MMAP Implementation

We have to deal with the following challenges in order to implement MMAP effectively: 1) a model editor needs to be able to edit new model elements introduced by extending the metamodel and constrained by OCL; 2) a model weaver needs to be able to capture new model elements as join points and deal with new pointcuts defined by MMAP; and 3) the correctness of model weaving should be verified because it is difficult to check the consistency and the aspect interference due to the metamodel extension.

To solve these issues, we introduce the notions of edit-time structural reflection and verifying extensible model weaving. The contribution of this paper is to provide a method for constructing an AOM environment in terms of extensibility.

## 3 Reflective Model Editor

### 3.1 Concept

The reflective model editor allows a modeler to not only edit application models but also extend the metamodel. Fig. 4 is a screen shot that edits the invoice processing system (left side) in Fig. 2 and the AspectM metamodel (right side).

The concept of the edit-time structural reflection consists of two parts: the base editor and the metamodel editor. The former is the editor for base-level modeling, and the latter is the editor for modifying the AspectM metamodel and defining pointcut designators using MMAP primitive predicates. The metamodel editor exposes extension points. Only extension points are displayed on the editor screen as shown in Fig. 4. Other metamodel elements are not visible to a modeler, and not allowed to be modified. At an extension point, an extension operation such as *define subclasses* can be executed. This extension operation corresponds to *reification* in computational reflection. The result of extension operations enhances the functionality of the base editor. That is, new kinds of model elements can be used in the base editor. This corresponds to the *reflect*

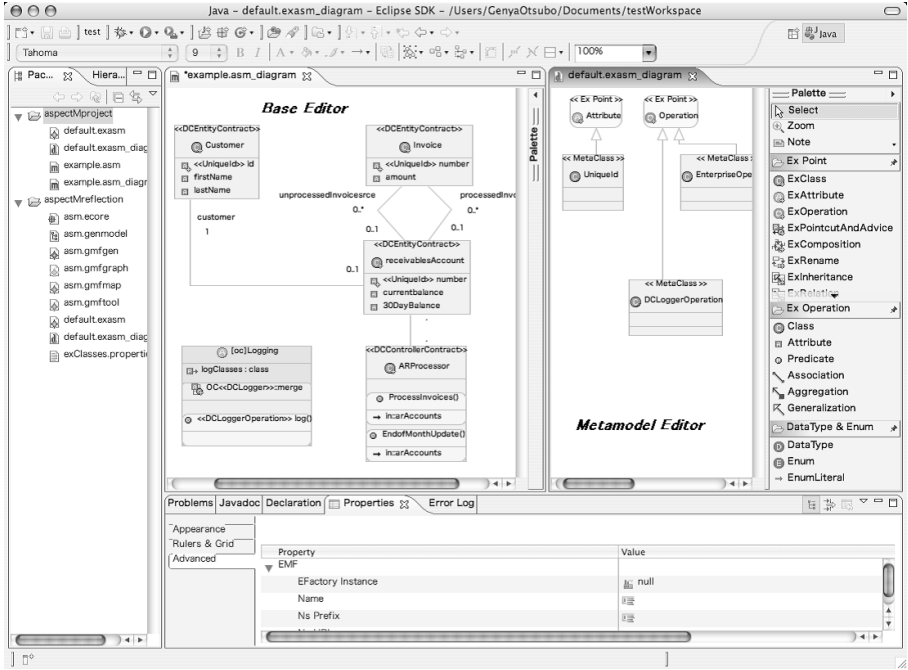


Fig. 4. Reflective model editor

concept in computational reflection. In reflective programming, a programmer can introduce new language features using MOP. In our approach, a modeler can introduce new model elements using MMAP.

### 3.2 Metamodel Extension Procedure

Using the example of the invoice processing system, we illustrate a procedure for extending the AspectM metamodel. As mentioned in section 2.2, the `Logging` aspect in Fig. 4 (left side) adds a log operation to the `DCEntityContracts` components whose `UniqueId` is not assigned by users. Although the bodies of the `logClasses` pointcut and the `addLog` advice whose type is `OC<<DCLogger>>` are invisible in Fig. 4, these bodies are defined in the same way as Fig. 2.

The following is the outline of extension steps: 1) execute extension operations; 2) assign a graphic notation to a new model element; 3) check the consistency between the previous metamodel and the new metamodel; 4) regenerate the AspectM metamodel; and 5) restart the base editor. In step 1, extension operations are executed at exposed extension points in order to introduce new domain-specific model elements. The constraints among new model elements can be specified using OCL. The model elements that violate the OCL descriptions can be detected by the editor. Pointcut designators are also defined as below.

```

define pointcut
DCEntityContract_UniqueId_isUserAssigned(c):
  meta-class-of('DCEntityContract', c) && member-of(a, c) &&
  meta-class-of('UniqueId', a) && member-of('isUserAssigned', 'UniqueId') &&
  value-of('true', 'isUserAssigned')

```

This pointcut selects all classes that match the following conditions: 1) the meta-class is `DCEntityContract`; 2) the value of the `isUserAssigned` is true. In case of Fig. 2, the negation of this pointcut designator selects the two classes `Customer` and `Invoice`. After steps 2 – 5, the new model element can be used in the base editor. In the reflective model editor, an extension model is separated from the original AspectM metamodel. Extension models can be accumulated as components for domain-specific modeling.

### 3.3 Implementation

The reflective model editor, a plug-in module for Eclipse, is developed using the Eclipse Modeling Framework (EMF) [3] and Graphical Modeling Framework (GMF) [5]. The former is a tool that generates a model editor from a metamodel, and the latter provides a generative component and runtime infrastructure for developing a graphical editor based on EMF. EMF consists of *core EMF*, *EMF.Edit*, and *EMF.Codegen*: the *core EMF* provides a meta model (Ecore) for describing models and runtime support; *EMF.Edit* provides reusable classes for building editors; and *EMF.Codegen* generate code needed to build a complete editor for an EMF model. Since an editor generated from EMF does not provide graphical facilities, GMF is used for this purpose.

The reflective mechanism is implemented as follows: 1) the original AspectM metamodel is defined as an EMF model, and the original base editor is generated using *EMF.Codegen*; 2) the metamodel extension specified by a modeler is saved as an EMF model, and the editor code for the extension is generated using *EMF.Codegen*; and 3) a new plug-in is generated from the code for the base editor and the extension, and replaced with the original plug-in.

## 4 Verifying Model Weaver

In this section, we show a method for constructing an extensible model weaver with a set of verifiers.

### 4.1 Model Weaving

The model weaver, which compounds base models, is implemented using DOM (Document Object Model) and Prolog. After the weaving, a model is translated into Java.

First, the weaver transforms the base and meta models into a set of Prolog facts. For example, the `Invoice` class and related metamodel elements are represented as follows.

```

-- from Invoice class
  meta-class-of('DCEntityContract', 'Invoice'), member-of('number', 'Invoice'),
  meta-class-of('UniqueId', 'number'), value-of('true', 'isUserAssigned').
-- from AspectM metamodel
  member-of('isUserAssigned', 'UniqueId').

```

Second, the model compiler converts a pointcut into a Prolog query, and checks whether the query satisfies the facts above. For example, the negation of the `DCEntityContract_UniqueId_isUserAssigned` pointcut selects `Customer` and `Invoice` as join points. The model weaver executes advice at these join points.

In the current MMAP, the `Advice` class is not exposed as an extension point because this extension needs a new weaver module that can handle new advice. Adopting our approach, the model weaver need not be modified even if the metamodel is modified by the reflective model editor. As shown here, the model weaver can deal with domain-specific join points and pointcuts introduced by using MMAP. That is, our model weaver is extensible.

## 4.2 Model Verification

In the model verification, we focus on the followings: 1) every model should conform to an extended metamodel and be well-formed; and 2) the result of weaving should reflect the intention of a modeler. The model verifier consists of a metamodel checker, a model structure checker, and an assertion checker.

**Metamodel checker.** There are two problems concerning metamodel extension. First, a base model might not conform to the modified metamodel even if the base model conforms to the previous metamodel. A base model that includes a class instantiated by a metaclass introduced by a metamodel (version 1) does not conform to a new metamodel (version 2) if the metaclass is deleted in the version 2. We should take into account not only base model evolution but also metamodel evolution. This issue is essential in continuous modeling-level reflection. Second, a woven model might not conform to the metamodel even if each base model before the weaving conforms to the metamodel. Since a new kind of model transformation can be introduced by adding user-defined aspects, a model transformed by these aspects might not conform to the metamodel if the aspects are not adequate.

**Model structure checker.** A woven model might not be well-formed due to the interference among aspects even if the model conforms to the metamodel. The model might include name conflicts, multiple inheritance, and cyclic inheritance.

**Assertion checker.** Although the mechanism of user definable pointcuts is effective, it is not easy for a modeler to check whether an introduced pointcut captures join points correctly. Although the precedence can be specified in `AspectM`, the intended results might not be obtained when the modeler makes a mistake. The mixture of illegal pointcuts and aspect precedence might cause unexpected weaving. The assertion checker verifies the intention of a modeler

to deal with these problems. The intention of the modeler can be specified as assertions described in MMAP primitive predicates.

The verification procedure is as follows: 1) translates base and meta models into Prolog facts; 2) generates Prolog queries from assertions; and 3) checks the satisfiability of the Prolog queries. Since an AspectM model is stored as an XML document, step 1 can be implemented as a translator from XML to Prolog. The following is an example of an XML model and generated Prolog facts. This model represents an operation `TransOp` whose type is `TransactionOperation`, a subclass of the `Operation` metaclass.

```
-- A model represented in XML
<ownedElement name="TransOp"
  xsi:type="asm:TransactionOperation" />
-- Generated Prolog facts
modelElement(
  [property('tagName', 'ownedElement'), property('name', 'TransOp'),
   property('xsi:type', 'asm:TransactionOperation')])
```

When a modeler wants to check the effect of an aspect for adding `TransOp` to the C class, he or she has only to specify the assertion *operation-of('TransOp', 'C')*.

## 5 Case Study and Evaluation

In this section, we show a case study using the AspectM support tool and discuss the effectiveness of the extension mechanism provided by MMAP. As a case study, we show a UML-based domain-specific language (DSL) for describing the external contexts of embedded systems.

### 5.1 DSL Construction

Currently, development of embedded systems is mainly conducted from the viewpoint of system functionalities: how hardware and software components are configured to construct a system—contexts are not considered explicitly in most cases. As a result, unexpected behavior might emerge in a system if a developer does not recognize any possible external contexts. It is important to analyze external contexts in order to detect the unfavorable behavior systematically at the early stage of the development.

To deal with these problems, we are developing a context-dependent requirements analysis method called CAMEmb (Context Analysis Method for Embedded systems) in which a context model is constructed from system requirements by using a DSL based on *UML Profile for Context Analysis* proposed by us.

Fig. 5 illustrates the result of the context analysis for a LEGO line trace car and its external contexts. The car runs tracing a line by observing a line color.

Table 3 shows a list of stereotypes introduced by our profile that can describe system elements, context elements, and associations between them: three kinds of stereotypes  $\ll \textit{Context} \gg$ ,  $\ll \textit{Sensor} \gg$ , and  $\ll \textit{Actuator} \gg$  are defined as an extension of the UML class; and five kinds of stereotypes  $\ll \textit{Observe} \gg$ ,

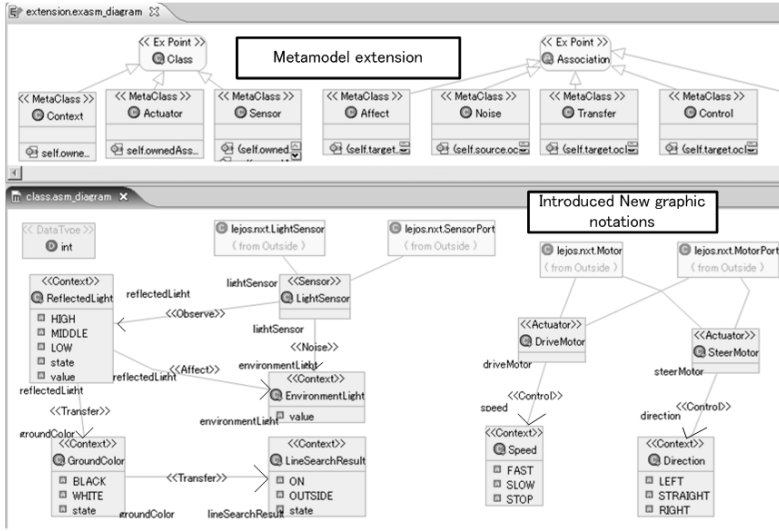


Fig. 5. Model editor for CAMEmb

Table 3. UML Profile for Context Analysis

Model element	Extension point	Definition
$\ll \textit{Context} \gg$	Class	Context
$\ll \textit{Sensor} \gg$	Class	Sensor
$\ll \textit{Actuator} \gg$	Class	Actuator
$\ll \textit{Observe} \gg$	Association	Sensor observes a context
$\ll \textit{Control} \gg$	Association	Actuator controls a context
$\ll \textit{Transfer} \gg$	Association	Data is transformed into a different form because a sensor cannot observe the original data directly
$\ll \textit{Noise} \gg$	Association	Noise from other contexts
$\ll \textit{Affect} \gg$	Association	Affected by other contexts

$\ll \textit{Control} \gg$ ,  $\ll \textit{Transfer} \gg$ ,  $\ll \textit{Noise} \gg$ , and  $\ll \textit{Affect} \gg$  are defined as an extension of the UML association. These new model elements are introduced by using MMAP as shown in the top of Fig 5. As shown in Fig 5, the line trace car must observe the light reflected from the ground because the car cannot directly observe the ground line. This is represented by  $\ll \textit{Transfer} \gg$ . Unfavorable behavior emerges when strong environment light is thrown to a line—the car cannot observe a line color correctly. The `EnvironmentLight` context and  $\ll \textit{Affect} \gg$  represent this unfavorable phenomenon. In AspectM, constraints among metamodel elements are specified in OCLs as below. This OCL indicates that  $\ll \textit{Actuator} \gg$  should have  $\ll \textit{Control} \gg$ , and can be verified by our extended model editor.

```
context  $\ll \textit{Actuator} \gg$ 
self.ownedAssociation -> select(oclIsTypeOf(Control))
```



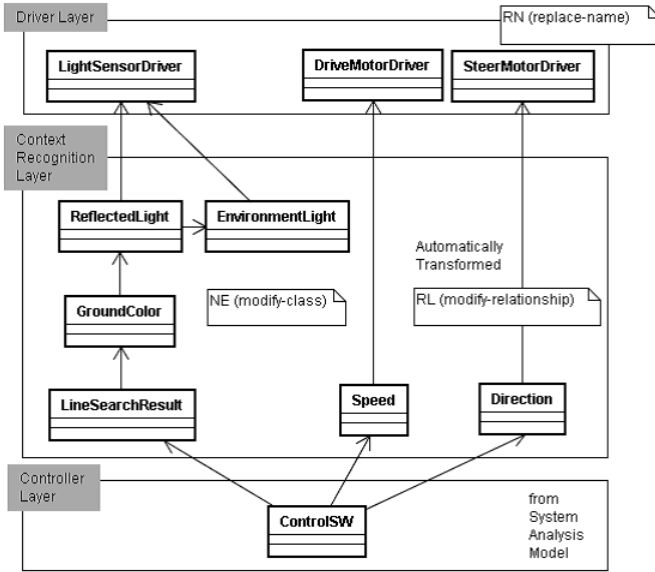


Fig. 6. Generated software design model

## 5.2 Model Weaver Construction

A model weaver supporting CAMEmb transforms context analysis models into a software design model that takes into account the contexts as shown in Fig. 6. The *Driver* and *Context Recognition* layers are automatically generated by the weaver. Although the *Controller* layer is created by hand currently, this can be transformed from a system analysis model. A set of contexts in the context analysis models are transformed into internal state classes recognized by a software controller.  $\ll Sensor \gg$  and  $\ll Actuator \gg$  are transformed into driver classes that operate hardware components. The software controller interacts with expected contexts by referring and changing the values of the internal state classes that communicate with driver classes.

A model weaver specific to CAMEmb can be constructed by defining a set of aspects. For example, the pointcut that captures sensors/actuators (*LightSensor*, *DriveMotor*, and *SteerMotor*) as join points is defined as follows: *meta-class-of('Actuator',c) || meta-class-of('Sensor',c)*.

Using model verifiers, we can check the followings: 1) the context analysis model conforms to the metamodel extended for supporting CAMEmb; and 2) the generated software design model is well-formed. The intention of a modeler can be also checked. For example, if the modeler wants to check whether the software controller can access to driver classes, he or she only has to specify an assertion such as *related-to('ControlSW',d) && meta-class-of('Driver',d)*.

**Table 4.** Tasks for constructing DSL tool

Task	Number
1.Extend the metamodel	8
2.Define meta-level OCLs	8
3.Define aspects commonly reused in CAMEmb	3
4.Define aspects specific to a line trace car	11
5.Reuse LEGO OS components such as lejos.nxt.LightSensor	4

### 5.3 Evaluation

Table 4 shows the number of tasks for defining DSL and constructing the CAMEmb weaver. Three aspects are not specific to the line trace car development. Although other eleven aspects are specific to the line trace car, they can be reused repeatedly if software product line (SPL) approach is adopted. In SPL, a product is constructed by assembling core assets, components reused in a family of products (line trace car family in this case).

The code size of the line tracing function was 223 LOC (lines of code), and 174 LOC was automatically generated from the context analysis model by using the CAMEmb model weaver. The percentage of automated generated code was 78 %. Since the classes in the context recognition layer are value object classes, these classes only need setters, getters, and calls to other connected context classes. It is easy to generate these programs. Since the driver classes only call the LEGO OS components, it is also easy to generate code. However, as mentioned in section 5.2, only the software controller program was coded by hand. This can be generated from system analysis model if our weaver supports state machine diagrams. We plan to develop a model weaver for state machine diagrams.

The CAMEmb tool for supporting MDA whose platform was LEGO OS could be constructed by only using MMAP. No extra programming was needed. We did not have to modify the metaclasses that were not exposed by MMAP. From our experience, MMAP was sufficient to extend the AspectM metamodel. We believe that the MOP approach in modeling-level is more effective than the full metamodel extension approaches in terms of the cost and usability.

## 6 Related Work

There has been research that has attempted to apply aspect orientation to the modeling phase. Stein, D. et al. proposed a method for describing aspects as UML diagrams [14]. In this work, an aspect at the modeling-level was translated into the corresponding aspect at the programming language level, for example as an aspect in AspectJ. An aspect in AspectM is not mapped to an element of a specific programming language, but operates on UML diagrams. In Motorola WEAVR [1], weaving for UML statecharts including action semantics is possible.

Domain-specific aspect-oriented extensions are important. Early AOP research aimed at developing programming methodologies in which a system was composed of a set of aspects described by domain-specific AOP languages [9]. Domain-specific extensions are necessary not only at the programming stage but also at

the modeling stage. Gray, J. proposed a technique of aspect-oriented domain modeling (AODM) [6] that adopted the Generic Modeling Environment (GME), a meta-configurable modeling framework. The GME provides meta-modeling capabilities that can be adapted from meta-level specifications for describing domains. The GME approach is heavyweight because meta-level specifications can be described fully. On the other hand, our approach is middleweight. Although all of the AspectM metamodel cannot be extended, domain-specific model elements can be introduced at relatively low cost.

MMAP is similar to an edit-time metaobject protocol (ETMOP) [2] that runs as part of a code editor and enables metadata annotations to customize the rendering and editing of code. An ETMOP programmer can define special metaclasses that customize a display and editing. Although ETMOP's research goal that is to provide mechanisms for making programs more visually expressive is similar to our goal, we focus on the provision of middleweight mechanisms for domain-specific expressiveness. We think that the concept of MMAP can be applied to not only class diagrams but also other UML diagrams such as use cases and state machine diagrams because these diagrams have corresponding metamodels and they can be extended.

Gybels, K. and Brichau, J. proposed pattern-based pointcut constructs using logic programming facilities [7]. Ostermann, K. et al. also proposed a pointcut mechanism based on logic queries written in Prolog [12]. The mechanism is implemented for a typed AOP language, called ALPHA. Although our pointcut definition method using Prolog is similar to their approaches, the target of our approach is not programming but modeling.

In UML 2.x, profiling mechanisms that have been improved from UML 1.x include part of our ideas because new kinds of stereotypes can be introduced by extending UML metamodel and OCL can be specified. However, most of the current UML 2.x tools still provide only the stereotype definition support and do not take into account the extensibility from the viewpoint of model compilation. The main contribution of our approach is to provide an extensive mechanism for AOM. This includes extension of model elements, join points, pointcuts, and verification. We also provide a concrete implementation strategy for realizing an integrated extensible AOM environment. Of course, our tool can be used as a substitute of the UML profiling facility. If we take a stand on UML profiling, we can consider our approach as a promising mechanism for integrating AOM with UML profiles.

## 7 Conclusion

In this paper, we demonstrated the effectiveness of an extensible AOM environment consisting of a model editor and a verifying model weaver. In order to support MMAP, we introduced the notions of edit-time structural reflection, extensible model weaving, and verification mechanisms. In our current implementation, we do not use OCL but Prolog because it is relatively easy to implement the reasoning and verification mechanisms in our prototype tool. Currently, we are developing a new version of AspectM in which OCL is adopted. Moreover,

we plan to develop a compilation method for translating aspects into a standard model transformation language called QVT (Queries, Views, and Transformations) [13] in order to enhance the power of model transformation. Although our current tool needs refining, it is an important step towards extensible AOM environments integrated with UML.

## References

1. Cottenier, T., Berg, A., Elrad, T.: The Motorola WEAVR: Model Weaving in a Large Industrial Context. In: International Conference on Aspect-Oriented Software Development (AOSD 2007), Industry Track (2007)
2. Eisenberg, A.D., Kiczales, G.: Expressive Programs through Presentation Extension. In: Proceedings of International Conference on Aspect-Oriented Software Development (AOSD 2007), pp. 73–84 (2007)
3. EMF, <http://www.eclipse.org/emf/>
4. Frankel, D.S.: Model Driven Architecture. John Wiley & Sons, Inc., Chichester (2003)
5. GMF, <http://www.eclipse.org/gmf/>
6. Gray, J., Bapty, T., Neema, S., Schmidt, D., Gokhale, A., Natarajan, B.: An Approach for Supporting Aspect-Oriented Domain Modeling. In: Pfenning, F., Smaragdakis, Y. (eds.) GPCE 2003. LNCS, vol. 2830, pp. 151–168. Springer, Heidelberg (2003)
7. Gybels, K., Brichau, J.: Arranging Language Features for More Robust Pattern-based Crosscuts. In: Proceedings of the 2nd International Conference on Aspect-Oriented Software Development (AOSD 2003), pp. 60–69 (2003)
8. Kiczales, G., des Rivieres, J., Bobrow, D.G.: The Art of the Metaobject Protocol. MIT Press, Cambridge (1991)
9. Kiczales, G., Lamping, J., Mendhekar, A., Maeda, C., Lopes, C., Loingtier, J., Irwin, J.: Aspect-Oriented Programming. In: Aksit, M., Matsuoka, S. (eds.) ECOOP 1997. LNCS, vol. 1241, pp. 220–242. Springer, Heidelberg (1997)
10. Kiczales, G., Hilsdale, E., Hugunin, J., et al.: An Overview of AspectJ. In: Knudsen, J.L. (ed.) ECOOP 2001. LNCS, vol. 2072, pp. 327–353. Springer, Heidelberg (2001)
11. Liskov, B.H., Wing, J.M.: A Behavioral Notion of Subtyping. ACM Transactions on Programming Languages and Systems (TOPLAS) 16(6), 1811–1841 (1994)
12. Ostermann, K., Mezini, M., Bockisch, C.: Expressive Pointcuts for Increased Modularity. In: Black, A.P. (ed.) ECOOP 2005. LNCS, vol. 3586, pp. 214–240. Springer, Heidelberg (2005)
13. QVT, <http://qvtp.org/>
14. Stein, D., Hanenberg, S., Unland, R.: A UML-based aspect-oriented design notation for AspectJ. In: Proceedings of International Conference on Aspect-Oriented Software Development (AOSD 2002), pp. 106–112 (2002)
15. Ubayashi, N., Tamai, T., Sano, S., Maeno, Y., Murakami, S.: Metamodel Access Protocols for Extensible Aspect-Oriented Modeling. In: Proceedings of the 18th International Conference on Software Engineering and Knowledge Engineering (SEKE 2006), pp. 4–10 (2006)

# Incremental Detection of Model Inconsistencies Based on Model Operations

Xavier Blanc<sup>1</sup>, Alix Mougenot<sup>2,\*</sup>, Isabelle Mounier<sup>2</sup>, and Tom Mens<sup>3,\*\*</sup>

<sup>1</sup> INRIA Lille-Nord Europe, LIFL CNRS UMR 8022,  
Université des Sciences et Technologies de Lille, France

<sup>2</sup> MoVe - LIP6, Université Pierre et Marie Curie, France

<sup>3</sup> Service de Génie Logiciel, Université de Mons-Hainaut, Belgium

**Abstract.** Due to the increasing use of models, and the inevitable model inconsistencies that arise during model-based software development and evolution, model inconsistency detection is gaining more and more attention. Inconsistency checkers typically analyze entire models to detect undesired structures as defined by inconsistency rules. The larger the models become, the more time the inconsistency detection process takes. Taking into account model evolution, one can significantly reduce this time by providing an incremental checker. In this article we propose an incremental inconsistency checker based on the idea of representing models as sequences of primitive construction operations. The impact of these operations on the inconsistency rules can be computed to analyze and reduce the number of rules that need to be re-checked during a model increment.

## 1 Introduction

Model driven development uses more and more complementary models. Indeed, large-scale industrial software systems are currently developed by hundreds of developers working on hundreds of models of different types (e.g. SysML, UML, Petri nets, architecture, work flow, business process) [1]. In such a context, model inconsistency detection is gaining a lot of attention as the overlap between all these models (that are often maintained by different persons) is a frequent source of inconsistencies.

Detection of inconsistencies was first introduced by Finkelstein et al. [2]. They defined the Viewpoints Framework, where each developer owns a viewpoint composed only of models that are relevant to him. The framework offers facilities to ensure consistency between viewpoints. The main insight is that model consistency cannot and should not be preserved at all times between all viewpoints [3]. The Viewpoints Framework suggests to allow for temporary model inconsistencies rather than to enforce model consistency at all times.

---

\* This work was partly funded by the french DGA.

\*\* This work was partly funded by Action de Recherche Concertée AUWB-08/12-UMH19, *Ministère de la Communauté française, Direction générale de l'Enseignement non obligatoire et de la Recherche scientifique, Belgique.*

In all approaches that deal with detection of inconsistencies [4,5,6,7,8,9], the detection invariably consists in analysing models to detect inconsistent configurations defined by inconsistency rules. Therefore, the larger the models, the longer the detection process takes. Moreover, the large number of inconsistency rules and their complexity are two other factors that make the detection process highly time consuming. The impact of model changes should also be considered by consistency checkers. Indeed, developers keep modifying and improving their models, and some of these modifications may give rise to new model inconsistencies. Due to the time it takes, re-checking the entire model after each such model increment is unfeasible in practice.

This situation explains why there is an increasing focus on scalability issues [6,9,10]. The challenge is to check inconsistencies on large models continuously during their frequent evolution. As the detection of inconsistencies implies to find structures within a model, efforts mainly target the process of the incremental detection in its whole (what rules to check and when) and aim at not performing a complete check of the model each time it evolves.

In this article, we propose to address this challenge by providing an incremental inconsistency checker that only adds a small fixed amount of memory to run on top of our classical inconsistency checker. Given a model that has already been checked for inconsistency, and given a model increment (i.e., a sequence of modifications to this model), our goal is to identify those inconsistency rules that need to be re-checked. Section 2 explains how to detect inconsistencies and gives a formal definition of an incremental checker. Our proposal is based on the operation-based model construction approach presented in [9], which is briefly revisited in section 3. Section 4 presents our incremental checker, and section 5 provides a case study to validate our approach. Section 6 presents related work in this domain and we conclude in Section 7.

## 2 Detection of Inconsistencies

### 2.1 Inconsistency Rules

Detection of inconsistencies consists in analyzing models to identify unwanted configurations defined by the inconsistency rules. If such configurations are found in the model, then the model is said to be inconsistent. Inconsistency rules can be compared to the negation of well-formedness rules of [11], structural rules of [12] and syntactic rules of [13].

One can see an inconsistency check as a function that receives as input a model and a set of inconsistency detection rules and that returns the evaluation result of each rule. If a rule evaluates to true (i.e., the model is inconsistent), then the model elements causing the inconsistency are also returned by the check function.

In this paper we use two inconsistency examples that are inspired by the class diagram part of the UML 2.1 specification [14]. Figure 1 presents a simplified fragment of the UML 2.1 meta-model for classes [14]. It will be referred to as CMM for Class Meta Model in the remainder of this article.

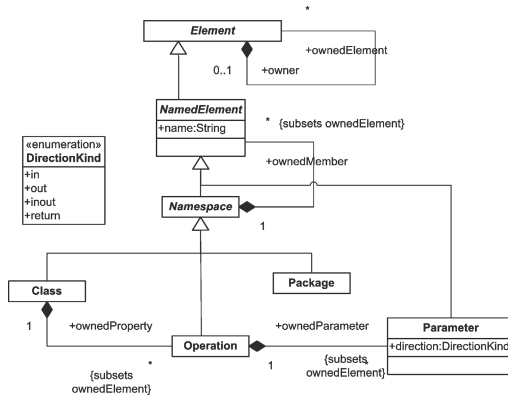


Fig. 1. CMM: A simplified fragment of the UML 2.1 meta-model

The two inconsistency rules we use are specified in the UML 2.1 specification: **OwnedElement** defines that an element may not directly or indirectly own itself; **OwnedParameter** defines that an operation can have at most one parameter whose direction is ‘return’.

Figure 2 shows a model instance of the CMM meta-model. This model is used as a running example to illustrate our approach. It is composed of a package (named ‘Azureus’) that owns two classes (named ‘Client’ and ‘Server’). The class ‘Server’ owns an operation (named ‘send’) that does not own any parameter. The model is consistent w.r.t. our two inconsistency rules.

## 2.2 Incremental Checking

During any model-driven software project, models are continuously modified by developers. As each modification can impact many model elements, checks should be performed as often as possible during the development life cycle in order to have a good control over the model consistency. However, as the time needed to perform a check can be very high, the challenge is to control efficiently the consistency of the model without burdening or delaying the developer in his other modeling activities.

One way of dealing with this problem is to provide an incremental checker. Incremental checks take into account the modifications made to a model. Rather than re-checking the entire model, one can analyze the impact of a set of modifications (the model increment  $\delta$ ) on the consistency of a model, and only re-check those inconsistency rules whose value may potentially have changed. In this way, the number of rules to be checked after modification may be reduced significantly, leading to an increased performance of the algorithm when compared to checking the inconsistency of the entire model.

Ideally, for an incremental check to be efficient, two considerations should be made. First, an inconsistency rule should be incrementally re-checked only if the modifications contained in the model increment change its previous evaluation.

Second, after a modification, only the part of the model that is concerned by the modifications needs to be analyzed in order to perform the new evaluation.

Following those two considerations while building an incremental checker, the incremental check should (1) evaluate only a subset of inconsistency rules and (2) analyze only a subset of model elements. Currently our approach only targets the first point and aims at filtering at a low level of granularity those rules that need to be re-checked after some model increment. It should be noted that our approach only needs a small fixed memory size to run on top of our classical inconsistency checker.

### 3 Detection of Inconsistencies Based on Model Construction

#### 3.1 Operation-Based Model Construction

In [9], we propose to represent models as sequences of elementary operations needed to construct each model element. The four elementary operations we defined are inspired by the MOF reflective API [15]:

1. **create**( $me, mc$ ) creates a model element  $me$  instance of the meta-class  $mc$ . A model element can be created if and only if it does not already exist in the model;
2. **delete**( $me$ ) deletes a model element  $me$ . A model element can be deleted if and only if it exists in the model and it is not referenced by any other model element;
3. **setProperty**( $me, p, Values$ ) assigns a set of  $Values$  to the property  $p$  of the model element  $me$ ;
4. **setReference**( $me, r, References$ ) assigns a set  $References$  to the reference  $r$  of the model element  $me$ .

Figure 3 is the construction sequence  $\sigma_c$  used to produce the model of Figure 2. In Figure 3, line 1 corresponds to the creation of the package; line

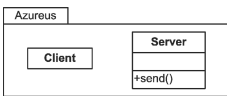


Fig. 2. Azureus UML model

```

1 create(p1,Package)
2 setProperty(p1,name, {'Azureus'})
3 create(c1,Class)
4 setProperty(c1,name, {'Client'})
5 create(c2,Class)
6 setProperty(c2,name,{'Server'})
7 setReference(p1,ownedMember,{c1,c2})
8 setReference(p1,ownedElement,{c1,c2})
9 create(o1,Operation)
10 setProperty(o1,name, {'send'})
11 setReference(c2, ownedProperty, {o1})
12 setReference(c2, ownedElement, {o1})
  
```

Fig. 3. Model construction operation sequence  $\sigma_c$



2 corresponds to the assignment of the name of the package; lines 3 and 5 correspond to the creation of the two classes; lines 4 and 6 to the assignment of the name of the two classes; line 7 links the two classes to the the package's owned members; line 8 does the same but with the owned element list of the package (the parameter list subsets the element list); lines 9 and 10 correspond to the creation of the operation and its name affectation; line 11 links the operation to the class' properties; line 12 does the same but with the element list of the class (the property list subsets the element list). This arbitrary sequence is used in the next sections to illustrate our incremental inconsistency checker.

### 3.2 Inconsistency Detection Rules

Our formalism allows to define any inconsistency rule as a logic formula over the sequence of model construction operations. As syntactic shortcut, we define the 'last' prefix to denote operations that are not followed by other operations canceling their effects. For instance, a `lastCreate(me, Class)` operation is defined as a `create(me, Class)` operation that is not followed by a `delete(me)` operation; and a `lastSetReference(me, ownedProperty, val)` operation is defined as a `setReference(me, ownedProperty, val)` operation for which the value of the ownedProperty reference of `me` in the model corresponds to `val`. A complete description of the semantics of the 'last' operations is provided in [9].

For the `OwnedProperty` inconsistency rule, the operations that can make a model inconsistent are the ones that modify a reference to the ownedParameter list of an operation and the ones that modify the direction of a parameter. More formally, those operations are `setReference(me, ownedParameter,  $\theta$ )` where  $\theta \neq \emptyset$  and `setProperty(me, direction, val)`.

This inconsistency rule can be formalised as follows:

$$\begin{aligned} \text{OwnedParameter}(\sigma) = \{ & me \mid \exists sr, sp1, sp2 \in \sigma \cdot \\ & sr = \text{setReference}(me, \text{ownedParameter}, \theta) \wedge \\ & sp1 = \text{setProperty}(p1, \text{direction}, 'return') \wedge \\ & sp2 = \text{setProperty}(p2, \text{direction}, 'return') \wedge \\ & p1, p2 \in \theta \wedge p1 \neq p2 \} \end{aligned}$$

Sequence  $\sigma_c$  of Figure 3 produces a model that is consistent with rule `OwnedParameter`, as it contains only one operation (line 9) that has no parameter.

For the `OwnedElement` inconsistency rule we presented in section 2.1, the only operation that can make a model inconsistent is the one that adds a reference to the ownedElement list of a model element. More formally, this operation is `setReference(me, ownedElement,  $\theta$ )` where `me` is an element and  $\theta$  is not empty. Such an operation produces an inconsistent UML model if and only if the set  $\theta$  is such that a cycle appears among the `ownedElement` references. The way to repair such an inconsistent model is to break the cycle by removing a relevant reference. One can easily check that sequence  $\sigma_c$  produces a model that is consistent with rule `OwnedElement`.

## 4 Incremental Checking Based on Model Operations

Our incremental inconsistency checker reduces the set of inconsistency rules that need to be re-checked. Our approach is based on analyzing the impact that operations of the model increment may have on the evaluation of the inconsistency rules. We define a partition of equivalence classes for construction operations and use this partition to classify the inconsistency rules. Section 4.1 presents the equivalence classes and section 4.2 explains how those classes can be used to classify the inconsistency rules. Section 4.3 then presents an example of this mechanism and highlights its benefits for building an incremental checker.

### 4.1 Partitioning of Operations

To reduce the set of inconsistency rules to re-check we rely on the fact that each rule is concerned by a limited set of possible construction operations. A re-check will only be necessary if at least one of these operations has been used in the model increment.

For instance, the `OwnedElement` inconsistency rule is impacted by `setReference` operations that modify the values of the `ownedElement` set of an element. As a consequence, this rule should only be re-checked if the model increment changes the values of the `ownedElement` reference set of an element. Any other operation in the model increment will not affect the evaluation result of the inconsistency rule.

In order to analyze inconsistency rules and to identify the operations that impact them, we propose a partitioning of construction operations. Given a meta-model  $MM$  and the set  $\mathcal{O}_{MM}$  of all construction operations that can be performed to build model instances of this meta-model, we propose the partition  $\mathcal{P}_{impact}(\mathcal{O}_{MM})$  of  $\mathcal{O}_{MM}$ . Two construction operations  $o_1$  and  $o_2$  belong to the same equivalence class if and only if : (i)  $o_1$  and  $o_2$  both create model element instances of the same meta-class; or (ii)  $o_1$  and  $o_2$  both change the values of the same reference; or (iii)  $o_1$  and  $o_2$  both change the values of the same property; or (iv)  $o_1$  and  $o_2$  both delete a model element.

This partition is finite since a meta-model holds a finite number of meta-classes and each of them holds a finite number of properties and references. The partition can be automatically computed for any meta-model based on the following guidelines:

- for each non abstract meta-class  $M$  there is an equivalence class  $C_M$  that contains all the creation operations of instances of this meta-class,
- for each property  $p$  there is an equivalence class  $SP_p$  that contains all the operations setting the property value.
- for each reference  $r$  there is an equivalence class  $SR_r$  that contains all the operations setting the reference value.
- a final equivalence class  $D$  contains all the delete actions regardless of the metaclass of the deleted model element.

The first column of figure 4 represents the partition  $\mathcal{P}_{impact}(\mathcal{O}_{CMM})$  of the CMM metamodel.

## 4.2 Impact Matrix

The partition  $\mathcal{P}_{impact}$  is used to identify the operations that may impact an inconsistency rule. From a conceptual point of view, an inconsistency rule defines a selection of specific operations within a sequence of construction operations. This selection can be abstracted by a set of equivalence classes of the partition. We name this set the corresponding equivalence classes of a rule.

For example, for inconsistency rule **OwnedElement**, the corresponding set of equivalence classes is the singleton  $\{SR_{ownedElement}\}$ . For inconsistency rule **OwnedParameter**,  $\{SR_{ownedParameter}, SP_{direction}\}$  is the corresponding equivalence set.

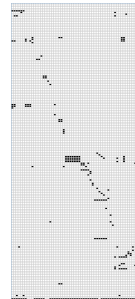
We can visualize the relation between equivalence classes and inconsistency rules by a matrix where the equivalence classes represent the rows and the inconsistency rules represent the columns. The matrix contains boolean values indicating the presence of a potential impact. The impact matrix for our CMM metamodel and the two inconsistency rules is shown in Figure 4.

This impact matrix can be used as a filter on the inconsistency rules that need to be re-checked after each model increment. For each operation contained in the model increment, the corresponding equivalence class is selected and the matrix is consulted to determine which rules need be re-checked. The impact matrix ensures that all rules whose evaluation may have changed will be re-checked. It should be noted that our approach only needs to store the impact matrix to run. The size of this impact matrix depends only on the number of rules and equivalence classes.

It should also be noted that our approach is a conservative approximation. It is possible that the impact matrix identifies rules to re-check even if their evaluation is not changed by the increment. Nonetheless, our approach effectively reduces the set of rules needed to be re-checked, thereby avoiding a waste of time on performing useless computations. We will present performance results in section 5.

	ownedElement	ownedParameter
$\hat{C}_{Package}$	false	false
$\hat{C}_{Class}$	false	false
$\hat{C}_{Operation}$	false	false
$\hat{C}_{Parameter}$	false	false
$\hat{S}P_{Name}$	false	false
$\hat{S}P_{Direction}$	false	<b>true</b>
$\hat{S}R_{ownedElement}$	<b>true</b>	false
$\hat{S}R_{ownedMember}$	false	false
$\hat{S}R_{ownedProperty}$	false	false
$\hat{S}R_{ownedParameter}$	false	<b>true</b>
$\hat{D}$	false	false

**Fig. 4.** Impact matrix for the CMM metamodel



**Fig. 5.** Impact matrix for the UML metamodel

### 4.3 Example

For the sequence  $\sigma_c$  of Figure 3 and the inconsistency rules of section 2.1, the model is consistent. Let  $\delta$  be the model increment of Figure 6 that creates two parameters and associates them with the ‘send’ operation through the `ownedParameter` reference. The first and second construction operations of  $\delta$  belong to equivalence class  $C_{Parameter}$ . The third operation belongs to equivalence class  $SR_{ownedParameter}$ . The fourth operation belongs to equivalence class  $SR_{ownedElement}$ . The impact matrix informs us that the rules `ownedElement` and `ownedParameter` have to be re-checked. Performing the re-check informs us that the model remains consistent after having applied the increment.

```

1 create(pa1,Parameter)
2 create(pa2,Parameter)
3 setReference(o1,ownedParameter,{pa1,pa2}) 1 setProperty(pa1,direction,{'return'})
4 setReference(o1,ownedElement,{pa1,pa2})   2 setProperty(pa2,direction,{'return'})

```

**Fig. 6.** a first model increment  $\delta$  on  $\sigma_c$

**Fig. 7.** a second model increment  $\delta'$  on  $\sigma_c.\delta$

Now, consider the second increment  $\delta'$  on  $\sigma_c.\delta$  in Figure 7 that changes the direction of the parameters. Both construction operations of  $\delta'$  belong to equivalence class  $SP_{Direction}$ . The impact matrix informs us that only `ownedParameter` rule needs to be re-checked. Computing the re-check only for this rule informs us that the model is inconsistent only for `ownedParameter`.

Our approach is centered around an impact matrix that expresses relationships between inconsistency detection rules and their equivalence classes. This matrix may be generated automatically, but such a generation depends on the language that is used to define the inconsistency rules. Indeed, the more expressive the language used to express the rule is, the more complex the automatic generation of the matrix will be. We will present in section 5 how we generated the impact matrix of UML 2.1 in a semi-automated way.

## 5 Validation

### 5.1 Prototype Implementation

In [9], we presented a global model inconsistency checker that has been realized using Prolog. Inconsistency rules were translated into Prolog queries and model construction operations were translated into Prolog facts. The global inconsistency checker has been integrated into the modeling environments Eclipse EMF and Rational Software Architect. It has been written in Java and is coupled with SWI-Prolog. From any given model, a model construction operation sequence is generated and added to the fact base. The Prolog engine then executes all queries representing inconsistency rules and returns the results to the user.

The Prolog query presented below corresponds to the inconsistency rule `OwnedParameter` we introduced in Section 2.1 to identify operations that own more than one ‘return’ parameter:

```
ownedParameter(X) :-
  lastCreate(X,Operation),
  lastSetReference(X,ownedParameter,L),
  lastSetProperty(Y,direction,'return'),
  lastSetProperty(Z,direction,'return'), Y\=Z,
  member(Y,L), member(Z,L).
```

When evaluating this query, Prolog returns all  $X$  such that `lastCreate(X, Operation)` is true in the sequence. For each identified operation  $X$ , Prolog will evaluate whether there are any pairs  $(Y,Z)$  of distinct return parameters owned by the operation. If the query returns a result for  $X$ , then the model is inconsistent since there is at least one operation in the resulting model that owns two return parameters.

The incremental checker we propose in this paper follows the architecture of the global inconsistency checker. It is also based on Prolog, the inconsistency rules are Prolog queries and the model construction operation sequences are stored in a Prolog fact base. The incremental checker differs from the global checker by relying on the impact matrix and by working with an extensible fact base in which new facts can be added dynamically. The incremental checker receives as input a sequence of construction operations that corresponds to a model increment of a sequence that is already stored in the fact base. It parses all operations of the increment. For each of them it uses the information stored in the impact matrix to mark all inconsistency rules that require a re-check. Once all operations of the model increment have been parsed, it is added to the fact base. The user is then asked whether he wants to perform an incremental re-check or whether he prefers to continue working with a possibly partially inconsistent model.

## 5.2 Case Study

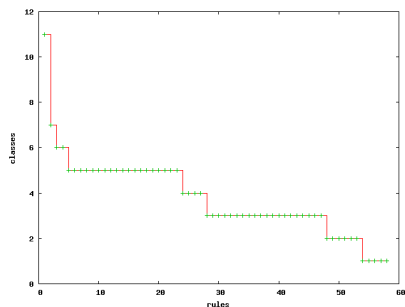
**The UML impact matrix.** We have validated our approach on the classes package of the UML 2 meta-model. The UML 2 classes package is composed of 55 meta-classes required to specify UML class diagrams. Those 55 meta-classes define a partition into 177 equivalence classes (cf. Section 4.1). The classes package defines 58 OCL constraints that we have considered as inconsistency rules. We translated these OCL constraints into Prolog queries and then built the impact matrix. The dimension of this matrix is  $177 \times 58$ .

In order to minimize errors when building the UML 2 matrix (which is quite big), we partially automate its construction. For that, we implemented a matrix builder that inputs inconsistency rules specified in Prolog and returns the corresponding impact matrix. It functions roughly as follows: (1) by default, all matrix values are set to `false`; (2) if the parsed inconsistency rule uses a `lastSetReference` or `lastSetProperty` construction operation, in the column

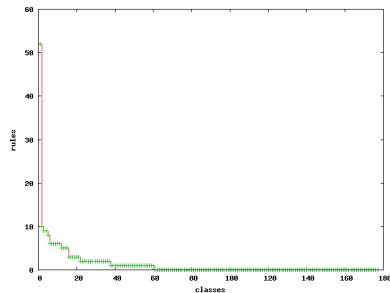
corresponding to the rule in the matrix, the equivalence class of the operation is set to `true`; (3) if the rule uses a `lastCreate`, in the column corresponding to the rule in the matrix, the equivalence class of create operation as well as the equivalence class  $D$  are set to `true`.

Figure 5 presents a screenshot of the UML 2 impact matrix where the 58 rules represent the columns and the 177 equivalence classes represent the rows. For the sake of visibility, a black square represents true while a white one represents false. It should be noted that the rules and the equivalence classes are ordered according to the meta-classes defining them. For instance, in the upper left corner of the matrix appear the rules that are defined in the `Association` meta-class and the equivalence classes of corresponding construction operations (i.e., `create(Association)` and `setReference(endType)`). The last line corresponds to the `delete` equivalence class; that's why it is quite black. Moreover, as inconsistency rules defined in a meta-class often use construction operations corresponding to the meta-class, there is a kind of diagonal of `true` values in the matrix. It should be noted that the block of  $4 \times 7$  `true` values in the middle of the matrix corresponds to the rules defined in the `MultiplicityElement` meta-class. This meta-class defines 7 inconsistency rules that specify the correct values of `lower` and `upper` multiplicities. Therefore, the `setProperty(lower)` and `setProperty(upper)` appear in all these rules. Finally, one can observe that the matrix is very sparse. As we will explain in the next subsection, many inconsistency rules are impacted by only a few equivalence classes.

**Analysis of the impact matrix.** Figure 8 is derived from the impact matrix and shows, for each rule, the number of equivalence classes that impact each inconsistency rule. The rules are shown on the x-axis, and are ordered according to the severity of their impact: one rule is impacted by 11 equivalence classes, one by 7 equivalence classes, two rules are impacted by 6 equivalence classes, 74.1% of the rules are impacted by 3 to 5 equivalence classes, and 18.9% of the inconsistency rules are impacted by 1 or 2 equivalence classes.



**Fig. 8.** Number of equivalence classes that impact a rule



**Fig. 9.** Number of rules impacted by each equivalence class

Figure 9 is also derived from the impact matrix and shows, for each equivalence class, the number of rules that are impacted by it. 66.1% of the equivalence classes do not affect model consistency, 22.0% of the classes impact 1 or 2 inconsistency rules, 9.0% of the classes impact 3 to 6 rules, 2.2% impact 8 to 10 rules and only the `delete` operation impacts almost all inconsistency rules. The case of the `delete` operation is particular. It really impacts nearly all UML OCL constraints but can only be performed on model elements that are not referenced by any other model element.

**Analysis of the rule complexity.** Next to this static analysis of the impact matrix, we have performed a complexity analysis of the inconsistency rules. Indeed, not all rules have the same complexity. In order to measure the complexity of a rule, we used a benchmark of the time needed to check each rule for different sizes of model chunks.

It appears that 13 rules out of 58 (22,4%) take much more time than the others to be checked. For a model size around three hundred thousand model elements (about 1.9 million operations), each of those 13 rules takes more than one second to be checked; all others need only a few milliseconds. The `ownedParameter` and the `ownedElement` rules we presented in the previous section belong to those 13 time-consuming rules. A manual inspection of those 13 rules revealed that 3 have a quadratic time complexity and the others have at most a linear complexity. The `ownedElement` rule we presented in the previous section is one of the three quadratic rules. The second one specifies that classifiers generalization hierarchies must be directed and acyclical, and the third one specifies that all members of a namespace should be distinguishable within it.

**Scalability analysis.** We stress tested our incremental checker on a real, large-scale UML model. A huge UML class model was obtained by reverse engineering the Azureus project, which possesses a messy architecture. The model construction sequence for this UML model contained about 1.9 million model construction operations.

We performed a static analysis of the construction operation sequence of the Azureus class model. According to our impact matrix, each rule is impacted on average by 42000 operations of the construction sequence. This means that, statistically, adding a new operation will have a probability of about 3% to require re-checking an inconsistency rule.

We also executed a runtime test of our incremental checker following the test performed by Eyged [6]. This test consists of loading a complete model and simulating all possible modifications that can be performed on all the model elements. Next, for each modification, an incremental check is performed. We have performed this runtime test on our Azureus model. As the Azureus model is a huge model, there are 1809652 modifications that can be realized. Those modifications have been automatically generated and for each of them an incremental check has been performed. The same test has been repeated six times in order to filter out possible noise. The result of this runtime test is that the worst time is 50.52 seconds (almost 1 minute), the best time is less than 0.1 ms (the

**Table 1.** Timing results in milliseconds (averaged over 6 runs) for incrementally checking the impact of modification operations applied to Azureus

model size	number of operations	worst result	best result	average result
part 1	380866	38204	$\approx 0$	1722
part 2	761732	38884	$\approx 0$	2677
part 3	1142598	41096	$\approx 0$	3725
part 4	1523464	47715	$\approx 0$	5168
full model	1904331	50521	$\approx 0$	5984

time needed to look in the matrix that no rule needs to be re-checked) and the average time is 6 seconds (cf. last column of Table 1).

In order to analyze the effect of model size on the performance of our consistency checker, we have split up the Azureus model into five parts with a linearly increasing size (the fifth part corresponding to the complete Azureus model) and we have applied the same runtime test but with a set of modification operations corresponding to the size of the part. Applying the runtime test to those sub-models, it turns out that best time remains roughly the same whereas the worst time has a curious growth. In fact, we did not observe (as we would have expected) a quadratic trend that would correspond to the time needed to check the most time-consuming rules. The reason is that the inconsistencies are not uniformly distributed among parts. Our hypothesis is that the worst time depends mainly on the ordering of the operations within the sequence. Finally, we have observed that the average time increases linearly (cf. last column of Table 1). This was confirmed by a linear regression model that had a very high “goodness of fit”, since the coefficient of determination  $R^2 = 0.994$  was very close to 1.

Without anticipating our conclusion, those timing results seem to show that, if the inconsistency rule set contains complex rules (such as `ownedElement`), once the model size becomes important (in the order of millions operations), the time needed to perform an incremental check cannot be instantaneous and continues to increase as the model size increases.

## 6 Related Work

Egyed proposed a framework dedicated to instant inconsistency detection [6]. This framework monitors the model elements that need to be analyzed during the check of an inconsistency rule. If an inconsistency is detected, all the relevant model elements are inserted in a corresponding “rule scope” in order to keep track of them (a rule scope defines a relation between one inconsistency detection rule and the set of model elements that need to be analyzed to evaluate this rule). After a set of modifications, the framework traces the rule scopes that are impacted by the modifications and then automatically re-checks the corresponding rules. This allows to reduce the set of rules to re-check and the set of model elements to analyze. Egyed presents very efficient performance charts for his approach but also makes the observation that such results are due to



the rules that have been considered. Indeed, all considered rules have only one root model element and their check only needs a bounded set of model elements linked either directly or indirectly to the root. With such rules, the size of the rule scope scales and the time needed to perform the check is almost instantaneous, independently of the model size. This is confirmed by our findings, but we would like to stress that not all inconsistency rules are of this kind. If we would apply Egyed’s approach to the `OwnedElement` rule on our sample model (cf. figure 2), 4 “rule scopes” will be built (one per model element). The size of those rule scopes will depend on the model size (the rule scope for the `Azureus` package will own all elements of the model). Now, if we consider that a modification changes the name of the `send` operation then three rule scopes will be impacted (the ones of the `send` operation, of the `Server` class and of the `Azureus` package). Then the complex `ownedElement` rule will be checked three times and the check will require some time. To conclude, the more complex the inconsistency rules and the bigger the models, the less efficient Egyed’s approach (or any other approach, for that matter), becomes.

Wagner et al. also provide a framework for incremental consistency checking in [16]. The framework monitors change events and tries to match them against detection rules that are defined as graph grammar patterns. If a match is detected then the rule is automatically re-checked. Wagner does not provide any performance analysis and does not ensure that his approach is scalable. Indeed, Wagner indicates that rules should not be time consuming in order to not block the user while he is building his models.

In [17] is presented an OCL incremental checker. The authors describe how to exploit the OCL description language to work on consistency invariant. The approach can be compared to ours because it enables to determine for each invariant the set of impacting OCL change events. And secondly, it describes how to compute an optimized invariant recheck code for each impacting change. However, OCL description language has a limited usage as it can only describe mono-contextual inconsistencies, in the context of software architecture models it is advocated to target multi-context/multi-paradigm inconsistencies as presented in [18,19].

In the database community, incremental consistency checking is an important research topic that has been addressed by various authors over the years. Their main goal is to preserve data integrity, and to detect whether database updates violate integrity constraints. For example, [20] proposed a logic-based approach, implemented in Prolog, to check integrity of deductive databases. We acknowledge that there is a lot to learn from database research, even though the focus for software models is different, since inconsistencies are omnipresent and inevitable during the modeling process [3], implying that we need more flexible techniques for managing and allowing inconsistencies.

## 7 Conclusion

In this paper we proposed an incremental inconsistency checker that is based on a sequence of model construction operations. Our approach consists of analyzing

the modifications performed on a model in order to identify a subset of inconsistency rules that need to be re-checked. The analysis is based on an impact matrix that represents dependencies between construction operation equivalence classes and inconsistency rules. Thanks to this matrix, a user can instantaneously know if the modifications he performs may or may not impact an inconsistency rule. With such knowledge he can then decide whether and when to execute the incremental check of impacted rules. Such an incremental check typically requires considerably less time than a full check. Moreover, our incremental checker scales up to huge models, as the only information required for it to run is stored in the impact matrix.

The definition of the impact matrix relies only on the meta-model and the inconsistency rules; it does not depend on the state of the models that are checked. Our incremental inconsistency checker can even consider several meta-models simultaneously in a homogenous way, since we represent models as construction operation sequences defined independently of any meta-model.

We aim to improve our incremental checker in two ways. First, we aim to classify inconsistency rules according to their severity and complexity. With such a classification, users will have more information to decide whether and when to re-check inconsistency detection rules that have been marked by our incremental checker. As a second improvement, we can not only reduce the set of rules to re-check but also the set of model elements to consider during the analysis. This would enable each rule to re-check only a relevant fragment of the whole model. Our objective is then to integrate our approach with an incremental checker such as the one proposed in [6].

The results we obtained can also contribute towards computer-supported collaborative work. Indeed, we have observed that many model construction operations are safe regarding inconsistency rules. In other words, those operations have no major negative impact on the model consistency, and can thus be performed by any one at any time. Hence, it would make sense to define a locking and transaction mechanism on top of construction operations instead of model elements in order to improve support for collaborative work.

## References

1. Selic, B.: The pragmatics of model-driven development. *IEEE Software* 20(5), 19–25 (2003)
2. Finkelstein, A.C.W., et al.: Inconsistency handling in multiperspective specifications. *IEEE Trans. Softw. Eng.* 20, 569–578 (1994)
3. Balzer, R.: Tolerating inconsistency. In: *Proc. Int' Conf. Software engineering (ICSE 1991)*, vol. 1, pp. 158–165 (1991)
4. Fradet, P., Le Metayer, D., Peiin, M.: Consistency checking for multiple view software architectures. In: *Proc. Joint Conf. ESEC/FSE 1999*, vol. 41, pp. 410–428. Springer, Heidelberg (1999)
5. 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 Press, Los Alamitos (2003)

6. Egyed, A.: Instant consistency checking for UML. In: Proceedings Int'l Conf. Software Engineering (ICSE 2006), pp. 381–390. ACM Press, New York (2006)
7. 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)
8. Malgouyres, H., Motet, G.: A UML model consistency verification approach based on meta-modeling formalization. In: SAC 2006, pp. 1804–1809. ACM, New York (2006)
9. 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 Press, New York (2008)
10. Egyed, A.: Fixing inconsistencies in UML design models. In: Proc. Int'l Conf. Software Engineering (ICSE 2007), pp. 292–301. IEEE Computer Society, Los Alamitos (2007)
11. Spanoudakis, G., Zisman, A.: Inconsistency management in software engineering: Survey and open research issues. Handbook of Software Engineering and Knowledge Engineering, 329–380 (2001)
12. 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)
13. Elaasar, M., Brian, L.: An overview of UML consistency management. Technical Report SCE-04-18 (August 2004)
14. OMG: Unified Modeling Language: Super Structure version 2.1 (January 2006)
15. OMG: Meta Object Facility (MOF) 2.0 Core Specification (January 2006)
16. Wagner, R., Giese, H., Nickel, U.A.: A plug-in for flexible and incremental consistency management. In: Workshop on consistency problems in UML-based Software Development - Satellite Workshop of MODELS (2003)
17. Cabot, J., Teniente, E.: Incremental evaluation of ocl constraints. In: Dubois, E., Pohl, K. (eds.) CAiSE 2006. LNCS, vol. 4001, pp. 81–95. Springer, Heidelberg (2006)
18. ISO/IEC 42010: Systems and software engineering architectural description. ISO/IEC WD3 42010 and IEEE P42010/D3 (2008)
19. Boiten, E., et al.: Issues in multiparadigm viewpoint specification. In: Foundations of Software Engineering, pp. 162–166 (1996)
20. Kowalski, R.A., Sadri, F., Soper, P.: Integrity checking in deductive databases. In: Proc. Int'l Conf. Very Large Data Bases (VLDB), pp. 61–69. Morgan Kaufmann, San Francisco (1987)

# Reasoning on UML Conceptual Schemas with Operations

Anna Queralt and Ernest Teniente

Universitat Politècnica de Catalunya  
{aqueralt,teniente}@lsi.upc.edu

**Abstract.** A conceptual schema specifies the relevant information about the domain and how this information changes as a result of the execution of operations. The purpose of reasoning on a conceptual schema is to check whether the conceptual schema is correctly specified. This task is not fully formalizable, so it is desirable to provide the designer with tools that assist him or her in the validation process. To this end, we present a method to translate a conceptual schema with operations into logic, and then propose a set of validation tests that allow assessing the (un)correctness of the schema. These tests are formulated in such a way that a generic reasoning method can be used to check them. To show the feasibility of our approach, we use an implementation of an existing reasoning method.

**Keywords:** Conceptual modeling, automatic reasoning, operation contracts.

## 1 Introduction

The correctness of an information system is largely determined during requirements specification and conceptual modeling, since errors introduced at these stages are usually more expensive to correct than those made during design or implementation. Thus, it is desirable to detect and correct errors as early as possible in the software development process. Moreover, this is one of the key problems to solve for achieving the goal of automating information systems building [15].

The correctness of a conceptual schema can be seen from two different points of view. From an internal point of view, correctness can be determined by reasoning on the definition of the schema itself, without taking the user requirements into account. This is equivalent to answering to the question *Is the conceptual schema right?*. There are some typical properties that can be automatically tested to determine this kind of correctness like schema satisfiability, operation executability, etc.

On the other hand, from an external point of view, correctness refers to the accuracy of the conceptual schema regarding the user requirements [1] and it can be established by answering to the question *Are we building the right conceptual schema?*. Testing whether a schema is correct in this sense may not be completely automated since it necessarily requires the user intervention. Nevertheless, it is desirable to provide the designer with a set of tools that assist him during the validation process.

A *conceptual schema* consists of a *structural part*, which defines the relevant static aspects of the domain, and a *behavioral part*, which specifies how the information

represented in the structural part changes as a result of the execution of system operations [11]. System operations specify the response of the system to the occurrence of some event in the domain, viewing the system as a black box and, thus, they are not assigned to classes. They define the only changes that can be performed on the IB.

Figs. 1 and 2 show a possibly incorrect conceptual schema of a (simplified) on-line auction site that we will use as a running example.

A test that the designer can perform to validate the internal correctness of the structural schema is to check whether it is satisfiable, that is, if it accepts at least one instance satisfying all the constraints. In our example, the following instantiation: *"Mick is a registered user who owns a book, and bids 200\$ for a bicycle, owned by Angie, who had set a starting price of 180\$"* satisfies all the graphical and textual constraints, which demonstrates that the structural schema is satisfiable.

However, the fact that the structural part is satisfiable does not necessarily imply that the whole conceptual schema also is. That is, when we take into account that the only changes admitted are those specified in the behavioral schema, it may happen that the properties fulfilled by the structural schema alone are no longer satisfied.

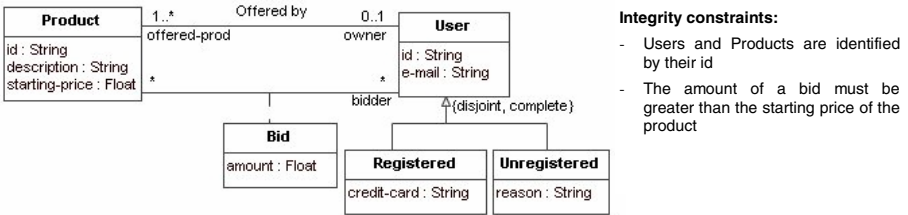


Fig. 1. The structural schema of an on-line auction site

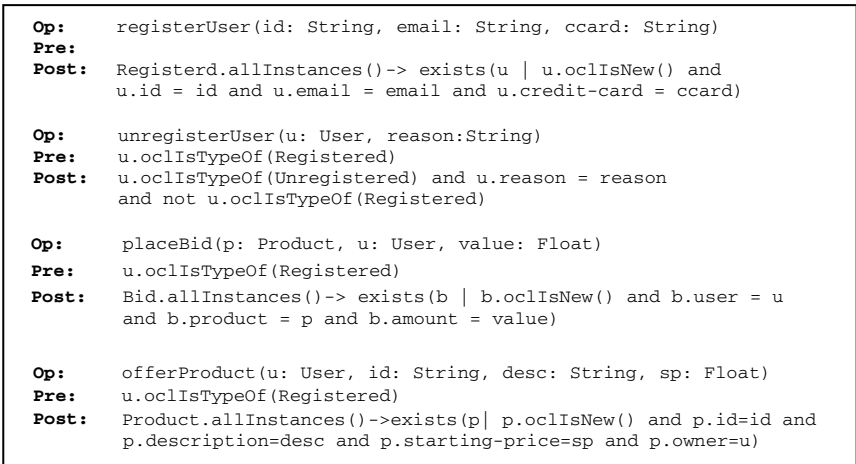


Fig. 2. A partial behavioral schema corresponding to the structural schema of Figure 1

In our example, although it is possible to find instances of *User* satisfying all the constraints as we have just seen, there is no operation that successfully populates this class. The operation *registerUser* seems to have this purpose but it never succeeds since it does not associate the new user with a *Product* by means of *Offered by*, which violates the cardinality constraint of the role *offered-prod*. As a consequence, since the only operation that creates a product (i.e. *offerProduct*) requires an existing user, there can not be any instance of *Product* either. Then, we have that this schema can never be populated using the operations defined and, although the structural part of the schema is semantically correct, the whole conceptual schema is not.

The main contribution of this work is to propose an approach to help to validate a conceptual schema with a behavioral part. To do this, we provide a method to translate a UML schema, with its behavioral part consisting of operations specified in OCL, into a set of logic formulas. The result of this translation is such that ensures that the only changes allowed are those specified in the behavioral schema, and can be validated using any existing reasoning method capable to deal with negation of derived predicates. To our knowledge, ours is the first approach that validates jointly the structural and behavioral parts of a UML/OCL conceptual schema.

We provide the designer with several validation tests which allow checking the correctness of a schema from the internal and external points of view mentioned above. Some of the tests are automatic and are directly generated from the conceptual schema while others are user-defined and give the designer the freedom to ask whichever questions he wants regarding situations that hold (do not hold) in the domain to ensure that they are (not) accepted by the schema. In both cases, the designer intervention is required to fix any problem detected by the tests.

We also show the feasibility of our approach by using an implementation of an existing reasoning method, which has had to be extended for our purposes.

Basic concepts are introduced in section 2. Section 3 presents our method to translate a schema with operations into logic. Section 4 presents our approach to validation. Section 5 shows its feasibility by means of an implementation. Section 6 reviews related work. Finally, we present our conclusions in section 7.

## 2 Basic Concepts

The *structural schema* consists of a taxonomy of entity types together with their attributes, a taxonomy of associations among entity types, and a set of integrity constraints over the state of the domain, which define conditions that each instantiation of the schema, i.e. each state of the information base (IB), must satisfy. Those constraints may have a graphical representation or can be defined by means of a particular general-purpose language.

In UML, a structural schema is represented by means of a class diagram, with its graphical constraints, together with a set of user-defined constraints, which can be specified in any language (Figure 1). As proposed in [21], we will assume these constraints are specified in OCL.

The content of the IB changes due to the execution of operations. The *behavioral schema* contains a set of *system operations* and the definition of their effect on the IB. System operations specify the response of the system to the occurrence of some event in the domain, viewing the system as a black box and, thus, they are not assigned to classes [11]. These operations define the only changes that can be performed on the IB.

An operation is defined by means of an operation contract, with a *precondition*, which expresses a condition that must be satisfied when the call to the operation is done, and a *postcondition*, which expresses a condition that the new state of the IB must satisfy. The execution of an operation results in a set of one or more *structural events* to be applied to the IB. Structural events are elementary changes on the content of the IB, that is, insertions or deletions of instances. We assume a strict interpretation of operation contracts [17] which prevents the application of an operation if any constraint is violated by the state satisfying the postcondition.

The operation contracts of the behavioral schema of our running example are shown in Figure 2. Each contract describes the changes that occur in the IB when the operation is invoked. Since we assume a strict interpretation, there is no need to include preconditions to guarantee the satisfaction of integrity constraints. However, if those preconditions were added, they would also be correctly handled by our method.

As we will see in Section 3, we translate a UML and OCL schema such as the one of the example into a set of first-order logic formulas in order to use a reasoning method to determine several properties on it. The OCL considered consists of all the OCL operations that result in a boolean value, including select and size, which can also be handled by our method despite returning a collection and an integer. The logic formalization of the schema consists of a set of rules and conditions defined as follows.

A *term* is either a variable or a constant. If  $p$  is a  $n$ -ary predicate and  $T_1, \dots, T_n$  are terms, then  $p(T_1, \dots, T_n)$  or  $p(\bar{T})$  is an *atom*. An *ordinary literal* is either an atom or a negated atom, i.e.  $\neg p(\bar{T})$ . A *built-in literal* has the form of  $A_1 \theta A_2$ , where  $A_1$  and  $A_2$  are terms. Operator  $\theta$  is either  $<, \leq, >, \geq, =$  or  $\neq$ .

A normal clause has the form:  $A \leftarrow L_1 \wedge \dots \wedge L_m$  with  $m \geq 0$ , where  $A$  is an atom and each  $L_i$  is a literal, either ordinary or built-in. All the variables in  $A$ , as well as in each  $L_i$ , are assumed to be universally quantified over the whole formula.  $A$  is the *head* and  $L_1 \wedge \dots \wedge L_m$  is the *body* of the clause. A *fact* is a normal clause of the form  $p(\bar{a})$ , where  $p(\bar{a})$  is a ground atom. A *deductive rule* is a normal clause of the form:  $p(\bar{T}) \leftarrow L_1 \wedge \dots \wedge L_m$  with  $m \geq 1$ , where  $p$  is the *derived* predicate defined by the deductive rule. A *condition* is a formula of the (*denial*) form:  $\leftarrow L_1 \wedge \dots \wedge L_m$  with  $m \geq 1$ .

Finally, a *schema*  $S$  is a tuple  $(DR, IC)$  where  $DR$  is a finite set of deductive rules and  $IC$  is a finite set of conditions. All these formulas are required to be *safe*, that is, every variable occurring in their head or in negative or built-in literals must also occur in an ordinary positive literal of the same body. An *instance of a schema*  $S$  is a tuple  $(E, S)$  where  $E$  is a set of facts about base predicates.  $DR(E)$  denotes the whole set of ground facts about base and derived predicates that are inferred from an instance  $(E, S)$ , and corresponds to the fixpoint model of  $DR \cup E$ .

### 3 Translation of the Conceptual Schema into Logic

Validation tests that consider the structural schema alone are aimed at checking that an instantiation fulfilling a certain property and satisfying the integrity constraints can exist. In this case, classes, attributes and associations can be translated into base predicates that can be instantiated as desired, as long as integrity constraints are satisfied, in order to find a state of the IB that proves a certain property [16].

However, when considering also the behavioral schema, the population of classes and associations is only determined by the events that have occurred. In other words,

the state of the IB at a certain time  $t$  is just the result of all the operations that have been executed before  $t$ , since the instances of classes and associations cannot be created or deleted as desired. For instance, according to our schema in Fig. 1 and the operations defined, *Angie* may only be an instance of *Registered* at a time  $t$  if the operation *registerUser* has created it at some time before  $t$  and the operation *unregisterUser* has not removed it between its creation and  $t$ .

For this reason, it must be guaranteed that the population of classes and associations at a certain time depends on the operations executed up to that moment. To do this, we propose that operations are the basic predicates of our logic formalization, since their instances are directly created by the user. Classes and associations will be represented by means of derived predicates instead of basic ones, and their derivation rules will ensure that their instances are precisely given by the operations executed.

This approach clearly differs from our previous work [16, 18], where we proposed to formalize classes, attributes and associations as base predicates. Note, however, that a formalization of this kind does not ensure that instances of classes and associations result from the execution of operations.

### 3.1 Deriving Instances from Operations

Classes and associations are represented by means of derived predicates whose derivation rules ensure that their instances are given by the occurrence of operations, which are the base predicates of our formalization of the schema. Then, an instance of a predicate  $p$  representing a class or association exists at time  $t$  if it has been added by an operation at some time  $t2$  before  $t$ , and has not been deleted by any operation between  $t2$  and  $t$ . Formally, the general derivation rule is:

$$\begin{aligned} p([P,]P_1, \dots, P_m, T) &\leftarrow \text{add}P([P,]P_1, \dots, P_m, T2) \wedge \neg \text{deleted}P(P_i, \dots, P_j, T2, T) \wedge T2 \leq T \wedge \text{time}(T) \\ \text{deleted}P(P_i, \dots, P_j, T1, T2) &\leftarrow \text{del}P(P_i, \dots, P_j, T) \wedge T > T1 \wedge T \leq T2 \wedge \text{time}(T1) \wedge \text{time}(T2) \end{aligned}$$

where  $P$  is the OID (Object Identifier), which is included if  $p$  is a class.  $P_i, \dots, P_j$  are the terms of  $p$  that suffice to identify an instance of  $p$  according to the constraints defined in the schema. In particular, if  $p$  is a class (or association class),  $P = P_i = P_j$ . The predicate *time* indicates which are the time variables that appear in the derived predicate we are defining. As well as those representing operations, *time* is a base predicate since its instances cannot be deduced from the rest of the schema. Predicates *addP* and *delP* are also derived predicates that hold if some operation has created or deleted an instance of  $p$  at time  $T$ , respectively. They are formalized as follows.

Let *op-addP<sub>i</sub>* be an operation of the behavioral schema, with parameters  $Par_1, \dots, Par_n$  and precondition  $pre_i$  such that its postcondition specifies the creation of an instance of a class or association  $p$ . For each such operation we define the following rule:

$$\text{add}P([P,]Par_1, \dots, Par_k, T) \leftarrow \text{op-add}P_i([P,]Par_1, \dots, Par_m, T) \wedge pre_i(T_{pre}) \wedge T_{pre} = T - I \wedge \text{time}(T)$$

where  $Par_1, \dots, Par_k$  are those parameters of the operation that indicate the information required by the predicate  $p$ , and  $T$  is the time in which the operation occurs. The literal  $pre_i(T_{pre})$  is the translation of the precondition of the operation, following the same rules used to translate OCL integrity constraints [16]. Note that, since the precondition must hold just before the occurrence of the operation, the time of all its facts is  $T - I$ .



Similarly, for each operation  $op-delP_i(Par_1, \dots, Par_m, T)$  with precondition  $pre_i$  that deletes an instance of  $p$  we define the derivation rule:

$$delP(Par_1, \dots, Par_j, T) \leftarrow op-delP_i(Par_1, \dots, Par_m, T) \wedge pre_i(T_{pre}) \wedge T_{pre} = T - I \wedge time(T)$$

where  $Par_1, \dots, Par_j$  are those parameters of the operation that identify the instance to be deleted. Thus, if  $p$  is a class or association class,  $delP$  will have a single term in addition to  $T$ , which corresponds to the OID of the deleted instance.

To completely define the above derivation rules for each predicate representing an element of the structural schema, we need to know which OCL operations of the behavioral schema are responsible for creating or deleting its instances. For our purpose, we assume that operations create instances with the information given by the parameters or delete instances that are given as parameters. A single operation can create and/or delete several instances. We are not interested in query operations since they do not affect the correctness of the schema.

Several OCL expressions can be used to specify that an instance exists or not at postcondition time. For the sake of simplicity, we consider a single way to specify each of these conditions, since other OCL expressions with equivalent meaning can be easily rewritten in terms of the ones we consider. Under this assumption, we define the rules to identify the creation and deletion of instances in OCL postconditions:

- R1. An instance  $c(I, A_1, \dots, A_m, T)$  of a class  $C$  is added by an operation if its postcondition includes the OCL expression:  $C.allInstances() \rightarrow exists(i | i.ooclIsNew() \text{ and } i.attr_1 = a_1)$  or the expression:  $i.ooclIsTypeOf(C)$  and  $i.attr_1 = a_1$ , where each  $attr_i$  is a single-valued attribute of  $C$ .
- R2. An instance  $c(I, P_1, \dots, P_m, A_1, \dots, A_m, T)$  of an association class  $C$  is added by an operation if its postcondition includes the expression:  $C.allInstances() \rightarrow exists(i | i.ooclIsNew() \text{ and } i.part_1 = p_1 \text{ and } \dots \text{ and } i.part_n = p_n \text{ and } i.attr_1 = a_1 \text{ and } \dots \text{ and } i.attr_m = a_m)$  or the expression:  $i.ooclIsTypeOf(C)$  and  $i.part_1 = p_1$  and  $\dots$  and  $i.part_n = p_n$  and  $i.attr_1 = a_1$  and  $\dots$  and  $i.attr_m = a_m$ , where each  $part_i$  is a participant that defines the association class, and each  $attr_j$  is a single-valued attribute of  $C$ .
- R3. An instance  $r(C_1, C_2, T)$  of a binary association  $R$  between objects  $C_1$  and  $C_2$ , with roles  $role-c_1$  and  $role-c_2$  in  $r$  is added by an operation if its postcondition contains the OCL expression:  $c_1.role-c_2 = c_2$ , if the multiplicity of  $role-c_2$  is at most 1 or the expression:  $c_1.role-c_2 \rightarrow includes(c_2)$ , if the multiplicity of  $role-c_2$  is greater than 1. This rule also applies to multi-valued attributes. Creation or deletion of instances of n-ary associations with  $n > 2$  cannot be expressed in OCL unless they are association classes, which are considered in the previous rule.
- R4. An instance  $c(I, A_1, \dots, A_m, T)$  of a class  $C$  is deleted by an operation if its postcondition includes the expression:  $C_{gen}.allInstances() \rightarrow excludes(i)$  or the expression:  $not\ i.ooclIsTypeOf(C_{gen})$ , where  $C_{gen}$  is either the class  $C$  or a superclass of  $C$ .
- R5. An instance  $c(I, P_1, \dots, P_m, A_1, \dots, A_m, T)$  of an association class is deleted by an operation if its postcondition includes:  $C.allInstances() \rightarrow excludes(i)$ , or:  $not\ i.ooclIsTypeOf(C)$ , or if any of its participants  $(P_1, \dots, P_n)$  is deleted.
- R6. An instance  $r(C_1, C_2, T)$  of a binary association  $R$  between objects  $C_1$  and  $C_2$ , with roles  $role-c_1$  and  $role-c_2$  in  $r$  is deleted by an operation if its postcondition includes:  $c_1.role-c_2 \rightarrow excludes(c_2)$  or if any of its participants  $(C_1 \text{ or } C_2)$  is deleted.

For instance, according to the previous translation rules, the class *Registered* of our example will be represented by means of the clauses:

$$\begin{aligned} \text{registered}(U, Id, Email, Ccard, T) &\leftarrow \text{addRegistered}(U, Id, Email, Ccard, T2) \\ &\quad \wedge \neg \text{deletedRegistered}(U, T2, T) \wedge T2 \leq T \wedge \text{time}(T) \\ \text{deletedRegistered}(U, T1, T2) &\leftarrow \text{delRegistered}(U, T) \wedge T > T1 \wedge T \leq T2 \wedge \text{time}(T1) \wedge \text{time}(T2) \end{aligned}$$

where  $U$  corresponds to the unique OID required by every instance of a class. In turn, *addRegistered* and *delRegistered* are derived predicates whose definition depends on the operations of the behavioral schema that insert and delete instances of the class *Registered*. The operation *registerUser* creates an instance of *registered*( $U, Id, Email, C\text{-}card, T$ ) according to R1, since its postcondition includes the expression `Registered.allInstances()->exists(u | u.ocIsNew() and u.e-mail=e-mail and u.id=id and u.credit-card=ccard)`. Since the other operations do not create instances of *Registered*, there is a single derivation rule for *addRegistered*:

$$\text{addRegistered}(U, Id, Email, Ccard, T) \leftarrow \text{registerUser}(U, Id, Email, Ccard, T) \wedge \text{time}(T)$$

We also need to find which operations are responsible for deleting instances of *Registered* in order to specify the derivation rule of *delRegistered*. The operation *unregisterUser* is the only one that deletes instances of *Registered* according to R4, since it includes the OCL expression `not u.ocIsTypeOf(Registered)`. Its postcondition also includes the creation of an unregistered user, but this will be taken into account when specifying the derivation rules of *addUnregistered* for predicate *unregistered*. This time the precondition is not empty, and requires that  $u$  is an instance of *Registered*, so the derivation rule in this case is:

$$\text{delRegistered}(U, T) \leftarrow \text{unregisterUser}(U, T) \wedge \text{registered}(U, Id, E, Cc, T_{pre}) \wedge T_{pre} = T - 1 \wedge \text{time}(T)$$

Since a modification can be regarded as a deletion followed by an insertion, no specific derived predicates are needed to deal with them.

### 3.2 Constraints Generated

Since we assume that events cannot happen simultaneously, we need to define constraints to guarantee that two operations cannot occur at the same time. Constraints are expressed as formulas in denial form, which represent conditions that cannot hold in any state of the IB. Therefore, for each operation  $o$  with parameters  $P_1, \dots, P_n$  we define the following constraint for each parameter  $P_i$ :

$$\leftarrow o(P_1 1, \dots, P_n 1, T) \wedge o(P_1 2, \dots, P_n 2, T) \wedge P_i 1 \langle \rangle P_i 2$$

And for each pair  $o_1, o_2$  of operations we define the constraint:

$$\leftarrow o_1(P_1, \dots, P_m, T) \wedge o_2(Q_1, \dots, Q_m, T)$$

In our example, *unregisterUser*( $U, Reason, T$ ) requires the constraints:

$$\begin{aligned} &\leftarrow \text{unregisterUser}(U, R, T) \wedge \text{unregisterUser}(U2, R2, T) \wedge U \langle \rangle U2 \\ &\leftarrow \text{unregisterUser}(U, R, T) \wedge \text{unregisterUser}(U2, R2, T) \wedge R \langle \rangle R2 \end{aligned}$$

and, for each other operation of the schema, a constraint like:

$$\leftarrow \text{unregisterUser}(U, R, T) \wedge \text{registerUser}(Id, Email, Ccard, T)$$

Moreover, the constraints of the UML structural schema are also translated into this kind of formulas. The set of constraints needed is exactly the one resulting from the translation of the structural schema [16], but now they are defined in terms of derived predicates instead of basic ones.

## 4 Our Approach to Validation

Our approach to validation is aimed at providing the designer with different kinds of tests that allow him to assess the correctness of the conceptual schema being defined. All of them take into account both the structural and the behavioral parts of the conceptual schema.

We express all tests in terms of checking the satisfiability of a derived predicate. So, for each validation test to be performed, a derived predicate (with its corresponding derivation rule) that formalizes the desired test is defined. With this input, together with the translated schema itself, any satisfiability checking method that is able to deal with negation of derived predicates can be used to validate the schema. We illustrate our approach using the translation of our example obtained as explained in Section 3.

### 4.1 Is the Conceptual Schema Right?

The tests devoted to check the internal correctness of the schema can be automatically defined, i.e. they can be performed without the designer intervention. Some of them correspond to well known reasoning tasks (such as schema satisfiability) while others refer to additional properties that can be automatically drawn from the conceptual schema and which are an original contribution of this paper.

#### 4.1.1 Checking Strong Satisfiability

A schema is strongly satisfiable if there is at least one fully populated state of the IB satisfying all the constraints [12]. In the presence of operations, this means checking whether they allow creating at least a complete valid instantiation.

To perform this test, we need to define a derived predicate such that it is true when the schema is strongly satisfiable, i.e. if it is possible to have an instance of all classes and associations of the schema. In our example:

$$\text{sat} \leftarrow \text{registered}(U, \text{Uid}, \text{Email}, \text{Ccard}, T) \wedge \text{unregistered}(U2, \text{Uid2}, \text{Email2}, \text{Reason}, T) \wedge \\ \text{product}(P, \text{Pid}, \text{Descr}, \text{St-pr}, T) \wedge \text{bid}(B, \text{Prod}, \text{Bidder}, \text{Amt}, T) \wedge \text{offeredBy}(P2, \text{Owner}, T)$$

As we discussed in the introduction, the schema of our example is not strongly satisfiable when the behavior of the operations is taken into account. To avoid this mistake, we may replace the original operation *registerUser* by the following one responsible for creating both an instance of *Registered* and an instance of a *Product* that will be offered by the new user when he is registered:

```
Op:      registerUser(id: String, email: String, ccard: String, pid:
           String, descr: String, st-price: Float)
Pre:
Post:   Registered.allInstances()->exists(u|u.oclIsNew() and u.e-mail
           = email and u.c-card=ccard and u.offered-prod->exists(p |
           p.oclIsNew() and p.id=pid and p.description=descr and
           p.starting-price=st-price))
```

Now, if we check satisfiability of the predicate *sat*, the answer is that the schema is strongly satisfiable. The following sample instantiation shows that all classes can be populated at time 4. It only includes instances of base predicates, since the derived ones can be obtained from them. Since our base predicates correspond to the operations, the sample instantiations obtained give a sequence of operation calls that leads to a state that is valid according to the schema:

```
{registerUser(john, john@upc.edu, 111, p1, pen, 10, 1), unregisterUser(john, 2),
  registerUser(mary, mary@upc.edu, 222, p2, pen, 20, 3), placeBid(mary, p1, 25, 4)}
```

That is, we need to register a new user *John* at time 1 and then unregister him to have an instance of *Unregistered*. After that, we create another user *Mary* to have an instance of *Registered*. Finally, to populate the class *Bid*, *Mary* bids for the pen *p1*.

#### 4.1.2 Automatically Generated Tests

Following the ideas suggested by model-based testing approaches [20], there are some tests that can be automatically drawn from the concrete schema to be validated. As usual, they will help the designer to detect potentially undesirable situations admitted by the schema. Note, however, that we can already determine these situations at the conceptual schema level while, in general, model-based testing requires an implementation of the software system to execute the tests. The definition of an exhaustive list of such kind of tests is out of the scope of this paper.

For instance, in our example, although a product may have no owner according to the cardinality constraint 0..1 of *owner*, it will always have exactly one owner in practice with the given operations. This means that there is probably something that the designer overlooked when specifying the behavioral schema like an operation to allow users withdrawing offered products or that the cardinality constraint should be just 1.

The derivation rule that formalizes this situation, which can be automatically generated from the information provided by the conceptual schema, is the following:

$$\begin{aligned} \text{unownedProd} &\leftarrow \text{product}(P, Id, Descr, St-price, T) \wedge \neg \text{hasOwner}(P, T) \\ \text{hasOwner}(P, T) &\leftarrow \text{offeredBy}(P, Owner, T) \end{aligned}$$

The absence of a sequence of operations satisfying *unownedProd* shows that the conceptual schema does not admit products without owner and, therefore, that the cardinality constraint of *owner* is not properly defined. We assume that the designer decides to define the cardinality constraint of *owner* to exactly 1 to fix this situation.

The general form of the previous test is as follows. If the predicate *minCardAssoc* is not satisfiable, it means that there is a potentially undesirable situation:

$$\begin{aligned} \text{minCardAssoc} &\leftarrow \text{classJ}(P_j, \dots, T) \wedge \neg \text{hasAssoc}(P_j, T) \\ \text{hasAssoc}(P, T) &\leftarrow \text{assoc}([A], P_1, \dots, P_i, \dots, P_m, T) \end{aligned}$$

for each  $j < i$  representing a participant of *Assoc*, where  $P_i$  is the participant with minimum cardinality 0.

#### 4.1.3 Testing Properties of the Operations

When dealing with operations additional validation tests can be performed, namely applicability and executability of each operation [4]. An operation is *applicable* if there is a state where its precondition holds. An operation is *executable* if it can be

executed at least once, i.e. if there is a state where its postcondition holds, together with the integrity constraints, and such that its precondition was also true in the previous state.

To illustrate these properties, let us consider an additional operation *removeProduct* to delete products:

```
Op:    removeProduct(p: Product)
Pre:    p.owner->isEmpty()
Post:    Product.allInstances()->excludes(p)
```

As can be seen, the precondition of this operation requires that the product being removed has no owner, which is not possible according to the cardinality constraint 1 of *owner*, just redefined in the previous section. This means that this operation is not applicable and the designer should avoid this situation by, for example, modifying the precondition.

The formalization for an operation *O* with precondition *pre(t)* is:

$$applicable\_O \leftarrow pre(T)$$

If this *applicable\_O* is not satisfiable, the operation is not applicable.

Although an operation is applicable, it may never be successfully executed because it always leaves the IB in an inconsistent state. For instance, let us consider an additional operation *removeUser* that deletes the specified user as long as he or she has not bidden for any product:

```
Op:    removeUser(u: User)
Pre:    u.bid->isEmpty()
Post:    User.allInstances()->excludes(u)
```

This operation is applicable, since its precondition can be satisfied, but the postcondition removes a user, which is necessarily the owner of some product according to the cardinality constraint 1..\* of *offered-prod*. Since this operation does not remove the products offered by the user, the resulting state of the IB will always violate the cardinality constraint 1 of *owner* for all products offered by *u*. This means that the execution of this operation will always be rejected because it is impossible to satisfy its postcondition and the integrity constraints at the same time.

To check executability, an additional rule has to be added to the translation of the schema to record the execution of the operation. In this case, if *executed\_O* is satisfiable, then *O* is executable:

$$executed\_O \leftarrow o(P_1, \dots, P_n, t) \wedge pre(T-1)$$

## 4.2 Is It the Right Conceptual Schema?

Once the internal correctness of the schema is ensured by the previous tests, the designer will need to check its external correctness, i.e. whether it satisfies the requirements of the domain.

Our approach allows testing whether a certain desirable state that the designer may envisage is acceptable or not according to the current schema. The designer may define such a state either by means of a set of instances that classes and associations should contain at least; or by a derived predicate that defines it declaratively. Once a test is executed, the designer should compare the obtained results to those expected

according to the requirements and apply modifications to the conceptual schema if necessary.

For instance, an interesting question could be “*May a user place a bid on a product he is offering?*.” To test this situation, the designer should define the rule:

$$\text{bidderAndOwner} \leftarrow \text{bid}(B, \text{Prod}, \text{Usr}, \text{Amt}, T) \wedge \text{offeredBy}(\text{Prod}, \text{Usr}, T)$$

In this case, *bidderAndOwner* is satisfiable, as shown by the sample instantiation:

$$\{ \text{registerUser}(\text{john}, \text{john@upc.edu}, 111, \text{prod1}, \text{pen}, 10, 1), \text{placeBid}(\text{john}, \text{prod1}, 15, 3) \}$$

This result might indicate that the conceptual schema should restrict a user to place a bid on the products he owns either by defining an additional constraint in the structural schema or by strengthening the precondition of the operation *placeBid*.

As mentioned above, the designer may also specify additional tests by giving several instances of classes and associations and check whether there is at least a state that contains them (probably in addition to other instances). As an example, the designer could wonder whether a certain user, e.g. *joan*, may place two bids for the same product, e.g. *book1*. This situation may be tested by determining whether there is a state that contains the instances  $\{\text{bid}(1, \text{book1}, \text{joan}, 9, 1), \text{bid}(5, \text{book1}, \text{joan}, 25, 7)\}$ , obtaining a negative answer in this case since, according to the semantics of associations, two instances of *Bid* cannot be defined by the same instances of *User* and *Product*. Since a user should be able to rebid for a product, this schema is not correct and should be modified by changing the definition of *Bid*.

By studying the results of the previous tests, and with his knowledge about the requirements of the system to be built, the designer will be able to decide if the schema is correct, and perform the required changes if not.

Checking the external correctness of a schema can also be partially automated by generating additional tests that check other kinds of properties. For instance, given a recursive association *Assoc*, it may be interesting to check whether an instance of the related class can be associated to itself. If the predicate *assocHasCycles* is satisfiable, then a constraint to guarantee that the association is acyclic or irreflexive, as it is usual in practice, may be missing:

$$\text{assocHasCycles} \leftarrow \text{assoc}(X, X, T)$$

## 5 Implementing Our Approach within an Existing Method

We have studied the feasibility of our approach by using an existing reasoning procedure, the CQC-Method [8], to perform the tests. To do this, we have extended a Prolog implementation of this method to incorporate a correct treatment of the time component of our atoms. We have executed this new implementation on our example to perform all validation tests that we have explained throughout the paper. We have also needed to implement the translation of the UML/OCL schema into logic.

The CQC Method is a semidecision procedure for finite satisfiability and unsatisfiability. This means that it always terminates if there is a finite example or if the tested property does not hold. However, it may not terminate in the presence of solutions with infinite elements. Termination may be assured by defining the maximum number of elements that the example may contain.

Roughly, the CQC Method is aimed at constructing a state that fulfills a goal and satisfies all the constraints in the schema. The goal to attain is formulated depending on the specific reasoning task to perform. In this way, the method requires two main inputs besides the conceptual schema definition itself. The *goal to attain*, which must be achieved on the state that the method will try to construct; and the set of *constraints to enforce*, which must not be violated by the constructed state.

Then, to check if a certain property holds in a schema, this property has to be expressed in terms of an initial goal to attain ( $G_0$ ) and the set of integrity constraints to enforce ( $F_0$ ), and then ask the CQC Method to attempt to construct a sample IB to prove that the initial goal  $G_0$  is satisfied without violating any integrity constraint in  $F_0$ .

This means that, to perform our validation tests, we need to provide the CQC Method with the formalization of our schema, i.e. the derived predicates that represent classes and associations, the set of constraints of the schema as  $F_0$  and the derived predicate formalizing the validation test to perform as  $G_0$ .

### 5.1 Variable Instantiation Patterns

The CQC Method performs its constraint-satisfiability checking tests by trying to build a sample state satisfying a certain condition. For the sake of efficiency the method tests only those variable instantiations that are relevant, without losing completeness. The method uses different *Variable Instantiation Patterns (VIPs)* for this purpose according to the syntactic properties of the schema considered in each test. The key point is that the VIPs guarantee that if the variables in the goal are instantiated using the constants they provide and the method does not find any solution, then no solution exists.

The VIP in which we are interested is the *discrete order VIP*. In this case, the set of constants is ordered and each distinct variable is bound to a constant according to either a former or a new location in the total linear order of constants maintained. The value of new variables is not always *static* (i.e. a specific numeric value), it can be a relative position within the linear ordering of constants. These are called *virtual constants*. For instance, in the ordering of constants  $\{1, d, 6\}$ ,  $d$  is a virtual constant such that  $1 < d < 6$ . Then, its possible absolute values are 2 to 5. It may happen that the goal succeeds or fails without the need for further instantiations, and in this case  $d$  will never be bound to a concrete value.

To correctly instantiate the variables representing occurrence times that we have introduced in our translation of the conceptual schema, it has been necessary to add a *temporal VIP*. This new VIP has some similarities with the *discrete order VIP*, since they both deal with discrete values, order comparisons and negation, but it extends it to be able to bind a constant, either virtual or static, with its immediate successor. This is needed because our derivation rules require that preconditions hold exactly in the time immediately previous to the postcondition, not at any time before the postcondition. Then, we use a separate set of constants, with its own ordering, to deal with variables representing event times and we instantiate them with our *temporal VIP*.

For instance, assume we are attempting to derive an *Unregistered* user which must hold at time  $d$ , being  $d$  a virtual constant and  $\{1, d, 5\}$  our set of temporal constants. According to the precondition of *unregisterUser*, the user must be registered at  $d-1$ . Thus, since  $1 < d < 5$ , the time variable of the corresponding instance of *Registered* must

be instantiated either with 1 or with a virtual constant  $f, f=d-1$ . So, the relevant sets of constants are  $\{[1, d], 5\}$  and  $\{1, [f, d], 5\}$ , where constants between brackets are tied so that no new constant can be ever placed between them.

The *temporal VIP* is formalized as follows. A variable instantiation step performs a transition from  $(T \in KT_i)$  to  $(\theta \in KT_{i+1})$  that instantiates the temporal variable  $T$  according to one of the VIP-rules, where  $\theta$  is a ground substitution of  $T$  and  $KT_i$  is the set of temporal constants. Let  $d_i$  denote virtual constants,  $c_i$  denote static constants and  $k_i$  denote either static or virtual constants, and let  $G_c$  be the current goal. The *temporal VIP* consists of the VIP-rules of the *discrete order VIP*, extended by the following rules, that apply when instantiating a temporal constant  $T$  such that  $T = k_i - 1, k_i \in KT_i$ :

- Tmp1.  $\theta = T / c_{prev}$  and  $KT_{i+1} = KT_i$ , where  $c_{prev} = c_{suc} - 1, \{c_{suc}, c_{prev}\} \subseteq KT_i, \{T = c_{suc} - 1\} \in G_c$
- Tmp2.  $\theta = T / k$  and  $KT_{i+1} = KT_i$ , where  $\{k, k_{suc}\} \subseteq KT_i, \{T = k_{suc} - 1\} \in G_c$ , there is no constant  $k_{prev}$  such that  $k < k_{prev} < k_{suc}$  and  $k$  is tied to  $k_{suc}$  in  $KT_{i+1}$ .
- Tmp3.  $\theta = T / c_{new}$  and  $KT_{i+1} = KT_i \cup \{c_{new}\}$ , where  $c_{new} = c_{suc} - 1, c_{new} \notin KT_i, c_{suc} \in KT_i, \{T = c_{suc} - 1\} \in G_c$ , there is no  $d_{prev}$  tied to  $c_{suc}$  in  $KT_i$  and there is no  $c_{prev} \in KT_i$  such that  $c_{prev} < c_{suc}$  and  $\{|d_i \mid d_i \in KT_i \text{ and } c_{prev} < d_i < c_{suc}\} < |c_{suc} - c_{prev}| - 1$ .
- Tmp4.  $\theta = T / d_{new}$  and  $KT_{i+1} = KT_i \cup \{d_{new}\}$ , where  $d_{new} \notin KT_i, d_{suc} \in KT_i, \{T = d_{suc} - 1\} \in G_c$ , there is no  $d_{prev}$  tied to  $d_{suc}$  in  $KT_i$ ,  $d_{new}$  is tied to  $d_{suc}$  in  $KT_{i+1}$  and there are no  $c_i, c_j \in KT_i$  such that  $c_i < d_{suc} < c_j$ , there is no  $c_m$  with  $c_i < c_m < c_j$  and  $\{|d_i \mid d_i \in KT_i \text{ and } c_i < d_i < c_j\} < |c_j - c_i| - 1$ .

## 6 Related Work

In this section we focus on those approaches that deal with UML schemas with a behavioral part. Thus, we leave out from our comparison to previous work those approaches that only deal with the structural schema [10, 12, 16], since satisfiability of the structural part does not necessarily imply that the whole conceptual schema is also satisfiable; as well as the first proposals to deal with behavior, in the context of deductive conceptual schemas [4, 5].

Due to its relevance, and despite not dealing with UML schemas, we believe it is worth including the Alloy language and analyzer [14] in this comparison. Alloy provides interesting validation capabilities for expressive schemas by searching for examples of the tests specified by the designer. The preconditions and postconditions of the operations can be checked manually, before and after each execution.

One of the first approaches to check satisfiability of UML schemas with operations is [6]. General constraints are handled, but they must be expressed in Z instead of OCL, which is the language recommended by the UML to formalize constraints and operations. Besides checking satisfiability of the structural schema, operations to insert, delete and update the instances of each class or association are automatically generated.

An approach to reason on UML/OCL schemas is HOL-OCL [2]. The method uses a theorem prover to determine some properties, such as equivalence of two integrity constraints, or applicability and executability of operations.



Another interesting tool to validate UML/OCL conceptual schemas is USE [9], which allows to test if a given instantiation is accepted by the schema taking into account the OCL constraints. Preconditions and postconditions can also be validated, but the execution of the operation has to be simulated manually.

Recently, and also for UML/OCL schemas, [3] reports a set of properties regarding the correctness of operations such as applicability or executability.

All the previous approaches have an important common drawback. None of them takes into account the definition of operations when determining whether a state is accepted or not by the schema. This means that a state may be reported as valid when, in fact, it is impossible to construct using the operations defined. This also damages the results obtained when testing the applicability of operations, since the state that satisfies a precondition may not be obtained by means of the operations defined. In fact, all these approaches would give an incorrect answer to 5 out of the 6 properties tested in this paper.

One of the approaches that does not share this drawback is [7], which combines state and event-based descriptions of a system to enable the automatic verification of dynamic properties regarding the system behavior. It may handle UML class diagrams but assumes that the system behavior is specified in the B and CSP languages, instead of OCL, and it is mainly aimed at testing properties related to the correct sequencing of the operations specified in the conceptual schema.

The other approach that takes the operations into account when determining whether a state is accepted or not by the schema belongs to the Rodin project. It combines UML-B [19] and ProB [13], the former to represent the schema and translate it into the B language, and the latter to validate it by animation. However, UML-B only accepts a subset of the UML that is suitable for translation into B, and constraints and operations must be directly expressed in B by the designer. Additionally, ProB requires that the search space is made finite by enumerating the values to be used in the animation. Since the fact that a property does not hold for those values does not mean that it can never hold, completeness is not guaranteed.

Finally, all of the approaches are able to check either the internal correctness [3, 6, 13] or the external correctness of the schema [2, 9, 14], but not both as we do.

## 7 Conclusions and Further Work

We have proposed a new approach to validate a UML conceptual schema, with textual OCL constraints and operations. To our knowledge, ours is the first approach that validates jointly the structural and behavioral parts of a UML/OCL schema.

Our approach allows determining automatically whether the conceptual schema is correctly defined, through tests about the accomplishment of desirable properties; and provides also a help to the designer to check that the schema defined is the right conceptual schema in the sense that it correctly specifies the requirements.

This is achieved by translating the UML conceptual schema, including its behavioral part, into a logic representation which incorporates the effect of operations in terms of the instances of classes and associations that are created or deleted. In this way, we ensure that the only changes allowed are those defined in the behavioral schema. With this logic representation, we can formalize each validation test in terms of checking the satisfiability of a derived predicate. Then, any satisfiability checking method able to deal with negation of derived predicates can be used to validate the schema.

We have also shown the feasibility of our approach by using and extending an implementation of an existing reasoning procedure, called CQC-Method [8], and applying it to our running example.

There are some interesting directions for further work, like applying the decidability results of our previous work [18] to schemas with a behavioral part, extending our approach to validate conceptual schemas with derived UML information or investigating the applicability of this approach to large conceptual schemas.

*Acknowledgements.* This work has been partly supported by Ministerio de Ciencia y Tecnología under TIN2008-00444/TIN, Grupo Consolidado, and TIN2008-03863.

## References

1. Adrion, W.R., Branstad, M.A., Cherniavsky, J.C.: Validation, Verification and Testing of Computer Software. *ACM Comput. Surv.* 14(2), 159–192 (1982)
2. Brucker, A.D., Wolff, B.: *The HOL-OCL Book*. Swiss Federal Institute of Technology (ETH), 525 (2006)
3. Cabot, J., Clarisó, R., Riera, D.: Verifying UML/OCL Operation Contracts. In: Leuschel, M., Wehrheim, H. (eds.) *IFM 2009*. LNCS, vol. 5423, pp. 40–55. Springer, Heidelberg (2009)
4. Costal, D., Teniente, E., Urpí, T., Farré, C.: Handling Conceptual Model Validation by Planning. In: Constantopoulos, P., Vassiliou, Y., Mylopoulos, J. (eds.) *CAiSE 1996*. LNCS, vol. 1080, pp. 255–271. Springer, Heidelberg (1996)
5. Díaz, O., Paton, N.W., Iturrioz, J.: Formalizing and Validating Behavioral Models through the Event Calculus. *Information Systems* 23(3/4), 179–196 (1998)
6. Dupuy, S., Ledru, Y., Chabre-Peccoud, M.: An Overview of RoZ: A Tool for Integrating UML and Z Specifications. In: Wangler, B., Bergman, L.D. (eds.) *CAiSE 2000*. LNCS, vol. 1789, pp. 417–430. Springer, Heidelberg (2000)
7. Evans, N., Treharne, H., Laleau, R., Frappier, M.: Applying CSP || B to Information Systems. *Software and System Modeling* 7, 85–102 (2008)
8. Farré, C., Teniente, E., Urpí, T.: Checking Query Containment with the CQC Method. *Data and Knowledge Engineering* 53(2), 163–223 (2005)
9. Gogolla, M., Büttner, F., Richters, M.: USE: A UML-based Specification Environment for Validating UML and OCL. *Science of Computer Programming* 69(1-3), 27–34 (2007)
10. Hartmann, S.: Coping with Inconsistent Constraint Specifications. In: Kunii, H.S., Jajodia, S., Sølvberg, A. (eds.) *ER 2001*. LNCS, vol. 2224, pp. 241–255. Springer, Heidelberg (2001)
11. Larman, C.: *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development*, 3rd edn. Prentice Hall PTR, Englewood Cliffs (2004)
12. Lenzerini, M., Nobili, P.: On the Satisfiability of Dependency Constraints in Entity-Relationship Schemata. In: *Proc. 13th International Conference on Very Large Databases - VLDB 1987*, pp. 147–154 (1987)
13. Leuschel, M., Butler, M.: *ProB: An Automated Analysis Toolset for the B Method*. *Software Tools for Technology Transfer* (2008) DOI: s10009-007-0063-9
14. MIT. The Alloy Analyzer. MIT Software Design Group, <http://alloy.mit.edu>

15. Olivé, A.: Conceptual Schema-Centric Development: A Grand Challenge for Information Systems Research. In: Pastor, Ó., Falcão e Cunha, J. (eds.) CAiSE 2005. LNCS, vol. 3520, pp. 1–15. Springer, Heidelberg (2005)
16. Queralt, A., Teniente, E.: Reasoning on UML Class Diagrams with OCL Constraints. In: Embley, D.W., Olivé, A., Ram, S. (eds.) ER 2006. LNCS, vol. 4215, pp. 497–512. Springer, Heidelberg (2006)
17. Queralt, A., Teniente, E.: Specifying the Semantics of Operation Contracts in Conceptual Modeling. In: Spaccapietra, S. (ed.) Journal on Data Semantics VII. LNCS, vol. 4244, pp. 33–56. Springer, Heidelberg (2006)
18. Queralt, A., Teniente, E.: Decidable Reasoning in UML Schemas with Constraints. In: Bellahsene, Z., Léonard, M. (eds.) CAiSE 2008. LNCS, vol. 5074, pp. 281–295. Springer, Heidelberg (2008)
19. Snook, C., Butler, M.: UML-B: Formal Modeling and Design Aided by UML ACM Trans. on Soft. Engineering and Methodology 15(1), 92–122 (2006)
20. Utting, M., Legeard, B.: Practical Model-Based Testing. Morgan Kaufmann, San Francisco (2006)
21. Warmer, J., Kleppe, A.: The Object Constraint Language: Getting Your Models Ready for MDA, 2nd edn. Addison-Wesley Professional, Reading (2003)

# Towards the Industrialization of Data Migration: Concepts and Patterns for Standard Software Implementation Projects

Klaus Haller

COMIT AG, Pflanzschulstr. 7, CH-8004 Zürich, Switzerland  
klaus.haller@comit.ch

**Abstract.** When a bank replaces its core-banking information system, the bank must migrate data like accounts from the old into the new system. Migrating data is necessary but not a catalyst for new business opportunities. The consequence is cost pressure to be addressed by an efficient software development process together with an industrialization of the development. Industrialization requires defining the deliverables. Therefore, our data migration architecture extends the ETL process by migration objectives to be reached in each step. Industrialization also means standardizing the implementation, e.g. with patterns. We present data migration patterns describing the typical transformations found in the data migration application domain. Finally, testing is an important issue because test-case based testing cannot guarantee that not a single customer gets lost. Reconciliation can do so by checking whether each object in the old and new system has a counterpart in the other system.

**Keywords:** Data Migration, Patterns, ETL, Standard Software, ERP.

## 1 Motivation

In the last years, many Swiss banks replaced old, less-flexible and expensive-to-maintain core-banking systems with new ones like Avaloq or Finnova [1]. Replacing the systems requires not only setting up and customizing the new system but also migrating data like customers or accounts into the new system.<sup>1</sup> Data migration is necessary, but only performed once. Furthermore, it is not an enabler for business processes. Strict budgets are the consequence requiring an industrialization of the data migration development. Industrialization is often narrowed down to having a software development process like CMMI [2]. But industrialization also has a technical aspect, i.e. standardizing artifacts to be developed and concepts for constructing them.

---

<sup>1</sup> To prevent confusion we want to point out the difference between database migration and data migration. In data migration, application-related artefacts like triggers must not be migrated. Instead, data might have to be transformed to fit into the new system's database schema. In contrast, migrating a *database* (e.g. from Microsoft SQL server to Oracle) demands not only to copy the data but also all application-related artefacts (triggers, constraints etc.).

Typical examples are patterns [3] or the three-tier-architecture for data-intensive applications [4]. Our vision is providing concepts allowing the industrialization of our application domain *data migration*.

The concepts in this paper reflect our experience with data migration as part of several Avaloq core-banking system implementation projects in Swiss banks. They are an outcome of COMIT's industrialization efforts for core-banking system implementation projects, the LeanStream initiative [5]. Up to now, our work on data migration concentrated on a migration infrastructure architecture [6] and on project management issues [7]. This paper complements our previous work by focusing on the industrialization of the development by equipping practitioners with blueprints for their implementations. The core concept is the ETL (extract, transform, load) process known from data warehousing [8]. We assign data migration specific objectives to each step. If different developers develop code (or use ETL tools) for the steps, the results might look completely different. However, we observed only very few different underlying patterns, which we compile in this paper. Developers should look at a migration problem in a project and remember immediately the right pattern(s) he or she has to adopt and use. By focusing on the data structure before and after the usage of the pattern, we characterize the patterns in a universally applicable way.

We organize the rest of our paper as following: Section 2 discusses related work followed by a presentation of our general migration architecture (Section 3). Section 4 explains the most important implementation techniques (language constructs and tools). Sections 5-7 describe the patterns for the different data migration steps, i.e. extract the data from the old system, transform it with respect to the new schema, and, finally, load the data into the new system. Detecting failures, especially lost data, is a major issue for data migration. We devote Section 8 to this challenge.

## 2 Related Work

Data migration is a practitioners' topic, i.e. only very few publications exist. However, the pioneering work comes from academia: the *butterfly approach* [9]. The butterfly approach provides a phase model with five steps: (i) analysis, (ii) development of the data mappings, (iii) building up a sample data set in the target system, (iv) migration of the system components to the target system without any data, and (v) step-by-step data migration. The key architectural element is a temporary message queue. Messages in the message queue are either "waiting" or "processed". The message queue has two operating modes. In the first mode, it processes messages in the state "waiting" respectively newly arriving messages from the old system for the migration. Processed messages switch their state to "processed" but do not leave the message queue yet. In the second mode, "processed" messages are released into the target system whereas newly arriving messages are stored in the queue in the mode "waiting". The message queue switches regularly between the two modes. The assumption is that the number of messages in the queue gets smaller and smaller.

The butterfly architecture suits well for batch processing with one or just a few message queues. The more interactive the processing and the more systems are coupled, the more difficult and expensive the butterfly approach becomes.

The most extensive work on *data migration project management* is a book written by Morris [10]. The author focuses mainly on project managers having to set-up and organize a migration project for the first time. He also provides a high-level overview about the most important technical issues. An Endava white paper has the same focus [11]. Shorter articles (e.g. [12, 13]) target the same audience, but only discuss very basic problems and pitfalls.

*Tool descriptions* focus on how to use (often target system specific) tools for data migration. Examples are a book about SAP's data migration tools [14] or explanations of how to migrate to a new product version or how to get away from a competitor's product [15, 16]. Furthermore, Carreira and Galhardas describe a specific language designed especially for data migration purposes [17].

Broadening the view, also data warehousing respectively the ETL process mentioned above are related [8]. Schema mapping [18], an area where tremendous research took place, is related due to the goal of mapping schemata with their attributes. Though we would have been more than happy to use (semi-)automatic techniques to reduce costs, the focus is too different. Companies buy new software to get additional functionality. This requires transforming and enriching the data being migrated from the old to the new system, which is difficult to automate.

### 3 Generic Migration Architecture

Each data migration project has to decide whether to follow the source-push or the target-pull paradigm [11]. **Target-pull** means migrating only data necessary for the target system, whereas **source-push** migrates *all* data of the old system into the new one. On a first glance, the latter one sounds appealing. One cannot forget any data because everything is in the target system. However, it is highly uneconomical. Usually only around 10% of the attributes have to be migrated. Users and application management understand and know these attributes well. Other attributes require more effort and an in-depth analysis of the application. Many attributes are there for pure technical reasons or to store intermediate results. So if the application is not very simple, it is too expensive to analyze every attribute. Target-pull, i.e. hunting (and migrating) attributes of the old system needed by the new one, is the option of choice. But certainly, it makes sense to have a copy of the old system in a read-only database before the old system is switched off.

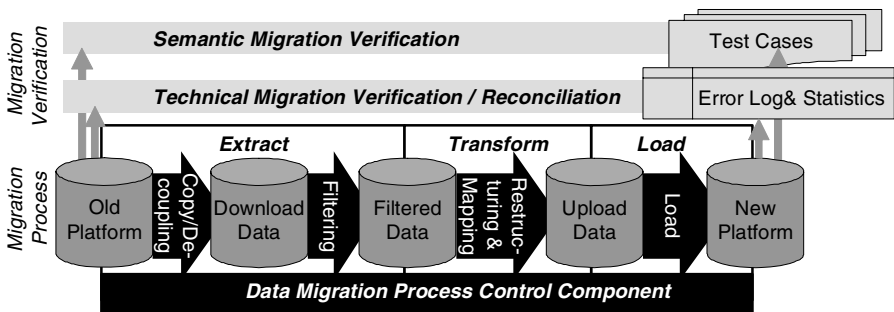


Fig. 1. Generic Migration Architecture

The centre of our data migration architecture in Figure 1 is an ETL process. The *extract step* aims firstly on decoupling the data migration project and the old system. It copies the data to a different server named download area. Thus, the project cannot affect the daily business of the bank. Secondly, the extract step identifies the data to be migrated and filters everything else. Customers, for example, not doing any business with the bank for years might not be of interest. The filtering is the one and only point where the decision is made whether an object is migrated or not. If an object passes this border, it must reach the target systems.

Next, the filtered data runs through the *transformation step*. If the database schemata of the old and new system differ, the data must be restructured in the transformation step. If the domain values are different (e.g. one system stores the currency as “USD” whereas the second one used the full name “US DOLLAR”), the transformation step accomplishes the mapping. Finally, after the transformation step, the data is *loaded* into the new system.

The ETL process illustrates the migration process of a single object type like customers or accounts. However, core-banking systems have many object types, i.e. there is one ETL process for each object type. Furthermore, often additional tasks must be performed like calculating statistics and histograms for the optimizer. Many processes and additional tasks, possibly to be performed in a certain order, make it too risky for a pure manual orchestration. Therefore, a *data migration process control component* stores the execution order. It does not necessarily perform the complete migration without manual intervention, but might automate certain steps.

After all data has been migrated, one verifies that the migrated data is correct and complete based on two complementary migration verification techniques. *Test cases* are selected sample objects, e.g. addresses and accounts of five typical and five very important customers. A tester checks manually all attributes like owner, IBAN, interest rate etc. in the old and in the new system for these objects (semantically migration verification). The second technique, *reconciliation*, is an automatic technical verification. It checks whether *all* objects have been migrated, but not whether all attributes are correct. It allows detecting e.g. five missing accounts out of ten millions.

## 4 Programming Paradigms

The generic data migration architecture assigns goals to each step. Fulfilling the goals can be done with different programming paradigms. The choice of the programming paradigm and the tool respectively programming language depends on each project’s situation. The source and target systems’ databases are relevant, knowledge in the project, availability of tools, the project duration etc. Therefore, we focus on the ideas of the three main paradigms (row-based implementation, set-oriented implementation, ETL tool). We discuss their different advantages on a qualitative level and provide concrete examples using PL/SQL respectively Oracle Warehouse Builder.

In the examples, the source schema has one table for natural persons (OLD\_PERSONS) and one for juristic persons (OLD\_COMPANIES). The target

schema consists of one table CUSTOMERS. All rows of the natural and juristic persons tables are migrated if they represent customers (TYPE='Customer'). To illustrate transformations, natural persons having a Ph.D. (attribute PHD='+') get a "Dr." prefix to their names in the target table.

```

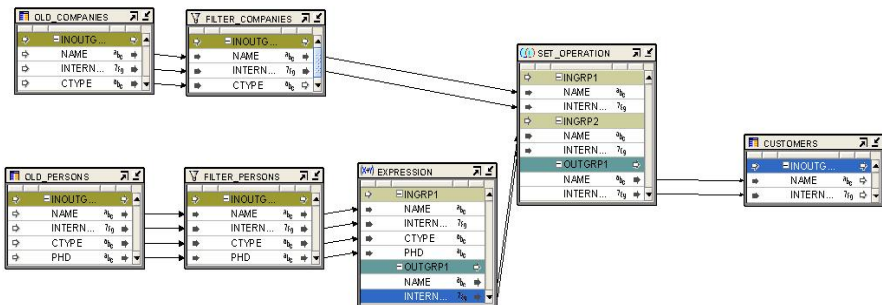
1 DECLARE
2   CURSOR c_old_persons
3     IS SELECT name, internal_id, phd FROM old_persons WHERE ctype = 'CUSTOMER';
4   CURSOR c_old_companies
5     IS SELECT name, internal_id FROM old_companies WHERE ctype = 'CUSTOMER';
6   newname varchar2(100);
7 BEGIN
8   FOR cp IN c_old_persons
9   LOOP
10    IF (cp.phd = '+') THEN newname := 'Dr. ' || cp.NAME;
11    ELSE newname:=cp.name; END IF;
12    INSERT INTO customers (NAME, internal_id)
13      VALUES (newname, cp.internal_id);
14  END LOOP;
15  FOR cc IN c_old_companies
16  LOOP
17    INSERT INTO customers (NAME, internal_id)
18      VALUES (cc.NAME, cc.internal_id);
19  END LOOP;
20 COMMIT;
21 END;
```

### Example 1: Row-oriented Implementation Paradigm using PL/SQL

```

1 BEGIN
2   INSERT INTO customers(name, internal_id)
3     SELECT CASE WHEN phd='+' THEN 'Dr. '||name ELSE name END, internal_id
4     FROM old_persons WHERE ctype='CUSTOMER'
5   UNION
6     SELECT name, internal_id
7     FROM old_companies WHERE ctype='CUSTOMER';
8 COMMIT;
9 END;
```

### Example 2: Set-oriented Implementation Paradigm using PL/SQL



### Example 3: ETL-Tool-based Implementation using Oracle Warehouse Builder



## 4.1 The Row-Oriented Implementation Paradigm

The row-oriented implementation paradigm specifies the migration in an imperative way using e.g. Java with JDBC or PL/SQL scripts. There is one script for each object type (addresses, persons etc.). The key idea is that each script has a loop enclosing the mapping (Example 1, lines 8-13 and 14-17). Inside the loop, a cursor accesses and processes one row per iteration of the loop. The actual implementation of the migration deals only with one source table row per iteration (lines 10-12 and line 16). The advantage of a row-based implementation is, firstly, that everyone familiar with imperative programming understands the concept. Secondly, the concept hides data-parallelism. Using cursors means that one does not have to consider the whole table at once but only the recent row. The ultimate benefit of having programming tasks with a lower complexity is that staffing the project becomes easier. However, some optimization possibilities are lost which a database optimizer might have otherwise.

## 4.2 The Set-Oriented Implementation Paradigm

The set-oriented implementation paradigm also uses an imperative programming model. Instead of hiding data-parallelism using cursors, it uses set-oriented SQL-statements like SELECT. In our example, all relevant data of table OLD\_PERSONS respectively of table OLD\_COMPANIES is selected and transformed in one statement (Example 2, lines 3-4 and 6-7). Complex transformations are more difficult to be implemented in a single step. Then, it might be wise implementing the transformation in more steps and storing intermediate results in temporary tables. The highly compact implementation allows database optimizers to execute the code more efficiently than row-oriented implementations. The disadvantage is the higher level of abstraction requiring programmers feeling comfortable with data set-oriented thinking.

## 4.3 ETL-Tool-Based Implementation Technique

ETL-tools often provide a visual programming language for defining data-flows. Data flows have one or more data sources. In Example 3 on the left, the tables OLD\_COMPANIES and OLD\_PERSONS are such sources. A data sink collects the result (table CUSTOMERS). Between the data sources and the data sink(s) operators can be placed for manipulating the data. In our example, the operators FILTER\_PERSONS and FILTER\_COMANIES filter objects not being customers. The operator EXPRESSION changes the names of persons depending on the PHD attribute. The companies thread and the customers thread come together at the UNION set operation implementing a UNION.

ETL tools provide a visual way of programming. The systems are very robust. However, complex migrations might require large data-flows which might be difficult to understand. The main obstacle against ETL tools is that learning them might take a long time if the knowledge does not already exist.

## 5 Extract Step Patterns

The extract step fulfils two goals. It downloads data from the productive system in the first sub-step. Afterwards, in the second sub-step, it filters the data. The purpose of the **download sub-step** is decoupling. A data migration project works on a separate project server, such that the project does not interfere with the daily operations on the old system. The decoupling requires downloading a copy of all possibly needed tables (e.g. customers, customer accounts, and banks in the example in Figure 2, but not account bookings). Generally, it is not wise to be too selective with the tables to be downloaded. Firstly, downloading a missing table later might only be allowed during dedicated service windows of the old system. Secondly, the data might become inconsistent. Assume one downloads all customer accounts on May 2<sup>nd</sup> and account balances on May 15<sup>th</sup>. If accounts are opened or closed between May 2<sup>nd</sup> and May 15<sup>th</sup>, there are suddenly accounts without account balances or vice versa. Such inconsistencies result in errors or testing problems. Thus, one missing table might require downloading a large number of tables to ensure consistency.

The **filtering sub-step** is conceptually important for the migration verification (see Section 8). Objects passing the filter must make it into the target system. This rule must be enforced strictly. Otherwise, it becomes difficult to decide whether an object was excluded on purpose or was forgotten. Such questions are especially difficult to answer if they arise weeks after the implementation.

The filtering allows excluding superfluous objects, e.g. customers who died ten years ago. Filtering also excludes objects to be migrated manually. Manual migration is more economical if there are only a few objects of a certain type (usually less than 100-1000). Also, some objects might already be in the target system. A core-banking system might e.g. already store all stock exchanges in a table. However, it is important to understand that no transformation takes place in the filtering step. But certainly, the object model in the old and new system might differ resulting in splitting object sets. Figure 2 illustrates the aspect. The old system stores all banks in one table. The new one distinguishes between the roles of banks. There are banks the bank does business with directly, e.g. because the bank has nostro accounts with them. Other banks are for reference purposes only, e.g. banks in Central Asia to which money could be sent by SWIFT. Thus, the banks of the old system are divided during the filtering into “business partner” banks and “reference data” banks.

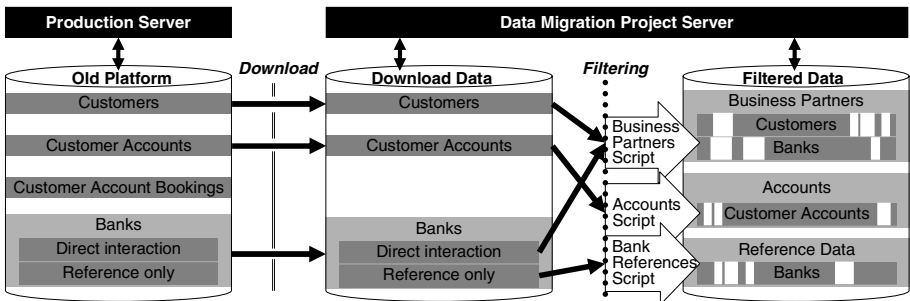


Fig. 2. Extract Step

The download sub-step copies tables and therefore does not need special patterns for the implementation. The filtering is more complex. In the following, we present the three main filtering patterns mostly needed in projects. Our presentation relies on the sample schema in Figure 3. The schema stores all accounts of the old system in table T\_ACCOUNT. Customer accounts (in contrast to internal accounts) refer to their owner in table T\_CUSTOMER. The third table T\_INTERESTRATE stores the accounts' interest rates and how they changed over time. With the help of this sample schema, the three patterns are introduced quickly.

- **Attribute value based filtering.** The pattern decides whether a row is selected for each row independently of other rows or tables. One example is choosing all accounts from table T\_ACCOUNT with PRODUCT='SAVINGS ACCOUNT' (Result Set 1 in Figure 3).
- **Selection table based filtering.** The pattern decides whether a row is selected based on information in a second table. The pattern determines a key for each of the rows in table one. In the second table, the pattern looks for rows having a matching key. Depending on the identified rows in the second table, the row of the first table passes the filter. An example is choosing all rows from table T\_ACCOUNT having an owner with BRANCH\_ID=10 stored in table T\_CUSTOMER or not having a customer as an owner (Result Set 2). It could be implemented e.g. based on a join condition like:

```
SELECT a.*
FROM T_ACCOUNT a LEFT OUTER JOIN T_CUSTOMER c
ON a.OWNER_ID=c.CUSTOMER_ID
WHERE c.CUSTOMER_ID IS NULL OR c.BRANCH_ID=10
```

- **Aggregation based filtering.** Aggregation functions in SQL determine a value based on information in several rows, e.g. the highest value or the average. Similarly, aggregation based filtering decides whether a row is filtered based not only on the information of the row itself. It considers also other rows of the same table. A good example is choosing the latest interest rate for each account, i.e. the currently valid one (Result Set 3). Table T\_INTERESTRATE stores two interest rates for account 1000765208, one valid from 6.8.2007, the other one from 1.1.2007. For choosing the actual valid interest rate, one must look at *all* interest rates of account 1000765208. Thus, the filter chooses the interest rate valid from 6.8.2007 and to skip the one from 1.1.2007.

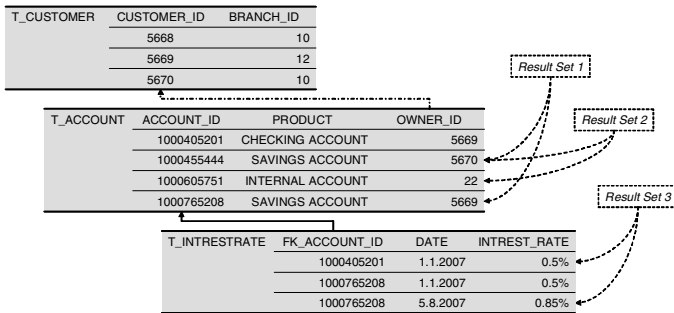


Fig. 3. Sample Tables for Filtering Patterns

## 6 Transformation Implementation Patterns

### 6.1 Pattern Group Mapping

Mapping is similar to working with a dictionary. You look for the value of the old system (e.g. “Germany” or “United States”). In the same row, but in a different column, you find the value for the new system (“DEU” and “USA”). The pattern group *mapping* provides two implementation patterns (Figure 4):

- Mapping table.** A mapping table stores a value of the old system (“Germany”) and the corresponding value of the new system (“DEU”) in each row. Mapping tables are specified best as Excel sheets by experts with business knowledge. Then, the excel file is loaded into the database system. However, if the table is very small, it might make sense to use a CASE statement instead of a mapping table. Figure 4 provides a simple example based on the mapping table MAP\_COUNTRY. Simple means that there is one attribute used for choosing the row (NAME), and one attribute is delivered back (ISO\_CODE\_3). The new value is determined by a join statement.

```
SELECT c.CUSTOMER_ID, m.ISO_CODE_3
FROM CUS_OLD c LEFT OUTER JOIN MAP_COUNTRY m
ON c.nationality=m.name
```

- Mapping function.** Some mappings are more complex and too difficult to be specified using a mapping table. A good example is temperature conversion from degree Celsius to Fahrenheit, where e.g.  $3.21^{\circ}\text{C}=(3.21*9/5+32)\text{F}$  or if the assets under management and the margin of a customer are mapped to a classification of the customer. In this situation, a mapping function is needed as represented by *f* in Figure 4. A corresponding mapping SQL statement would be:

```
SELECT CUSTOMER_ID, f (CONTRIBUTION_MARGIN, ASSETS)
FROM CUS_OLD
```

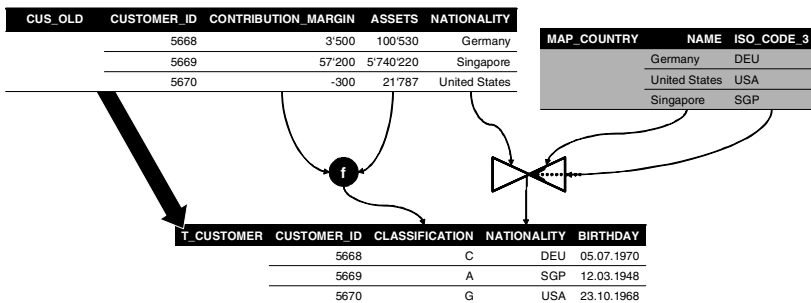


Fig. 4. Sample Tables *Mapping* Pattern Group

### 6.2 Pattern Group Restructuring

The old and the new system usually have different object models resulting in different database schemata. Restructuring patterns help transform existing data to fit into the database schema of the target system. The three main patterns are:

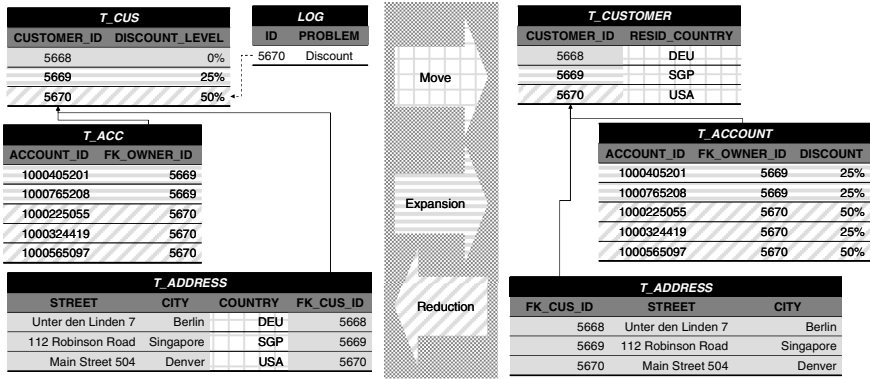


Fig. 5. Restructuring Pattern Group Examples

- Simple Attribute Move.** The old and the new data schema store the same attribute in different tables. For example, the left schema in Figure 5 models the country of residence as address information and stores it in the address table *T\_ADDRESS*. The right schema emphasizes the tax perspective. It stores the country of residence as customer information in table *T\_CUSTOMER*. The simple attribute move pattern “moves” the information during the transformation step to a different table, i.e. from *T\_ADDRESS* to *T\_CUSTOMER*.
- Expansion.** Both schemata have a semantically similar attribute but modeled on a different level of granularity. In Figure 5, the left schema provides a discount level for each customer (*T\_CUS*). Each customer can get one discount level for all her bank charges, e.g. 0%, 50%, or even 100%. The right schema allows a more sophisticated fee modeling. Each account can have a different discount level. If the data from the left schema is migrated into the right one, the discount level information is expanded by copying the value into *each* account.
- Reduction.** It is the opposite of expansion. The old system allows a more granular modeling than the new one. Thus, the migration is an approximation of the old data. Information gets lost. If the migration in Figure 5 takes place from right to left, customer 5670’s accounts have different discount levels in the right schema. But the customer can have only one in the left schema. Depending on the circumstances, it might be mandatory to log such loss of information (table *LOG*), because customers must be informed about changes. Thus, it is important not only to have a log table but also to have a process in place how to deal with such problems.

## 7 Load Patterns

When the data is transformed, the migration team loads the data into the target system. The implementation of the loading is the decision of the vendor. The vendor can choose from three patterns (Figure 6): the direct approach, the simple API one, and the workflow API one. The *direct approach* provides no API. All data is inserted

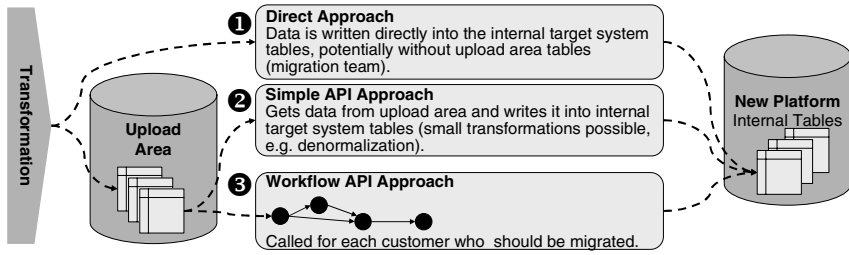


Fig. 6. Data Loading Approaches

directly into the internal tables. The *simple API approach* provides an upload area with API tables. The migration team inserts data into the API tables and invokes an API load procedure, which writes the data into the internal tables of the system. The *workflow-based API approach* also comes with an upload area with API tables. However, the API invokes the workflow separately for each object in the API table. The workflow is the same used e.g. by the GUI if new objects are entered manually.

Before we compare the patterns, we want to point out the vendor's dilemma. Customers are not willing to pay a premium for superior support for loading data during the migration. But the vendor risks his reputation if the project fails due to data migration problems. For a better understanding of the patterns, we compare them considering the dimensions in Table 1. *Error detection* considers whether the migration team gets feedback for each object whether it was migrated successfully. If not, a reason shall be given. *Conformity* compares data migrated by the data migration team and data manually entered via a GUI. The migrated data shall comply with the same requirements as manually entered data. *Vendor effort* rates the investment the vendor has to make. The *migration team training* addresses how much training the implementation team needs to work efficiently. The *migration team implementation effort* reflects the effort a trained team has for the implementation.

If the new system implements the **direct approach**, the core-banking system does not detect any migration errors. At most, some triggers or constraints might prevent the most severe mistakes. The conformity of migrated and manually entered data might be weak if the migration team does not implement exactly the same checks

Table 1. Load Step Strategies

	Technical Dimensions		Vendor Costs	Costs Migration Team	
	Error Detection	Conformity		Team Training	Implementation
<i>Direct Approach</i>	No support	Not guaranteed, difficult to achieve	No effort	High, in-depth understanding of internal tables needed	High(est) due to the need to implement all checks
<i>Simple API</i>	Handled by API	Some conformity, but not guaranteed	High, if conformity desired	Low, requires good vendor documentation	Overhead for guaranteeing conformity
<i>Workflow-based API</i>	Handled by API	Guaranteed	Initial costs for framework, rest low	Low, requires good vendor documentation	No overhead for extra checks

applied to manually entered data respectively if not all restrictions are enforced by the database schema. However, the direct approach is the cheapest one for the vendor. It costs nothing. On the other side, the migration team needs much training (respectively learns by trial and error during the project, which is quite expensive). Also, the implementation is costly because the migration team has to implement many consistency checks.

The **simple API approach** means that the API copies the data from the API tables (possibly with some changes) into the internal tables of the system. The API can check for failures or non-compliances to the data model. The vendor either has to implement the same checks again he already uses for the GUI (high costs) or there is only a limited conformity guarantee. The benefit of an API for the migration team is that the team needs less training due to a clearly defined API. The migration team's implementation effort is restricted to missing conformity checks; therefore, it loses time by running into mistakes. The extra effort of the migration team depends how much the customization can change, because the changes require adopting the conformity checks or might be a source for mistakes.

If a vendor implements the **workflow-based API approach**, the workflows used for checking the consistency and inserting new data into the system are identical for data inserted via a GUI or data being migrated. The API uses existing workflows and returns already defined error messages. The vendor has initial costs for a framework. Afterwards, he has nearly no additional efforts no matter how many object types have to be considered. Also the migration team benefits from this approach. It has low training costs and gets data consistency guaranteed by the API.

## 8 Technical Migration Verification

A standard method for checking the functional correctness of applications is using *test cases*. In data migration projects, this means checking whether *all attributes of selected objects* are correctly migrated. Additionally, customers like banks or external auditors want to be sure that no data is lost. Every single customer, account, etc. must be checked. This is a task to be automated and usually termed technical migration verification or *reconciliation*. The focus is on checking relevant, *selected attributes of all objects*. Result is a reconciliation sheet. It is produced after each test data migration as a feedback for the migration team and after the final data migration. In the latter case, it enables the bank to decide whether the new system can replace the old one.

Based on our experience, we suggest that a *reconciliation sheet* consists of two parts, statistics and migration errors. Statistics provide an aggregated high level view, e.g. how many objects (accounts or also the sum of assets under management) exist in both systems and which only in one of the two. The migration errors part lists the "needles in a haystack". If three out of three million accounts are missing or have different attribute values, the error section lists keys identifying the wrong or missing objects together with the failure information ("object is missing" or "attribute BALANCE has different values").

We distinguish three patterns for deriving a reconciliation sheet (Table 2). The **top-down pattern** is the simplest one. It is used only if a project has not (or has yet not

**Table 2.** Reconciliation Strategies

Pattern	Idea	Identification	Usage Restrictions	Recon Sheet Section
<i>Top-down</i>	Counting, potentially grouped by	Object type level, based on table or characteristic attributes	No restrictions.	Statistics
<i>Bottom-up equivalence</i>	Comparing row by row	Key candidate are equivalent in both cases, attributes to be compared belong in both systems to the same object type	Key candidate attributes or attributes to be compared must not be involved in a restructuring	Comparison, results can be aggregated to statistics
<i>Bottom-up fingerprint</i>	Comparing aggregated row information	Aggregated rows have a common key attribute (but not a key for each row)	Useful in case of restructuring	Comparison, results can be aggregated to statistics

had) enough time to implement a sophisticated reconciliation. At least, it informs whether a large number of objects are missing. It creates the statistics section only by counting the objects in the old and new system, possibly considering subtypes. The accounts' reconciliation sheet in Figure 7 illustrates the aspect with the statistics for the accounts with subtype information (customer, nostro, etc.).

For identifying which single account got lost or has a wrong type, the comparison section of the reconciliation sheet must be created. The **bottom-up equivalence pattern** is one possibility. It creates a unique key for each row in the tables of the old and new system and looks whether there is a corresponding one in the other table. The attribute ACCOUNT\_ID, for example, is a good key for the tables T\_ACCOUNT\_OLD and T\_ACCOUNT\_NEW. The pattern can be implemented as following:

```
SELECT o.ACCOUNT_ID, n.ACCOUNT_ID,
       CASE WHEN o.ACCOUNT_ID is not null AND n.ACCOUNT_ID is not null THEN 'OK'
            ELSE 'FAILED'
       END as match
FROM t_account_old o FULL OUTER JOIN t_account_new n ON n.account_id=o.account_id
```

It is mandatory to use a full outer join to identify rows in the old *or* the new system missing a counterpart in the other one. In our example, account 1000765208 exists only in the old system and 5000565097 is a phantom only existing in the new system. If the keys match, selected attributes are checked for correctness. The equivalence comparison includes *relevant and comparable* attributes. The only comparable attributes for accounts is the account type, which fails for account 9500000084. To get this result, we extend the matching join-condition as following:

```
SELECT o.ACCOUNT_ID, n.ACCOUNT_ID,
       CASE WHEN o.ACCOUNT_ID is not null AND n.ACCOUNT_ID is not null THEN 'OK'
            ELSE 'FAILED'
       END as match,
       CASE WHEN o.ACCOUNT_TYPE = n.ACCOUNT_TYPE THEN 'OK'
            ELSE 'ERROR'
       END as equal
FROM t_account_old o FULL OUTER JOIN t_account_new n ON
n.account_id=o.account_id
```

In practice, the bottom-up equivalence pattern works well for 90-95% of the situations. The interest rates example in Figure 7 is one where it fails. A good reconciliation would use a pair <ACCOUNT\_ID, LIMIT> as a key and compare the



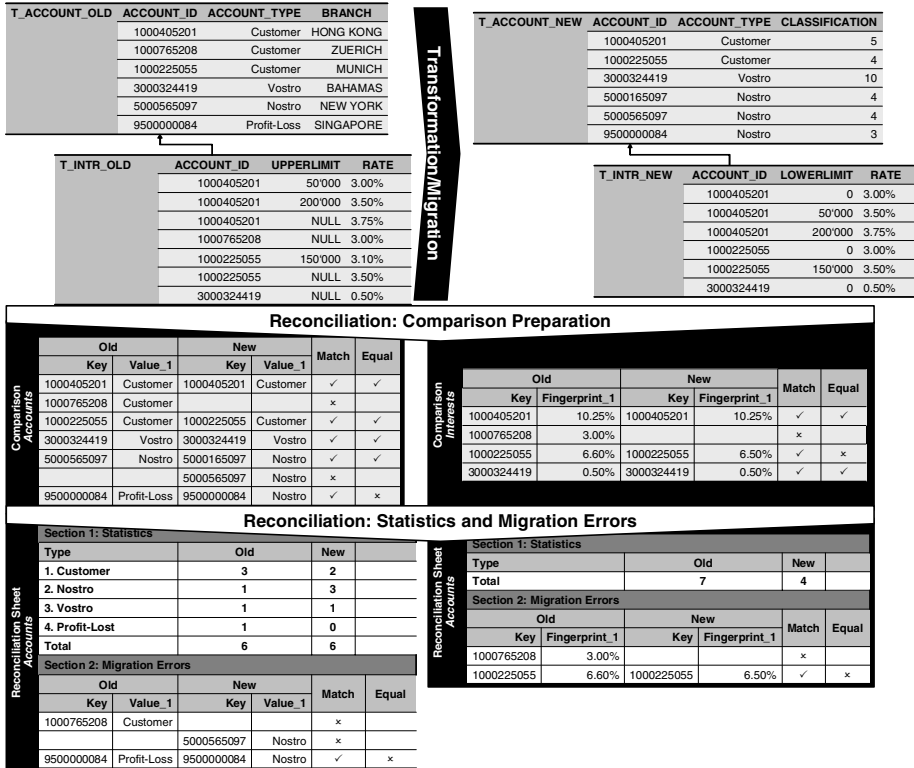


Fig. 1. Reconciliation Sheet Generation Process

interest rate as the most relevant attribute. This is not possible because the limit in the old system is an upper limit whereas the one in the new system a lower one. The worst thing one can do in such a situation is to copy the code used to transform the upper to a lower limit. If this is done, the reconciliation looks always perfect. The data migration step and the reconciliation have the same input and process the data in the same way. Thus, the results are the same no matter how wrong the transformation itself is. In such situations, the **bottom-up fingerprint pattern** helps. A fingerprint (a kind of hash value) is constructed using all relevant attributes, but it is not necessarily a semantically sensible piece of information.

We discuss now three sample fingerprints for the situation above. The simplest fingerprint is to look whether interests exist in the old and the new system for exactly the same accounts. Better would be option two, i.e. to look whether accounts with interests have always the same number of interests in both systems (like account 1000405201 having three ones). The third approach, which we used in our projects, is to sum up the interest rates for each account. It is semantically nonsense to calculate  $3.00\%+3.50\%+3.75\%=10.25\%$  for account 1000405201. However, the rate information is included and the number of limits also influences the result. Our fingerprint does not guarantee that the limit - rate relationship is correct. However, such systematic failures should be detected by the manual migration verification,

which should include test cases with accounts with complex interest rate information. This fingerprint could be implemented as following:

```
SELECT o.ACCOUNT_ID, n.ACCOUNT_ID,
CASE WHEN o.ACCOUNT_ID is not null AND n.ACCOUNT_ID is not null THEN 'OK'
      ELSE 'FAILED'
END as match,
CASE WHEN o.FINGERPRINT= n. FINGERPRINT THEN 'OK'
      ELSE 'ERROR'
END as equal
FROM      (SELECT ACCOUNT_ID, SUM(RATE) as fingerprint
           FROM T_INTR_OLD GROUP BY ACCOUNT_ID) o
FULL OUTER JOIN (SELECT ACCOUNT_ID, SUM(RATE) as fingerprint
                FROM T_INTR_NEW GROUP BY ACCOUNT_ID) n
ON n.account_id=o.account_id
```

Data migration is often overlooked, but it is crucial for success when replacing an old by a new system. Our data migration architecture relies on an ETL process based data migration architecture. It defines clear objectives for the different ETL steps. Decoupling and filtering takes place in the extract step, mapping and restructuring data to fit into the schema of the target system follow in the transformation step. Getting the data into the target system with a feedback about the success takes place during the load step. Furthermore, we present the typical patterns developers find in their project such that they can rely on *simple* building blocks for their implementation. By also addressing the reconciliation challenge which is unique for data migration projects, all our concepts together form a blueprint for the implementation tasks in data migration projects. Companies can easily incorporate our work into their development processes. Thereby, they improve the standardization and industrialization of data migration in their projects.

## References

1. Gabriel, C.: Plattform-Wechsel: Parforce-Übung mit weitreichenden Folgen, Schweizer Bank, Zürich (June 2007)
2. CMMI for Development, Version 1.2, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA (2006)
3. Fowler, M.: Patterns of Enterprise Application Architecture. Addison-Wesley Longman, Boston (2002)
4. Fraternali, P.: Tools and Approaches for Developing Data-Intensive Web Applications: A Survey. ACM Computing Surveys 31(3) (September 1999)
5. LeanStream® – COMIT Implementationsmethodik, V. 3.0, Comit AG, Zürich (2007)
6. Haller, K.: Datenmigration bei Standardsoftware-Einführungsprojekten. Datenbank-Spektrum 8(25), 39–46 (2008)
7. Haller, K.: Data Migration Project Management and Standard. In: 5th Conference on Data Warehousing (DW 2008), St. Gallen, Switzerland. Lecture Notes in Informatics (2008)
8. Chaudhuri, S., Dayal, U.: An Overview of Data Warehousing and OLAP Technology. SIGMOD Record 26(1), NY (1997)
9. Wu, B., Lawless, D., Bisbal, J., et al.: The Butterfly Methodology: A Gateway-free Approach for Migrating Legacy Information Systems. In: ICECCS, Como, Italy (1997)
10. Morris, J.: Practical Data Migration. British Computer Society, Swindon (2006)
11. Data Migration – The Endava Approach, White Paper, London (2006)
12. Burry, C., Mancusi, D.: How to plan for data migration, Computerworld, May 21 (2004)

13. Hudicka, J.R.: An Overview of Data Migration Methodology. Select Magazine, Independent Oracle Users Group, Chicago, IL (April 1998)
14. Willinger, J., Gradl, J.: Data Migration in SAP R/3. Galileo Press, Boston (2004)
15. Anavi-Chaput, V., et al.: Planning for a Migration of PeopleSoft 7.5 from Oracle/UNIX to DB2 for OS/390 (Red Book), IBM, Poughkeepsie, NY (2000)
16. Manek: Microsoft CRM Data Migration Framework (White Paper), Microsoft Corporation (2003)
17. Crreira, P., Galhardas, H.: Efficient development of data migration transformations. In: SIGMOD, Paris, France (2004)
18. Rahm, E., Bernstein, P.: A survey of approaches to automatic schema matching. VLDB Journal 10, 334–350 (2001)

# Defining and Using Schematic Correspondences for Automatically Generating Schema Mappings

Lu Mao, Khalid Belhajjame, Norman W. Paton, and Alvaro A.A. Fernandes

School of Computer Science  
University of Manchester  
Oxford Road, Manchester, UK  
{lmao, khalidb, norm, alvaro}@cs.man.ac.uk

**Abstract.** Mapping specification has been recognised as a critical bottleneck to the large scale deployment of data integration systems. A mapping is a description using which data structured under one schema are transformed into data structured under a different schema, and is central to data integration and data exchange systems. In this paper, we argue that the classical approach of correspondence identification followed by (manual) mapping generation can be simplified through the removal of the second step by judicious refinement of the correspondences captured. As a step in this direction, we present in this paper a model for schematic correspondences that builds on and extends the classification proposed by Kim *et al.* to cater for the automatic derivation of mappings, and present an algorithm that shows how correspondences specified in the model proposed can be used for deriving schema mappings. The approach is illustrated using a case study from integration in proteomics.

**Keywords:** Schematic correspondences, schema mappings, mapping generation.

## 1 Introduction

Data integration has for the last two decades been the subject of active investigations within the database and the artificial intelligence communities [2, 4]. This is testified partly by the number of research papers, projects and prototypes that tackle data integration related issues [9]. The aim is to provide users with integrated access to data sets that reside in multiple sources and are stored using heterogeneous representations [13]. A data integration system can play a central role in multiple applications, e.g., it can be used for cross-querying of data stored in databases that belong to independent companies, or to promote collaboration in large scientific projects by providing investigators with a means for querying and combining results produced by multiple research labs [21]. The components at the heart of a data integration system are: the schemas of the sources, the data sets to be integrated, an integration schema over which users pose queries, and mappings that specify how data structured under the schemas of the sources can be transformed and combined into data structured according to the integration schema [6].

Despite the advances made, data integration seems to have had a limited impact in practice: existing data integration systems are mostly research prototypes. The limited

adoption of this technology is partly due to its cost ineffectiveness [8]. In particular, the specification of the mappings between the schemas of the sources and the integration schema has proved to be both time and resource consuming, and was recently recognised as a critical bottleneck to the large scale deployment of data integration systems [8, 15].

Mapping specification is generally a two-phase process. In the first phase, the correspondences specifying how the elements of the integration schema relate to the elements of the sources' schemas are specified. Correspondences are primarily used to identify the elements of the integration schema and the source schemas that are semantically equivalent, i.e., represent information that belongs to the same domain concept. For example, the correspondence  $\langle S_{int}.Staff, S_1.Employee \rangle$  specifies that the relation *Staff* in the integration schema  $S_{int}$  is semantically equivalent to the relation *Employee* of the source schema  $S_1$ . Different types of correspondences may be drawn between two schemas [5, 19]. For example, correspondences can be used to relate one element of a given schema to one element of another schema, e.g.,  $\langle S_{int}.Staff, S_1.Employee \rangle$ , or to relate multiple elements of one schema to one element of another schema, e.g., the correspondence  $\langle S_{int}.Staff, \{S_1.Employee, S_1.Department\} \rangle$  states that the relation *Staff* in the integration schema is semantically equivalent to some combination of the relations *Employee* and *Department* of the source schema  $S_1$ . There are several models for drawing schematic correspondences in the literature [7, 12, 14, 17], of which the model proposed by Kim *et al.* [12] is perhaps the most comprehensive. In the second phase of mapping specification, the views that implement the mappings necessary for rewriting the queries issued against the integration schema into queries over the schemas of the sources are specified. Examples of techniques that can be used for specifying the mappings based on identified correspondences were informally described by Kim *et al.* [11].

The specification of schema mappings is largely a manual activity. In this paper, we show that the views implementing the mappings can be automatically derived if the correspondences defined by Kim *et al.* [12] are refined in a number of carefully targeted ways. In essence, it is the argument of this paper that the classical approach to correspondence identification followed by (manual) mapping generation can be made more cost-effective through the removal of the second step based on judicious refinement of the correspondences captured. Take for example the correspondence  $\langle S_{int}.Staff, \{S_1.Employee, S_1.Department\} \rangle$ , presented earlier. This correspondence does not specify the correspondences between the attributes of *Staff* and those of *Employee* and *Department*. Neither does it specify how the tuples of the *Staff* and *Department* relations should be combined. Because of this, an algorithm would not be able to derive the view that can be used for populating the relation *Staff* using the tuples in *Employee* and *Department*. The main contributions of this paper are therefore:

- A model for schematic correspondences that builds on and extends the classification proposed by Kim *et al.* to enable the automatic derivation of mappings.
- An algorithm for automatically generating the views that implement the mappings between two schemas based on these more expressive schematic correspondences.

Accordingly, the paper is organised as follows. We begin by presenting the model for specifying schematic correspondences in Section 2. We go on to show how the

correspondences specified in this model can be used for automatically generating the views that implement the mappings between two schemas in Section 3. We show how schematic correspondences can be used for deriving mappings between proteomics data sources in Section 4. We analyse existing proposals for modelling schematic correspondences and automating mapping specification in Section 5. Finally, we close the paper by underlying our main contributions in Section 6.

## 2 Schematic Correspondences

Schematic correspondences are used to associate elements of one schema (referred to in what follows as the *source schema*) to elements that are semantically equivalent in another schema (referred to as the *target schema*). For the purpose of this paper we assume that source and target schemas are specified using the relational model [3]. Therefore, the elements connected by schematic correspondences are relations and/or attributes. As mentioned earlier, the model of schematic correspondences presented in this paper is built on the classification proposed by Kim *et al.* [12]. This classification identifies a *wide range* of correspondences that may occur between two schemas. Nevertheless, the correspondences as defined by Kim *et al.* do not convey sufficient information for deriving the views that express the mappings between schemas. Consider, for example, a correspondence that connects the relations *Employee* and *Address* in the source schema to the relation *Staff* in the target schema. To be able to specify the view for populating the relation *Staff* using the tuples in both *Employee* and *Address*, information specifying how the tuples in *Employee* and *Address* are to be combined is needed. In the following, we augment the model of correspondences proposed by Kim *et al.* to support the automatic generation of mappings. In doing so, we distinguish between two kinds of correspondences: *relation correspondences* and *attribute correspondences*.

### 2.1 Relation Correspondences

This family of correspondences associates relations from the source schema with relations that are semantically equivalent in the target schema. We distinguish between four kinds of relation correspondences depending on the number of relations used to represent a given concept in the source and target schemas, namely *one-to-one relation correspondence*, *many-to-many relation correspondence*, *many-to-one relation correspondence* and *one-to-many relation correspondence*.

*One-To-One Relation Correspondence.* This kind of correspondence associates one relation in the source schema with one relation that is semantically equivalent to it in the target schema. For example, a one-to-one relation correspondence can be used to associate the *Company* relation in the source schema with the *Corporation* relation in the target schema: the tuples of both relations represent business organisations. We define a one-to-one relation correspondence by the tuple:

$$\langle r_s, r_t, p_s, p_t, AC \rangle$$

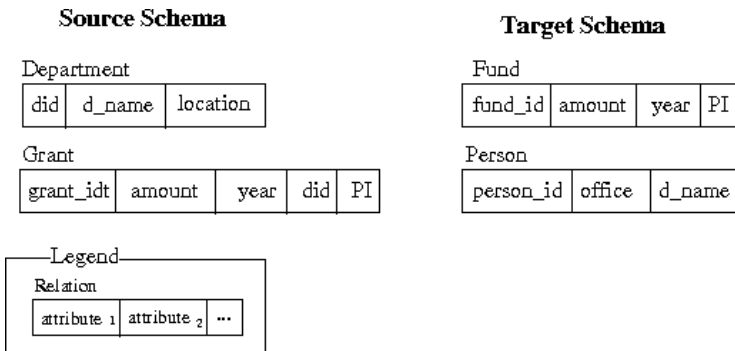
where  $r_s$  is a relation in the source schema and  $r_t$  is a relation in the target schema.  $AC$  is a set of attribute correspondences that specifies the relationships between the

attributes of the relation  $r_s$  and those of  $r_t$ . Attribute correspondences will be presented in Section 2.2.  $p_s$  and  $p_t$  are two selection predicates specifying which instances of  $r_t$  can be used to populate  $r_s$  and which instances of  $r_s$  can be used to populate  $r_t$ , respectively. Note that  $r_s$  and  $r_t$  can have different names and different structures, i.e., one or more attributes in  $r_s$  may not have any corresponding attributes in  $r_t$ , or vice versa.

*Many-To-Many Relation Correspondence.* This kind of correspondence associates multiple relations from the source schema  $R_s$  with multiple relations in the target schema  $R_t$ . It specifies that the combination of the relations in  $R_s$  is semantically equivalent to the combination of the relations in  $R_t$ . We define a many-to-many relation correspondence by the tuple:

$$\langle R_s, R_t, type_s, type_t, JP_s, JP_t, HPP_s, HPP_t, AC \rangle$$

where,  $R_s$  is a set of relations in the source schema and  $R_t$  is a set of relations in the target schema. To specify the mapping from the relations in  $R_s$  to the relations in  $R_t$ , we need information specifying how the relations in  $R_s$  and the relations in  $R_t$  are to be combined, respectively. This is specified using the fields  $type_s$  and  $type_t$  which take either the value *Vertical partitioning* or the value *Horizontal partitioning*.  $type_s$  specifies how the relations in  $R_s$  are to be combined, whereas  $type_t$  specifies how the relations in  $R_t$  are to be combine. Vertical partitioning indicates that the relations in  $R_s$  (resp. in  $R_t$ ) should be combined by joining them using  $JP_s$  (resp.  $JP_t$ ), a conjunction of attribute comparison predicates. Figure 1 illustrates an example of vertical partitioning in which the combination of the relations *Department* and *Grant* in the source schema corresponds to the combination of the relations *Fund* and *Person* in the target schema. The relations *Department* and *Grant* are combined using the join predicate '*Department.did = Grant.did*', and the relations *Fund* and *Person* are combined using the join predicate '*Fund.PI = Person.person\_id*'. Note that the attribute *PI* represents the identifier of the principal investigator. As we shall see later in Section 3, we use outer join predicate instead of natural join to avoid any loss of information.



**Fig. 1.** Example of many-to-many vertical partitioning

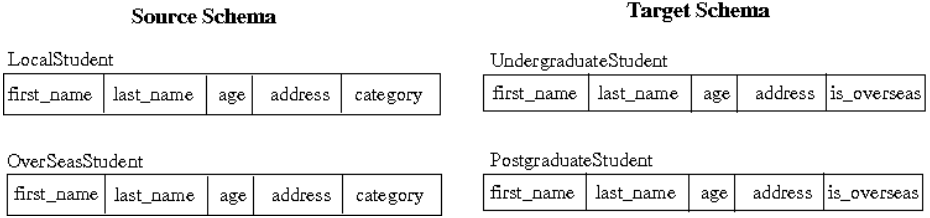
A horizontal partitioning indicates that the relations in  $R_s \cup R_t$  represent the same domain concept. As an example, consider the many-to-many relation correspondence that connects the relations *Undergraduate* and *Postgraduate* from the source schema to the relations *LocalStudent* and *OverseasStudent* in the target schema (see Figure 2). The relations *Undergraduate*, *Postgraduate*, *LocalStudent* and *OverseasStudent* capture information about the same semantic domain, namely *Student*. Differently from vertical partitioning in which the relations are combined using the join relational operator, in the case of horizontal partitioning the relations in the source schema (resp. in the target schema) have compatible structures and are combined using the union relational operator. Note that this assumes that semantically corresponding attributes in the relations to be combined are given the same name. It is worth noting that horizontal partitioning may be defined between relations that do not have identical structures. Consider the example of horizontal partitioning presented earlier in Figure 2. The relation *PostgraduateStudent* may have, in addition to the attributes of *UndergraduateStudent*, the attributes *supervisor* and *advisor*. Applying a union operator in this case is not possible since the two relations have different structures. In this case, it is possible to use the outer union operator to combine the relations *UndergraduateStudent* and *PostgraduateStudent* [3]. This operator deals with the problem of inconsistency in structure between two relations  $r_1$  and  $r_2$  by applying the union operator over the relations  $r'_1$  and  $r'_2$ , where  $r'_1$  (resp.  $r'_2$ ) is obtained by adding the attributes in  $r_2$  (resp.  $r_1$ ) that are missing in  $r_1$  (resp.  $r_2$ ) and padding them with null values. For example, the outer union of the *UndergraduateStudent* and *PostgraduateStudent* relations is a relation that has the same structure as *PostgraduateStudent* and in which the tuples representing undergraduate students have null values for the attributes *supervisor* and *advisor*.

To specify the view for populating a relation in the target schema we need information specifying which tuples obtained by the union of the relations in the source schema are to be considered. For example, we know that not all local and overseas students are undergraduate students, and, therefore, we cannot populate the relation *UndergraduateStudent* using all the tuples obtained by the union of the relations *LocalStudent* and *OverseasStudent*.  $HPP_s$  and  $HPP_t$  are used for this purpose. These are two sets, the elements of which are predicates: an element of  $HPP_t$  is a conjunction of predicates that specifies which tuples should be used for populating a given relation in the target schema; similarly an element of  $HPP_s$  is a conjunction of predicates that specifies which tuples should be used for populating a given relation in the source schema. We assume in the following the existence of the function  $getHPPredicate(corr, r)$  that returns the conjunction in  $corr.HPP_t$  that is associated with the relation  $r$ . For example,  $getHPPredicate(corr_{student}.HPP_s, UndergraduateStudent) = '(category = "undergrad")'$ , where  $corr_{student}$  is the correspondence that relates the relations *LocalStudent* and *OverseasStudent* to the relations *Undergraduate* and *Postgraduate*.

As in one-to-one relation correspondences,  $AC$  is a set specifying the correspondences between the attributes of the relations in  $R_s$  and those of the relations in  $R_t$ .

*Many-To-One and One-To-Many Relation Correspondences.* These can be seen as specific cases of many-to-many relation correspondences. A many-to-one relation correspondence associates multiple relations from the source schema  $R_s$  to one relation  $r_t$  in the target schema. It specifies that the relation obtained by combining the relations





**Fig. 2.** Example of many-to-many horizontal partitioning

in  $R_s$  is semantically equivalent to  $r_s$ . As for many-to-many relation correspondences, the relations in  $R_s$  can be combined by joining them in the case of vertical partitioning, or by using the union operator in the case of horizontal partitioning. A many-to-one relation correspondence can be defined by the tuple:

$$\langle R_s, r_t, type_s, JP_s, HPP_s, p_t AC \rangle$$

where,  $R_s$  is a set of relations in the source schema and  $r_t$  is a relation in the target schema.  $type_s$ ,  $JP_s$ ,  $HPP_s$  and  $AC$  have the same meaning as in the definition of many-to-many relation correspondence.  $p_t$  is a selection predicate specifying which tuples can be used to populate  $r_t$ .

A one-to-many relation correspondence associates one relation from the source schema to multiple relations in the target schema. It is defined by the tuple:

$$\langle r_s, R_t, type_t, JP_t, HPP_t, AC \rangle$$

where,  $r_s$  is a relation in the source schema and  $R_t$  is a set of relations in the target schema.  $type_t$ ,  $JP_t$ ,  $HPP_t$  and  $AC$  have the same meaning as in the definition of many-to-many relation correspondence.  $p_s$  is a selection predicate specifying which tuples can be used to populate  $r_s$ .

## 2.2 Attribute Correspondences

This family of correspondences associates attributes of relations that belong to the source schema with attributes that are semantically equivalent in the target schema. As for relation correspondences, attribute correspondences can be classified into four kinds depending on the number of attributes involved from the source and target schemas.

*One-To-One Attribute Correspondence.* This correspondence associates one attribute in the source schema with one attribute that is semantically equivalent in the target schema. We define a one-to-one attribute correspondence by the tuple:

$$\langle r_s.att_s, r_t.att_t, f_{s \rightarrow t}, f_{t \rightarrow s} \rangle$$

where  $att_s$  and  $att_t$  are attributes of the relations  $r_s$  and  $r_t$ , respectively.  $f_{s \rightarrow t}$  (resp.  $f_{t \rightarrow s}$ ) is a function specifying how a value of  $att_t$  (resp.  $att_s$ ) can be computed using a value of  $att_s$  (resp.  $att_t$ ).

*Many-To-Many Attribute Correspondence.* This correspondence associates multiple attributes  $Att_s$  in the source schema with multiple attributes  $Att_t$  in the target schema. It specifies that the combination of the attributes in  $Att_s$  is semantically equivalent to the combination of the attributes in  $Att_t$ . To specify the mapping between  $Att_s$  and  $Att_t$  we need information specifying how the values of the attributes in  $Att_s$  are to be combined to obtain the values of the attributes in  $Att_t$ , and vice versa. We therefore define many-to-many attribute correspondence by the tuple:

$$\langle Att_s, Att_t, f_{s \rightarrow t}, f_{t \rightarrow s} \rangle$$

where  $Att_s$  is a set of attributes in the source schema and  $Att_t$  is a set of attributes in the target schema. The correspondence also specifies two functions:  $f_{s \rightarrow t}$  for computing the values of the attributes in  $Att_t$  given the values of the attributes in  $Att_s$ , and  $f_{t \rightarrow s}$  for computing the values of the attributes in  $Att_s$  given the values of the attributes in  $Att_t$ .

Notice that the attributes in  $Att_s$  do not have to belong to the same relation, i.e. attributes from different relations can be involved. The same observation applies to the attributes in  $Att_t$ .

*Many-to-one and one-to-many attribute correspondences.* These are specific cases of many-to-many attribute correspondences. A many-to-one attribute correspondence associates multiple attributes in the source schema with one attribute in the target schema, whereas a one-to-many attribute correspondence associates one attribute in the source schema with multiple attributes in the target schema. Both correspondences can be specified using the same tuple we used for defining many-to-many attribute correspondence, i.e.,  $\langle Att_s, Att_t, f_{s \rightarrow t}, f_{t \rightarrow s} \rangle$ :  $Att_s$  (resp.  $Att_t$ ) is a singleton in the case of one-to-many (resp. many-to-one) attribute correspondence. Below is an example of a many-to-one attribute correspondence that associates the attributes *fare* and *tax* in the source schema to the attribute *full\_price* in the target schema.

$$\langle \{fare, tax\}, \{full\_price\}, price(), getPriceDetails() \rangle$$

*price()* is a function with the signature given below. Given a fare  $f$ , and a tax  $t$ ,  $price(f,t)$  returns the full price, i.e., the sum of  $f$  and  $t$ . *getPriceDetails()* is a function with the signature given below. Given a full price  $p$ ,  $getPriceDetails(p)$  returns a pair  $\langle f, t \rangle$ , where  $f$  is the fare and  $t$  is the tax. In the following, we use  $getPriceDetails(p) \downarrow_{fare}$  and  $getPriceDetails(p) \downarrow_{tax}$  to denote  $f$  and  $t$ , respectively.

$$\begin{aligned} price &: domain(fare) \times domain(tax) \rightarrow domain(full\_price) \\ getPriceDetails &: domain(full\_price) \rightarrow domain(fare) \times domain(tax) \end{aligned}$$

Schematic correspondences as defined above can be used, as we shall see in the next section, for automatically generating the mappings between source and target schemas. To avoid (or minimise) human intervention during this process, every relation in the

target schema should be involved in zero or one schematic correspondence, and every attribute of a relation in the target schema should be involved in zero or one schematic correspondence. Failure to meet the above criteria will lead to conflicts between correspondences, the resolution of which requires the intervention of a human user. The resolution of conflicts between correspondences is outside the scope of this paper. Therefore, we assume henceforth that the above conditions are satisfied by the schematic correspondences between source and target schemas.

### 3 Mapping Generation

Given the model for schematic correspondences just presented, we can construct an algorithm (shown in Figure 3) for automatically generating the mappings. The outcome of a mapping generation process are the views specifying how data described using a source schema could be used to populate a target schema. Before presenting the details of this algorithm, we outline the notation that we will use:

- The algorithm for mapping generation outputs a set of views. A view is a named query. For the purpose of this work, we define a view,  $v$ , by the pair  $\langle name_v, query_v \rangle$ , where  $name_v$  is a string that identifies the view, and  $query_v$  is a query specified using the relational algebra.
- Given a relation  $r$ ,  $r.name$  denotes the name of  $r$  and  $r.attributes$  the list of attributes of  $r$ .
- Given a relation correspondence  $corr$ ,  $corr.source$  denotes the set of relations in the source schema that are involved in  $corr$ , and  $corr.target$  denotes the set of relations in the target schema that are involved in  $corr$ .
- Given a schematic correspondence  $corr$ , be it a relation or an attribute correspondence,  $getCorrCardinality(corr)$  is a function that specifies whether  $corr$  is a one-to-one, many-to-one, one-to-many or many-to-many correspondence.
- Given a set of attributes correspondences  $AC$ ,  $getAttCorrespondence(AC, r.att)$  returns the correspondence in which the attribute  $r.att$  is involved if such a correspondence exists, and returns null otherwise.
- Given a set of relations  $R = \{r_1, \dots, r_m\}$ , and a conjunction of predicates  $JP$ , we use  $\square \bowtie \square_{JP} R$  to denote the full outer join of the relations in  $R$  using  $JP$ , and  $\bigcup R$  to denote the union of the relations in  $R$  using the outer union operator.
- $append(l, e)$  is an operation that adds the element  $e$  to the end of the list  $l$ .

The mapping generation algorithm takes as input a target schema together with a set of relation correspondences. It iterates over the relations present in the target schema, retrieving for each of them the associated correspondences (Figure 3 line 3). If a relation has no correspondence (Figure 3 line 5) then no view is generated for that relation. If, on the other hand, it is associated with more than one correspondence, then this conflict in correspondences is reported to the user (Figure 3 line 7). Otherwise, the algorithm derives a view for the relation in question using the subroutine presented in Figure 4. This subroutine operates in two steps. Firstly, it specifies the relations taking part in the view and the way they are to be combined. If the correspondence  $corr$  used as input is a one-to-one or one-to-many relation correspondence then the view is assigned

Algorithm GenerateMappings

inputs  $Sc_t$ : a target schema.

$Corr$ : a set of schematic relation correspondences.

outputs  $View$ : a set of views.

begin

1  $View = \emptyset$

2 **FOR EACH**  $r_t \in Sc_t.relations$  **DO**;

3  $Corr_{r_t} = getCorrespondences(Corr, r_t)$

4 **IF**  $|Corr_{r_t}| = 0$  **THEN**

5  $Signal('No view is generated for the relation ' + r_t)$

6 **IF**  $|Corr_{r_t}| > 1$  **THEN**

7  $Signal('The relation ' + r_t + ' has more than one correspondence.')$

8 **IF**  $|Corr_{r_t}| = 1$  **THEN**

9  $Let\ corr\ be\ the\ only\ element\ of\ Corr_{r_t}$

10  $v = DeriveView(r_t, corr)$

11  $View = View \cup \{v\}$

12 **RETURN**  $View$

end

**Fig. 3.** Algorithm used for generating mappings

the source relation specified by the correspondence (Figure 4 lines 5 and 6). If, on the other hand,  $corr$  is a many-to-many or many-to-one relation correspondence then the following cases are possible:

- *corr specifies a vertical partitioning between the source relations*: The view is assigned the relation obtained by applying an outer join to the source relations in  $corr$  using  $corr.JP_s$  (Figure 4 lines 8 and 9). We use the outer join for combining the relations in the source schema instead of a natural join to avoid any loss of information. To further illustrate this, consider the example of vertical partitioning presented earlier in Figure 1. A department may not have any associated grants. Therefore, applying a natural join, instead of an outer join, for combining *Department* and *Grant* means losing available information about those departments when joining the two relations.
- *corr specifies a horizontal partitioning between the source relations*: The view is assigned the union of the source relations in  $corr$ . As pointed out earlier, we use the outer union operator instead of the union operator as the relations may not be union incompatible. In addition, the view is augmented with a selection specifying which tuples in the source relations are to be used for populating the target relation (Figure 4 lines 11 and 12).

Secondly, the subroutine specifies the elements that constitute the columns of the view. To do this, it iterates over the attributes of the target relation  $r_t$ , retrieving for each attribute  $att_t$  the associated attribute correspondence among the correspondences in  $corr.AC$ . The following four cases are possible:

## Algorithm DeriveView

inputs  $r_t$ : a relation in the target schema.

$corr$ : a schematic relation correspondence that involves  $r_t$ .

outputs  $view$ : a view for populating  $r_t$ .

begin

```

1  view = new View()
2  view.name = r_t.name
3  q = new Query()
4  IF (getCorrCardinality(corr) ∈ {'one - to - one', 'one - to - many'}) THEN
5    Let r_s be the source relation of corr
6    q ← σcorr.ps r_s
7  IF (getCorrCardinality(corr) ∈ {'many - to - many', 'many - to - one'}) THEN
8    IF (corr.type_s = 'VerticalPartitioning') THEN
9      q ← ⋈corr.JP_s corr.source
10   IF (corr.type_s = 'HorizontalPartitioning') THEN
11     p = getHPPredicate(corr, r_t)
12     q ← ⋈p corr.source
13  ViewColumns = new List()
14  FOR EACH (att_t ∈ r_t.attributes) DO
15    corr_att = getAttCorrespondence(corr.AC, att_t)
16    IF (corr_att = null) THEN // That is, if att_t is a missing attribute
17      IF (att_t IS a key attribute) THEN
18        append(ViewColumns, generateKey());
19      ELSE append(ViewColumns, 'null');
20    IF (getCorrCardinality(corr_att) = 'one - to - one') THEN
21      Let att_s be the source attribute in corr_att
22      append(ViewColumns, corr_att.fs→t + '(' + att_s + ')')
23    IF (getCorrCardinality(corr_att) ≠ 'many - to - many') THEN
24      Let atts1, ..., attsm be the attributes in the source schema that are associated with att_t
25      append(ViewColumns, corr_att.fs→t + '(' + atts1 + ',' + ... + ',' + attsm + ') ↓ ' + att_t)
26  q ← ΠViewColumns q
27  view.query = q
28  RETURN view
end

```

**Fig. 4.** Algorithm for deriving the view for populating a given relation in the target schema

- $att_t$  has no associated correspondence among the set of attribute correspondences specified by  $corr$ . In this case,  $att_t$  is considered as a missing attribute. If  $att_t$  is a key attribute then the call to function  $generateKey()$  is appended to the list of columns of the view, otherwise, the value  $null$  is appended to the list of columns of the view (Figure 4 lines 16-19).  $generateKey()$  is a function for generating unique identifiers: it is used to enable the mapping in the absence of attribute correspondences that specify the values of key attributes of the relation in the target schema.

- $att_i$  is involved in a one-to-one attribute correspondence. In this case, the corresponding source attribute is appended to the list of columns of the view (Figure 4 line 22).
- $att_i$  is involved in a many-to-many, many-to-one or one-to-many attribute correspondence. In this case, a call to the function specified by the attribute correspondence with the source attribute(s) used as input, is appended to the list of columns of the view (Figure 4 lines 24 and 25).

The next section describes an extended, realistic example of automatic mapping generation in the area of proteomics data.

## 4 Using Schematic Correspondences for Deriving Mappings Between Proteomics Data Sources

Proteomics is the study of the set of proteins produced by an organism with the aim of understanding the behaviour of these proteins under varying environments and conditions. There is a growing number of resources that offer a range of approaches for the capture, storage and dissemination of proteome experimental data. While the existence of such resources opens up possibilities for the proteomics community, the diversity of data models creates schema integration challenges. In this respect, we have integrated, in previous work [21], the schemas of four major proteomics data sources, namely PedroDB<sup>1</sup>, GPMDB<sup>2</sup>, PepSeeker<sup>3</sup> and Pride<sup>4</sup>. To do this, we manually specified the mappings between the schemas of these sources and the integration schema using the Automated toolkit [1]. In this section, we show that these mappings can be automatically generated if schematic correspondences of the form presented in this paper are used as input. Due to space limitation, we will present one example of a one-to-one relation correspondence and one example of a many-to-many relation correspondence. Three relations from the integration schema are involved in these examples, namely *IntProtein*, *IntProteinHit* and *IntPeak* (see Figure 5). The *IntProtein* relation describes a protein using an accession number, the name of the gene, its synonyms, the organism in which the protein is to be found, a textual description, the protein amino-acid sequence, *in vivo* modification and the reading frame *rf*. The *IntProteinHit* relation is used to store information about the protein against which all or some of the peptides have been aligned, and links to some information about the protein itself. A protein is (experimentally) identified using a mass spectrometer which produces a spectrum composed of a list of *peaks*. A peak is described by the *IntPeak* relation using the mass-to-charge ratio of the protein ions, *m-to-z*, the peak height, *abundance* and the isotopic pattern around the main peak, *multiplicity*<sup>5</sup>.

<sup>1</sup> <http://pedro.cs.manchester.ac.uk>

<sup>2</sup> <http://gpmdb.thegpm.org>

<sup>3</sup> <http://nwsr.smith.man.ac.uk/pepseeker>

<sup>4</sup> <http://www.ebi.ac.uk/pride>

<sup>5</sup> For further information about the integration schema, the reader is referred to [21].

**Integration Schema**

IntPeak

pk_id	m_to_z	abundance	multiplicity
-------	--------	-----------	--------------

IntProtein

p_id	accession_number	gene_name	synonyms	organism	description	sequence	modifications	rf
------	------------------	-----------	----------	----------	-------------	----------	---------------	----

IntProteinHit

hit_id	all_peptide_matched	expect	score	threshold	p_id
--------	---------------------	--------	-------	-----------	------

**PepSeeker Schema**

SearchMasses

sid	products	title	mass_min	mass_max	int_min	int_max	num_vals
-----	----------	-------	----------	----------	---------	---------	----------

**GPMDB Schema**

Protein

proid	expect	uid	pida	pidb	proseqid
-------	--------	-----	------	------	----------

ProSeq

proseqid	seq	label	label_aux	ref
----------	-----	-------	-----------	-----

**Fig. 5.** Examples of relations from the integration schema, GPMDB schema and PepSeeker schema

*Example 1.* The correspondence  $corr_1$  presented below is an example of a one-to-one relation correspondence that associates the *SearchMasses* relation in the schema of GPMDB with the *IntPeak* relation in the integration schema. It specifies that the attributes *pid* and *m-to-z* of the *IntPeak* relation are associated with the attributes *sid* and *products* of the *SearchMasses* relation, respectively, whereas *abundance* and *multiplicity* are missing attributes in the sense that they are not involved in any attribute correspondence.

```

corr1.source = {SearchMasses}
corr1.target = {IntPeak}
corr1.AC      = {{sid, pk_id, false}, {products, m - to - z, false}}
corr1.ps     = true
corr1.pt     = true

```

Using the schematic correspondence  $corr_1$  together with the *IntPeak* as input to the algorithm presented in Section 3, we obtained the view  $v_{IntPeak}$  presented below for populating the *IntPeak* relation of the integration schema. Notice that the missing attributes *abundance* and *multiplicity* are assigned the null value.

$$v_{IntPeak}.query \leftarrow \Pi_{sid, products, null, null} SearchMasses$$

*Example 2.* The schematic correspondence  $corr_2$  presented below is an example of a many-to-many relation correspondence that associates the relations *Protein* and *ProSeq*

in the schema of GPMDB with the relations *IntProtein* and *IntProteinHit* in the integration schema. It specifies that the relations *Protein* and *ProSeq* should be combined by joining them using the attribute comparison predicate '*Protein.proseqid = ProSeq.proseqid*', and that the relations *IntProtein* and *IntProteinHit* should be combined by joining them using the attribute comparison predicate '*IntProtein.p\_id = IntProteinHit.p\_id*'.

```

corr2.source = {Protein, ProSeq}
corr2.target = {IntProtein, IntProteinHit}
corr2.types = 'VerticalPartitioning'
corr2.AC      = {⟨proid, p_id, false⟩, ⟨proseqid, hit_id, false⟩, ⟨expct, expect, false⟩,
                ⟨seq, sequence, false⟩, ⟨label, accession_number, false⟩, ⟨rf, rf, false⟩}
corr2.JPs    = 'Protein.proseqid = ProSeq.proseqid'
corr2.JPt    = 'IntProtein.p_id = IntProteinHit.p_id'

```

As with *corr<sub>1</sub>*, we used the schematic correspondence *corr<sub>2</sub>* as input to the algorithm *DeriveView* and obtained the views  $v_{IntProtein}$  and  $v_{IntProteinHit}$  presented below for populating the relations *IntProtein* and *IntProteinHit*, respectively.

```

VIntProtein.query ←  $\Pi_{proid, label, null, null, null, null, seq, null, rf}$  (Protein  $\bowtie$  ProSeq)
VIntProteinHit.query ←  $\Pi_{proseqid, null, expect, null, null, proid}$  (Protein  $\bowtie$  ProSeq)

```

## 5 Related Work

A number of authors have proposed models for specifying schematic correspondences between heterogeneous schemas (e.g., in [19, 20]), in most of which a correspondence is defined as an association between attributes of two schemas, i.e., one-to-one or many-to-many attribute correspondences. Schematic correspondences of this form do not provide the necessary information for deriving the mappings between schemas. This is the case, for example of mappings that involve more than one relation in the source or target schemas. To overcome this problem, the authors in Clío [20] exploit functional dependencies and referential integrity constraints between the relations in the source and target schemas. For example, they rely on referential integrity constraints to specify the join predicates for joining relations in the case of vertical partitioning. This is a partial solution in that it does not always lead to the desired schema mapping. For example, referential integrity constraints that can be used for joining the relations involved in a mapping may not exist. And, even if they did exist, they may not encode the join condition that meets the semantics of the desired schema mapping.

Richer models for schematic correspondences were proposed by Pottinger *et al.* [16] and Quix *et al.* [18]. A schematic correspondence in these models can be used to model many-to-many relation and attribute correspondences, for instance. Yet, the information they provide is not sufficient, and human intervention may be needed for deriving the mappings. For example, they do not distinguish between horizontal and vertical partitioning, and they do not specify the condition that can be used for combining relations.

In other proposals, such as those by Magnani *et al.* [14] and Hakimpour [7] *et al.*, correspondences are defined as extensional constraints between the elements of two schemas. For example, correspondences of this form can be used to specify that the



extent of a relation is covered by the extent of another relation. While correspondences of this form are useful for merging schemas, as shown by the authors of these proposals, they do not provide the information necessary for deriving schema mappings.

Kedad *et al.* [10] proposed a solution similar to that proposed in Clio, for generating views that populate the elements of an integration schema. In doing so, they use as input extensional constraints such as those defined by Magnani *et al.* [14] together with functional dependencies and referential integrity constraints specified within the schemas of the sources. Because the correspondences used as input do not provide information that is sufficient for deriving the mappings that encode the semantics desired by the user, the solution proposed by the authors attempts to generate multiple alternatives for populating a given relation in the integration schema. The user is then responsible for choosing the view that implements the desired mapping. In our approach, we use input correspondences that provide information that is sufficient for determining and deriving the desired schema mappings.

## 6 Conclusions

We presented in this paper a model for schematic correspondences that extends and enriches those proposed by Kim *et al.* [12]. The model proposed outperforms existing correspondence models in that it covers the information necessary to cater for automatic derivation of schema mappings. Note that the model of schematic correspondences presented in this paper does not handle all the kinds of correspondences between schemas identified by Kim *et al.*, e.g., we did not consider schematic correspondences associating relations that are semantically not equivalent, i.e., that refer to different semantic domains. Instead, we focused on refining the correspondences that are useful for generating schema mappings. In this respect, we presented an algorithm for automatically generating the views implementing schema mappings between two schemas.

## References

1. Boyd, M., Kittivoravitkul, S., Lazanitis, C., McBrien, P., Rizopoulos, N.: Automed: A bay data integration system for heterogeneous data sources. In: Persson, A., Stirna, J. (eds.) CAiSE 2004. LNCS, vol. 3084, pp. 82–97. Springer, Heidelberg (2004)
2. Calvanese, D., De Giacomo, G., Lenzerini, M., Nardi, D., Rosati, R.: Information integration: Conceptual modeling and reasoning support. In: CoopIS, pp. 280–291. IEEE Computer Society Press, Los Alamitos (1998)
3. Codd, E.F.: Extending the database relational model to capture more meaning. *ACM Trans. Database Syst.* 4(4), 397–434 (1979)
4. Doan, A., Domingos, P., Halevy, A.Y.: Reconciling schemas of disparate data sources: A machine-learning approach. In: SIGMOD Conference, pp. 509–520 (2001)
5. Doan, A., Halevy, A.Y.: Semantic integration research in the database community: A brief survey. *AI Magazine* 26(1), 83–94 (2005)
6. Fagin, R., Kolaitis, P.G., Miller, R.J., Popa, L.: Data exchange: semantics and query answering. *Theor. Comput. Sci.* 336(1), 89–124 (2005)
7. Hakimpour, F., Geppert, A.: Global schema generation using formal ontologies. In: Spaccapietra, S., March, S.T., Kambayashi, Y. (eds.) ER 2002. LNCS, vol. 2503, pp. 307–321. Springer, Heidelberg (2002)

8. Halevy, A.Y., Franklin, M.J., Maier, D.: Principles of dataspace systems. In: Vansummeren, S. (ed.) PODS, pp. 1–9. ACM, New York (2006)
9. Halevy, A.Y., Rajaraman, A., Ordille, J.J.: Data integration: The teenage years. In: Dayal, U., Whang, K.-Y., Lomet, D.B., Alonso, G., Lohman, G.M., Kersten, M.L., Cha, S.K., Kim, Y.-K. (eds.) VLDB, pp. 9–16. ACM, New York (2006)
10. Kedad, Z., Bouzeghoub, M.: Discovering view expressions from a multi-source information system. In: CoopIS, pp. 57–68. IEEE Computer Society, Los Alamitos (1999)
11. Kim, W., Choi, I., Gala, S.K., Scheevel, M.: On resolving schematic heterogeneity in multidatabase systems. In: Modern Database Systems, pp. 521–550. ACM Press and Addison-Wesley (1995)
12. Kim, W., Seo, J.: Classifying schematic and data heterogeneity in multidatabase systems. *IEEE Computer* 24(12), 12–18 (1991)
13. Lenzerini, M.: Data integration: A theoretical perspective. In: Popa, L. (ed.) PODS, pp. 233–246. ACM, New York (2002)
14. Magnani, M., Rizopoulos, N., McBrien, P., Montesi, D.: Schema integration based on uncertain semantic mappings. In: Delcambre, L.M.L., Kop, C., Mayr, H.C., Mylopoulos, J., Pastor, Ó. (eds.) ER 2005. LNCS, vol. 3716, pp. 31–46. Springer, Heidelberg (2005)
15. McCann, R., AlShebli, B.K., Le, Q., Nguyen, H., Vu, L., Doan, A.: Mapping maintenance for data integration systems. In: Böhm, K., Jensen, C.S., Haas, L.M., Kersten, M.L., Larson, P.-Å., Ooi, B.C. (eds.) VLDB, pp. 1018–1030. ACM, New York (2005)
16. Pottinger, R., Bernstein, P.A.: Creating a mediated schema based on initial correspondences. *IEEE Data Eng. Bull.* 25(3), 26–31 (2002)
17. Pottinger, R., Bernstein, P.A.: Merging models based on given correspondences. In: VLDB, pp. 826–873 (2003)
18. Quix, C., Kensché, D., Li, X.: Generic schema merging. In: 9th International Conference on Advanced Information Systems Engineering, pp. 127–141. Springer, Heidelberg (2007)
19. Rahm, E., Bernstein, P.A.: A survey of approaches to automatic schema matching. *VLDB J.* 10(4), 334–350 (2001)
20. Yan, L.-L., Miller, R.J., Haas, L.M., Fagin, R.: Data-driven understanding and refinement of schema mappings. In: SIGMOD Conference, pp. 485–496 (2001)
21. Zamboulis, L., Fan, H., Belhajjame, K., Siepen, J.A., Jones, A.C., Martin, N.J., Poulouvasilis, A., Hubbard, S.J., Embury, S.M., Paton, N.W.: Data access and integration in the ispider proteomics grid. In: Leser, U., Naumann, F., Eckman, B. (eds.) DILS 2006. LNCS (LNBI), vol. 4075, pp. 3–18. Springer, Heidelberg (2006)

# The Problem of Transitivity of Part-Whole Relations in Conceptual Modeling Revisited

Giancarlo Guizzardi

Ontology and Conceptual Modeling Research Group (NEMO)  
Computer Science Department,  
Federal University of Esp rito Santo (UFES), Brazil  
gguizzardi@inf.ufes.br

**Abstract.** Parthood is a relation of fundamental importance in a number of disciplines including cognitive science, linguistics and conceptual modeling. However, one classical problem for conceptual modeling theories of parthood is deciding on the transitivity of these relations. This issue is of great importance since transitivity plays a fundamental role both conceptually (e.g., to afford inferences in problem-solving) and computationally (e.g., to afford propagations of properties and events in a transitive chain). In this article we address this problem by presenting a solution to the case of part-whole relations between functional complexes, which are the most common types of entities represented in conceptual models. This solution comes in two parts. Firstly, we present a formal theory founded on results from formal ontology and linguistics. Secondly, we use this theory to provide a number of visual patterns that can be used to isolate scopes of transitivity in part-whole relations represented in diagrams.

## 1 Introduction

Parthood is a relation of fundamental importance from the cognitive and linguistic perspectives [1,2]. In conceptual modeling, part-whole relations have also been considered of substantial significance. It is present in practically all conceptual/object-oriented modeling languages (e.g., OML, UML, EER) and, although it has not yet been adopted as a modeling primitive in the semantic web languages, many authors have already pointed out its relevance for reasoning in description logics (e.g., [3]).

Theories of parts have been a central point of interest in philosophical enquiry since the pre-socratic philosophers and along the years many precise formal theories have been developed (e.g., General Extensional Mereology, Calculus of Individuals) [4]. These formal theories provide an important starting point for the understanding and axiomatization of the notion of Part. Nonetheless, despite their importance, there are many controversial properties that they ascribed to the part-whole relation that cannot be accepted by cognitive and conceptual theories of parthood [2,4,5]. One of these controversial properties is the unrestricted transitivity of parthood.

In philosophical ontology, all mereological theories include transitivity as an axiom for the formal part-whole relation. Also in many conceptual modeling languages, the part-whole relations are considered to be transitive (e.g., composition

relation in UML). However, there are many counter-examples in the literature of part-whole relations in which transitivity is not warranted by language or cognition. For instance: (i) Rio de Janeiro is part of Brazil and Brazil is part of the United Nations (UN), it is not the case that Rio de Janeiro is part of the UN; (ii) the heart is part of the musician, the musician is part of the orchestra, but the heart is not part of the orchestra [2,4].

In [4], we revised the classical typology of parthood relations proposed in [1]. As demonstrated there, the six linguistically-motivated types of part-whole relation proposed by [1] give rise to only four distinct ontological types, namely: (a) *subquantity-quantify* (e.g., alcohol-wine) – modeling parts of an amount of matter which are unified in a whole due to a topological connection relation; (b) *member-collective* (e.g., a specific tree – the black forest) – modeling a collective entity in which all parts play an equal role w.r.t. the whole; (c) *subcollective-collective* (e.g., the north part of the black forest- the black forest); (d) *component – functional complex* (e.g., heart-circulatory system, engine – car) - modeling an entity in which all parts play a different role w.r.t. the whole, thus, contributing to the functionality of the latter. Moreover, in [4], we have demonstrated that there is a strong connection between the issue of transitivity of parthood and the type of the relation being considered. For instance, it can be formally proved that *subquantity-quantify* and *subcollective-collective* are always transitive. Moreover, although *member-collective* is never transitive, a combination of *member-collective* and *subcollective-collective* is again always transitive.

Despite their relevance, these results do not suffice as a general solution for the problem of transitivity in conceptual modeling, since most of the entities which are represented in conceptual models are actually functional complexes (e.g., Persons, Cars, Computers, Cells, Organs, Organizations, Organizational Units). Parthood relations between functional complexes are neither transitive nor intransitive, but non-transitive, i.e., transitive in certain occasions and intransitive in others [6]. For this reason, the current attempts to provide real-world semantics for part-whole relations in the conceptual modeling literature simply exclude transitivity from the list of primary properties of part-whole relations [7]. This solution is, again, non-satisfactory. From both a conceptual and computational point of view, there are many benefits from explicitly reasoning with the transitivity of parthood. Examples include propagation of properties and events along the transitive chain of parts (e.g., spatial change, rotation, creation, destruction) and diagnostic reasoning with transitive parts in biomedical conceptual models. For this reason it is fundamental to understand why transitivity holds in some cases and not in others, and to determine the contexts in which part-whole relations are guaranteed to be transitive.

The contributions of this article are two-fold. Firstly, we build on a formal ontological analysis of relations [8] and on the pioneering theory of transitivity of linguistic functional parthood relations [2] to propose a formal theory and typology of part-whole relations between functional complexes. Secondly, we employ this theory to propose a number of visual patterns that can be used as a methodological support for the identification of contexts of transitivity for this mostly common type of part-whole relations in conceptual modeling.

The remainder of this article is organized as follows. Section 2 briefly discusses an ontological analysis of relations based on a Foundational Ontology. Section 3 employs this analysis to interpret the specific case of part-whole relations in functional

complexes. Section 4 uses the results of section 3 to propose a typology of functional part-whole relations and a number of visual patterns for isolating the context of transitivity in conceptual models. Finally, section 5 elaborates on some final consideration.

## 2 Background: An Ontological Analysis of Relations

In [8], we have presented an in depth analysis of domain relations from an ontological point of view. In particular, we have employed the Unified Foundational Ontology (UFO), a formal framework which has been constructed by considering a number of theories from formal ontology in philosophy, but also cognitive science, linguistics and philosophical logics. In a number of papers, UFO has been successfully employed to analyze and provide real-world semantics for conceptual modeling grammars and specifications. Here, we make a very brief presentation of this foundational ontology and concentrate only on the categories which are germane to the purposes of this article. For an in depth discussion on the categories of UFO, empirical evidence for the choice of its categories as well as formal categorization, one should see [4].

A fundamental distinction in this ontology is between the categories of *Objects* and *Tropes*. *Objects* are existentially independent entities. Examples include ordinary objects of everyday experience such as an individual person, an organization, an organ, a car, and The Rolling Stones<sup>1</sup>. The word *Trope*, in contrast, denotes, what is sometimes named an individualized (objectified) property, a moment, an accident, or property in particular. A trope is an individual that can only exist in other individuals. Typical examples of tropes are a color, a connection, an electric charge, a symptom, a covalent bond. Tropes have in common that they are all dependent of other individuals (their bearers), i.e., an important feature that characterizes all *tropes* is that they can only exist in other individuals (in the way in which, for example, electrical charge can exist only in some conductor, or that a covalent bond can only exist if those connecting atoms exist). To put it more technically, we say that they *inhere* on other individuals. *Inherence* (symbolized as *i*) is a formal relation that has the following meta-properties: (a) irreflexivity; (b) asymmetry; (c) intransitivity; (d) exclusive *existential dependence*, i.e., if *x* inheres in *y* then *x* cannot exist in a given situation without that *very specific y* existing in that same situation (existential dependence) and there is no *z* different from *y* such that *x* inheres in *z*. Finally, existential dependence can also be used to differentiate *intrinsic tropes* and *Relators* (relational tropes): *intrinsic tropes* are dependent of one single individual (e.g., color, a headache, a temperature); *relators* depend on a plurality of individuals (e.g., an employment, a marriage).

Another important distinction in the UFO ontology is within the categories of relations. Following the philosophical literature, it recognizes two broad categories of relations, namely, *material* and *formal* relations [8]. Formal relations hold between two or more entities directly, without any further intervening individual. Examples include the relations of *existential dependence* (*ed*), *Subtype*, *instantiation* (::), formal

---

<sup>1</sup> According to this definition, the category of objects can include quantities, collectives and functional complexes. However, all objects we consider in this article are examples of functional complexes.

parthood ( $<$ ), inherence ( $i$ ), among many others not discussed here [4]. Domain relations such as *working at*, *being enrolled at*, and *being the husband of* are of a completely different nature. These relations, exemplifying the category of *Material relations*, have material structure of their own. Whilst a formal relation such as the one between Paul and his headache  $x$  holds directly and as soon as Paul and  $x$  exist, for a material relation of *being treated in* between Paul and the medical unit  $MU_1$  to exist, another entity must exist which *mediates* Paul and  $MU_1$ . These entities are termed *relators*.

Relators are individuals with the power of connecting entities. For example, a medical treatment connects a patient with a medical unit; an enrollment connects a student with an educational institution; a covalent bond connects two atoms. The notion of relator is supported by several works in the philosophical literature [9] and, they play an important role in answering questions of the sort: what does it mean to say that John is married to Mary? Why is it true to say that Bill works for Company X but not for Company Y? Again, relators are special types of tropes which, therefore, are existential dependent entities. The relation of *mediation* (symbolized  $m$ ) between a relator  $r$  and the entities  $r$  connects is a sort of (non-exclusive) inherence and, hence, a special type of existential dependence relation. It is formally required that a relator mediates at least two distinct individuals [4].

An important notion for the characterization of relators (and, hence, for the characterization of material relations) is the notion of *foundation*. Foundation can be seen as a type of *historical dependence* [10], in the way that, for instance, an instance of *being kissed* is founded on an individual *kiss*, or an instance of *being punched by* is founded on an individual *punch*, an instance of *being connected to* between airports is founded on a particular flight connection. Suppose that John *is married to* Mary. In this case, we can assume that there is an individual relator  $m_1$  of type *marriage* that mediates John and Mary. The foundation of this relator can be, for instance, a wedding event or the signing of a social contract between the involved parties. In other words, for instance, a certain event  $e_1$  in which John and Mary participate can create an individual marriage  $m_1$  which existentially depends on John and Mary and which mediates them. The event  $e_1$  in this case is the foundation of relator  $m_1$ .

Now, let us elaborate on the nature of the relator  $m_1$ . There are many intrinsic tropes that John acquires by virtue of being married to Mary. For example, imagine all the legal responsibilities that John has in the context of this relation. These newly acquired properties are intrinsic tropes of John which, therefore, are existentially dependent on him. However, these tropes also depend on the existence of Mary. We name this type of trope *externally dependent tropes*, i.e., externally dependent tropes are intrinsic tropes that inhere in a single individual but are existentially dependent on (possibly multiple) other individuals. The individual which is the aggregation of all externally dependent tropes that John acquires by virtue of being married to Mary is named a *qua individual* (in this case, John-qua-husband-of-Mary). A qua individual is, thus, defined as an individual composed of all externally dependent tropes that inhere in the same individual and share the same foundation. In the same manner, by virtue of being married to John, Mary bears an individual Mary-qua-wife-of-John.

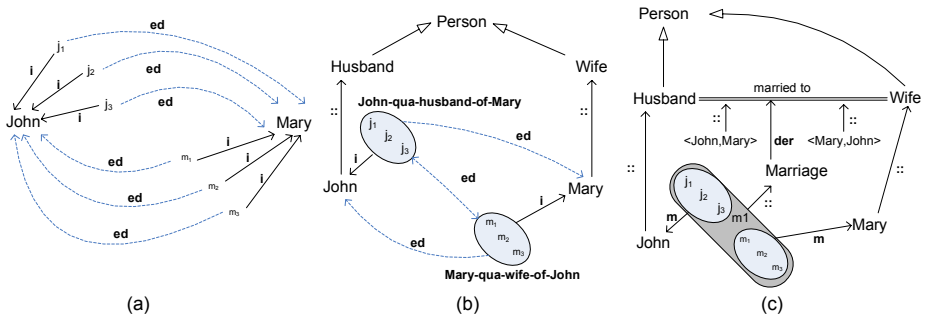
The notion of qua individuals is the ontological counterpart of what has been named *role instance* in the literature [11] and represent the properties that characterize a particular mode of participation of an individual in a relation. Now, the entity which

is the sum of all qua individuals that share the same foundation is a relator. In this example, the relator  $m_1$  which is the aggregation of all properties that John and Mary acquire by virtue of being married to each other is an instance of the relational property *marriage*.

The relator  $m_1$  in this case is said to be the *truthmaker* of propositions such as “John is married to Mary”, “Mary is married to John”, “John is the husband of Mary”, and “Mary is the wife of John”. In other words, material relations such as *being married to*, *being legally bound to*, *being the husband of* can be said to hold for the individuals John and Mary because and only because there is an individual relator marriage  $m_1$  mediating the two. Thus, as demonstrated in [8], material relations are purely linguistic/logical constructions which are founded on and can be completely derived from the existence of relators. In fact, in [8], we have defined a formal relation of derivation (symbolized as *der*) between a relator type (e.g., Marriage) and each material relation which is derived from it.

Finally, there is an intimate connection between qua individuals and role types: let T be a natural type (kind) instantiated by an individual x, and let R be a role type specializing T. We have that there is a qua individual type Q such that x instantiates R iff x bears an instance of Q. Alternatively, we have that for every role type R there is a relator type RR such that x instantiates R iff x is mediated by an instance of RR. Note that this conforms to the formal property of roles as *relationally dependent* types [12].

The summary of the discussion promoted in this section is illustrated in figures 1a-c. Figure 1.a, illustrates the inherence relation between John and his externally dependent tropes which are existentially dependent on Mary (as well as analogous relations in the converse direction). In figure 1.b, John instantiates the role type Husband (which is a specialization of the natural type (Male) Person) iff there is a qua individual John-qua-husband-of-Mary which inheres in John. Moreover, this figure illustrates that the qua individuals John-qua-husband-of-mary and Mary-qua-wife-of-John are mutually existentially dependent. In other words, John cannot be the Husband of Mary without Mary being the wife of John [4]. Finally, figure 1.c shows that the material relation *married to* is derived from the relator type Marriage and, thus, tuples such as  $\langle \text{John}, \text{Mary} \rangle$  and  $\langle \text{John}, \text{Mary} \rangle$  are instances of this relation iff there is an instance of Marriage that mediates the elements of the tuple.



**Fig. 1.** (a-left) Objects and their inhering externally dependent tropes. (b-center) Objects, their instantiating roles and their inhering qua individuals. (c) Material Relations are founded on relators that mediate their relata.

Notice that the relation between the two qua individuals and the relator  $m_1$  is an example of formal relation of parthood [8]. As previously discussed, formal parthood conforms to the meta-properties prescribed by mereology and, therefore, is always transitive. Another example of a parthood relation that conform to axioms of mereology is the spatial (temporal) part-whole relation between regions of space (or time) [9]. One of the major points advocated in this article is that the domain part-whole relations that interest us in conceptual modeling are not formal but material relations: the fact that Brazil is part of the United Nations or that Paul’s transplanted heart is part of his body demand for the existence of founding events and consequent relators.

### 3 Functional Complexes and Functional Dependence

As we previously discussed, the parts of a functional complex have in common the fact that they all possess a functional link with the complex. In other words, they all contribute to the functionality (or the behavior) of the complex. According to [2], parthood relations between complexes represent, aside from the mereological relation itself, relations of *functional dependence*. Take the example of figure 2. Following [2], we claim that this type of relationship represented between the types Heart and Body is what is termed *Generic Functional Dependence* between two types. This relationship can be defined as follows: (1)  $GFD(X,Y) \equiv \forall x (x::X) \wedge F(x,X) \rightarrow \exists y \neg(y = x) \wedge (y::Y) \wedge F(y,Y)$ .



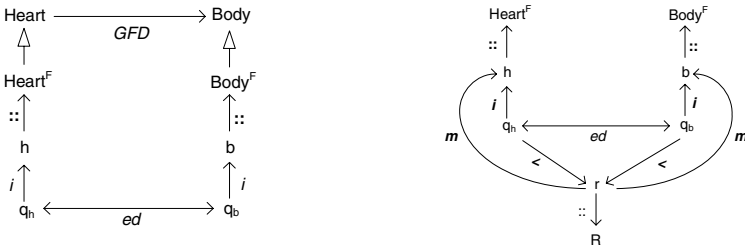
Fig. 2. A parthood relation between two Functional Complex Types

The predicate  $F(x,X)$  in formula (1) has the meaning  $x$  functions as an  $X$ . In Vieu and Aurnague’s theory [2], it is not necessary for an  $X$  that it functions as an  $X$ . So for instance, it is not the case that in every circumstance an engine functions as an engine. We thus can think of a type  $X^F$  which is a specialization of  $X$  according to the specialization condition expressed by the predicate  $F(x,X)$ , so that every  $X^F$  is a  $X$  functioning as a  $X$ . We name the type  $X^F$  a *functional restriction* of  $X$ . Notice that  $X^F$  in this case is a type which can be characterized by the qua individual  $q_x$ . This qua individual, in turn, stands for the tropes bearing in an  $X$ ’s while functioning as such, or the particular behaviour of an  $X$  while functioning as an  $X$ . For instance, an engine  $x$  can have the property of emitting a certain number of decibels or being able to perform certain tasks only when functioning as an engine.

In figure 3.a, we can create specializations of the types Heart and Body to the types  $Heart^F$  (*FunctioningHeart*) and  $Body^F$  (*FunctioningBody*). In this picture, the arrow with the hollow head represents subtyping. The symbols  $::$ ,  $i$  and  $ed$  represent instantiation, inherence and existential dependence, respectively. Whenever a heart functions as such, i.e., whenever it instantiates the type *FunctioningHeart*, there is a qua individual  $q_h$  that inheres in it. *Mutatis Mutandis*, the same goes to Body and



*FunctioningBody* in this picture. As represented in this picture, the qua individuals  $q_h$  and  $q_b$  are existentially dependent on each other. In this case,  $ed(q_h, q_b)$  can be interpreted as “the heart functioning behavior existentially depends on the body functioning behavior”. In this model the converse also holds, i.e., that  $ed(q_b, q_h)$ , or that “the body functioning behaviour existentially depends on the heart functioning behavior”. Additionally, according to our model, a heart functioning  $q_h$  must inhere in a heart  $h$ . Likewise, a body functioning  $q_b$  must inhere in a body  $b$ . From this we have that whenever a heart  $h$  functions as a heart (i.e.,  $i(q_h, h)$ ) there must exist a body functioning behavior  $q_b$  (from  $ed(q_h, q_b)$ ), which in turn, inheres a body  $b$  (i.e.,  $i(q_b, b)$ ). In other words, whenever a heart  $h$  functions as a heart, there must be a body  $b$  functioning as a body. Again, from the model of figure 3.a we can derive the converse information, namely, that whenever a body  $b$  functions as a body, there must be a heart  $h$  functioning as a heart.



**Fig. 3. (a-left)** Representation of Types with Generic Functional Dependence and their Functional Restrictions. **(b)** Representation of the relator instance composed of two functional qua individuals.

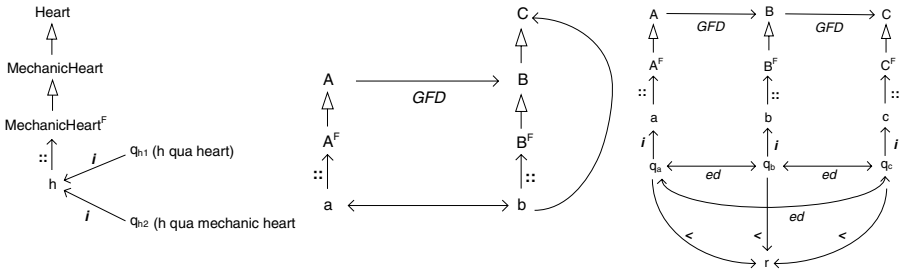
By definition of the relational qua individuals,  $q_h$  and  $q_b$  in figure 3.a are externally dependent tropes that compose a relator  $r$  that, in turn, can be said to mediate the instances of *FunctioningHeart* and *FunctioningBody*. This idea is depicted in figure 3.b. The symbols  $m$  and  $<$  in this picture represent the mediation relation and the formal proper parthood relation, respectively.

The relator universal  $R$  of which the relator  $r$  in figure 3.b is an instance, can be said to derive the material relation  $\varphi_R$  between the universals *FunctioningHeart* and *FunctioningBody*. We shall define here the more general binary predicate  $\varphi(x,y) \equiv \exists r m(r,x) \wedge m(r,y)$ . In other words,  $\varphi(x,y)$  holds iff there is a relator  $r$  which mediates these two individuals. More naturally, in this case, we can say that  $\varphi$  hold of  $x$  and  $y$  of type  $X$  and  $Y$  iff  $x$  to function as an  $X$  is depends on  $y$  functioning as a  $Y$ , and vice-versa. Notice that the functional restriction *FunctioningHeart* (*FunctioningBody*) is indeed *relationally dependent* and, consequently, it conforms to the characterization of role types previously discussed: a *FunctioningHeart* is a *Heart* functioning as a *Heart* in relation to a *Body* functioning as a *Body*, and vice-versa. To put in different terms, these functional restrictions of natural types are sorts of *Roles* types.

The predicate  $\varphi$  to hold for instances of functional restrictions  $X^F$  and  $Y^F$  requires the presence of a relator  $r$  to mediate these instances. This requires that the functional qua individuals inhering in the mediated instances of  $X^F$  and  $Y^F$  share a genuine

foundation. The formula (1) of generic functional dependence between X and Y can then be better expressed as: (2).  $GFD(X,Y) \equiv \forall x (x::X) \wedge F(x,X) \rightarrow \exists y (y::Y) \wedge F(y,Y) \wedge \varphi(x,y)$ . Notice that, by definition, a relator must mediate at least two distinct individuals. As a consequence, we have that  $\varphi(x,y)$  implies  $\neg(y = x)$ , rendering this condition superfluous in the consequent of formula (2).

Suppose that the universal X is a specialization of another universal A. Then not only every X is an A but whenever an X functions as such it also functions as an A [2]. For example, suppose that X and A are the types *MechanicHeart* and *Heart*, respectively. Whenever a *MechanicHeart* functions as a *MechanicHeart*, it also functions as a *Heart*, or alternatively, whenever a *MechanicHeart* bears the behaviour (or properties) of a functioning *MechanicHeart*, then it also bears the properties of a functioning *Heart*. This is illustrated in figure 4.a. We thus have that (3).  $(F(x,X) \wedge \text{Subtype}(X,Y)) \rightarrow F(x,Y)$ .



**Fig. 4.** (a-left) Propagation of Functioning to the Supertype. (b-center) Propagation of Functioning to the Supertype. (c-right) Transitivity of General Functional Dependence.

Suppose the situation depicted in figure 4.b. The universal A is generally functionally dependent on universal B. Thus, for every instance *a* of A that functions as such there is an instance *b* of B functioning as a B. Moreover, the predicate  $\varphi$  holds for *a* and *b*. Now, since *b* is also a C and, due to (3), *b* also functions as a C. Hence, we have that whenever an instance *a* of A functions as such there is an instance *b* of C that functions as a C. Since  $\varphi(a,b)$ , we can derive that  $GFD(A,C)$ . Thus, we have that the following is always true: (4).  $GFD(X,Y) \wedge \text{Subtype}(Y,Z) \rightarrow GFD(X,Z)$ .

Now, suppose the situation depicted in figure 4.c. In this model, every instance *a* of A functioning as an A bears a particular *q<sub>a</sub>* behaviour. The qua individual *q<sub>a</sub>* is existentially dependent on the qua individual *q<sub>b</sub>*, i.e., on the behaviour of a *b* functioning as a B. However, this model also represents that if *b* functions as a B (bears *q<sub>b</sub>*) there is a *c* functioning as a C, i.e., bearing a C behavior *q<sub>c</sub>*. Due to transitivity of existential dependence [4], we have that *q<sub>a</sub>* is existential dependent also on *q<sub>c</sub>*. Additionally, *q<sub>a</sub>* and *q<sub>b</sub>* share the same foundation and so do *q<sub>b</sub>* and *q<sub>c</sub>*. Thus, *q<sub>a</sub>* and *q<sub>c</sub>* also must share the same foundation. In other words, whatever is responsible for creating *q<sub>a</sub>* and *q<sub>b</sub>* must also be responsible for creating *q<sub>c</sub>*. By definition, a relator is an aggregation of qua individuals that share the same foundation. We can then

define a relator  $r$  which consists of  $q_a$ ,  $q_b$  and  $q_c$ . Consequently, we have that  $\varphi(a,b)$ ,  $\varphi(b,c)$  and  $\varphi(a,c)$ . Now, we have that for every instance  $a$  of  $A$  functioning as an  $A$ , there is an instance of  $c$  functioning as a  $C$ . Since  $\varphi(a,c)$ , we then have that  $GFD(A,C)$ . This argument shows that the following is always true: (5).  $GFD(X,Y) \wedge GFD(Y,Z) \rightarrow GFD(X,Z)$ .

Although formula (2) defines the notion of general dependence, we need in addition to establish that a functional dependence link holds precisely between two individual entities  $x$  and  $y$ : (6).  $IFD(x,X,y,Y) \equiv GFD(X,Y) \wedge x::X \wedge y::Y \wedge (F(x,X) \rightarrow F(y,Y))$ . This predicate termed *individual functional dependence* states that if an individual  $x::X$  is *individually functionally dependent* of another individual  $y::Y$  in a given situation then: (i) there is a generic functional dependence between their types; (ii)  $x$  and  $y$  are classified as those given types in that situation; (iii) for  $x$  to function as a  $X$  in that situation, then  $y$  must function as a  $Y$ .

An example of individual functional dependence is one between a particular heart  $h$  and a particular body  $b$  in figure 2. As discussed, there is a generic functional dependence between the types Heart and Body, and if in a given circumstance a heart  $h$  functions as a heart there is a body  $b$  that functions as a body in that circumstance.

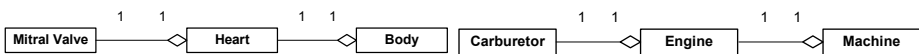
#### 4 A Typology of Functional Part-Whole Relations and Visual Patterns for Isolating the Scope of Transitivity

Let us now return to the example of figure 2 of a parthood relation between the universals Heart and Body. In this model, a particular heart  $h$  is not only functionally dependent of a body  $b$  in a given situation, but  $h$  is also part of  $b$ . This type of the parthood relation is termed in [2] *direct functional parthood of type 1*:

**Definition 1 (Direct Functional Part of type 1):** An individual  $x$  instance of  $X$  is a direct functional part of type 1 of an individual  $y$  of type  $Y$  (symbolized as  $d_1(x,X,y,Y)$ ) iff  $x$  is a part of  $y$  and  $x$  is individually functionally dependent of  $y$ . Formally,  $d_1(x,X,y,Y) \equiv ((x < y) \wedge IFD(x,X,y,Y))$ . ■

Examples of  $d_1$  include cuff-sleeve, stem-plant, carburetor-engine, finger-hand, hand-arm, arm-body, hand-body, heart-body, heart-circulatory system. In conformance with the findings of [3], we propose that a parthood relation between two *functional complexes* (such as the one depicted in figure 2) should be interpreted as a case of *direct functional parthood*. In this specific case, the model implies that:  $\forall x x::Heart \rightarrow \exists y y::Body \wedge d_1(x,Heart,y,Body))$ .

Now, suppose that we have a model such as the one represented of figure 5.a.



**Fig. 5. (a-left)** Examples of direct functional part of type 1. **(b)** Examples of direct functional part of type 1: transitivity always hold across parthood relations of this type.

In this case, both the relationships between Heart and Body, and between Mitral Valve and Heart, are mapped in the instance level to cases of *direct functional parthood*(1), i.e., (i)  $\forall x x::\text{Heart} \rightarrow \exists y y::\text{Body} \wedge d_1(x,\text{Heart},y,\text{Body})$ ; (ii)  $\forall x x::\text{MitralValve} \rightarrow \exists y y::\text{Heart} \wedge d_1(x,\text{MitralValve},y,\text{Heart})$ . The important question at this point is: from (i) and (ii), can we derive formula (iii)  $\forall x x::\text{MitralValve} \rightarrow \exists y y::\text{Body} \wedge d_1(x,\text{MitralValve},y, \text{Body})$ . Notice that (iii) follows from (i) and (ii) iff  $d_1$  is transitive. Thus, the this question can be rephrased as: is direct functional parthood(1) a transitive relation?

In the sequel we demonstrate that this is indeed the case. The abbreviations in the proofs are: (a) TFP (transitivity of formal parthood); (b) TLI (transitivity of the logical implication); (c) EC (Elimination of the Disjunction), and (d) IC (Introduction of the Disjunction).

**(T1) Theorem 1:  $d_1(x,X,y,Y) \wedge d_1(y,Y,z,Z) \rightarrow d_1(x,X,z,Z)$**

**Proof:**

- |   |   |
|---|---|
| 1. $d_1(x,X,y,Y)$   | T1  |
| 2. $d_1(y,Y,z,Z)$   | T1  |
| 3. $(x < y) \wedge \text{IFD}(x,X,y,Y)$   | 1, Definition 1                                     |
| 4. $\text{GFD}(X,Y) \wedge x::X \wedge y::Y \wedge (F(x,X) \rightarrow F(y,Y))$ | 3, (6)  |
| 5. $(y < z) \wedge \text{IFD}(y,Y,z,Z)$   | 2, Definition 1                                     |
| 6. $\text{GFD}(Y,Z) \wedge y::Y \wedge z::Z \wedge (F(y,Y) \rightarrow F(z,Z))$ | 5, (6)  |
| 7. $(x < z)$  | 3,5, TFP  |
| 8. $\text{GFD}(X,Z)$  | 4,6, (5)  |
| 9. $(x::X) \wedge (z::Z)$   | 4,6, EC   |
| 10. $(F(x,X) \rightarrow F(z,Z))$   | 4,6, TLI  |
| 11. $\text{GFD}(X,Z) \wedge (x::X) \wedge (z::Z) (F(x,X) \rightarrow F(z,Z))$   | 8,9,10, IC  |
| 12. $\text{IFD}(x,X,z,Z)$   | 11, (6)   |
| 13. $(x < z) \wedge \text{IFD}(x,X,z,Z)$  | 7,12, IC  |
| 14. $d_1(x,X,z,Z)$  | 13, Definition 1 <span style="float:right">□</span> |

We can generalize this result for any chain of direct functional dependence in a model. Another example of such case is depicted in figure 5.b.

In models such as 5.a-b, the parthood relation represents functional dependence in both directions. Take for instance figure 5.b. The minimum cardinality constraint of 1 in the Engine association end of the aggregation relation between Carburator and Engine implies that every instance of Carburator necessitates an Engine to function as a Carburator. Likewise, the minimum cardinality constraint of 1 in the Carburator association end of that relation implies that every Engine necessitates a Carburator to function as an Engine. [2] names this type of functional parthood in which *x is part of y* but *y as Y is individually functionally dependent on x as an X direct functional parthood* (2):

**Definition 2 (Direct Functional Part of type 2):** An individual  $x$  instance of  $X$  is a direct functional part of type 2 of an individual  $y$  of type  $Y$  (symbolized as  $d_2(x,X,y,Y)$ ) iff  $x$  is a part of  $y$  and  $y$  is individually functionally dependent of  $x$ . Formally,  $d_2(x,X,y,Y) \equiv (x < y) \wedge \text{IFD}(y,Y,x,X)$ . ■

Examples of  $d_2$  include wall-house, engine-car, electron-atom, atom-molecule, finger-hand, hand-arm, cell-heart, feather-canary. In the sequel, we prove that  $d_2$  is also transitive.

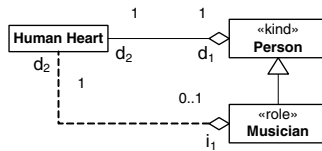
**(T2) Theorem 2:  $d_2(x,X,y,Y) \wedge d_2(y,Y,z,Z) \rightarrow d_2(x,X,z,Z)$**

**Proof:**

- |   |   |
|---|---|
| (1). $d_2(x,X,y,Y)$   | T2  |
| (2). $d_2(y,Y,z,Z)$   | T2  |
| (3). $(x < y) \wedge \text{IFD}(y,Y,x,X)$   | 1, Definition 2                                     |
| (4). $\text{GFD}(Y,X) \wedge y::Y \wedge x::X \wedge (F(y,Y) \rightarrow F(x,X))$ | 3, (6)  |
| (5). $(y < z) \wedge \text{IFD}(z,Z,y,Y)$   | 2, Definition 2                                     |
| (6). $\text{GFD}(Z,Y) \wedge z::Z \wedge y::Y \wedge (F(z,Z) \rightarrow F(y,Y))$ | 5, (6)  |
| (7). $(x < z)$  | 3,5, TFP  |
| (8). $\text{GFD}(Z,X)$  | 4,6, (5)  |
| (9). $(z::Z) \wedge (x::X)$   | 4,6, EC   |
| (10). $(F(z,Z) \rightarrow F(x,X))$   | 4,6, TLI  |
| (11). $\text{GFD}(Z,X) \wedge (z::Z) \wedge (x::X) (F(z,Z) \rightarrow F(x,X))$   | 8,9,10, IC  |
| (12). $\text{IFD}(z,Z,x,X)$   | 11, (6)   |
| (13). $(x < z) \wedge \text{IFD}(z,Z,x,X)$  | 7,12,IC   |
| (14). $d_2(x,X,z,Z)$  | 13, Definition 2 <span style="float:right">□</span> |

Whenever in a conceptual model we have a representation of a parthood relation between complex objects such as in figures 5.a-b, we have both a case of  $d_1$  and a case of  $d_2$ . In particular, the model of figure 5.b implies both the formulae: (i)  $\forall x x::\text{Carburator} \rightarrow \exists y y::\text{Engine} \wedge d_1(x,\text{Carburator},y,\text{Engine})$  and (ii)  $\forall x x:: \text{Engine} \rightarrow \exists y y:: \text{Carburator} \wedge d_2(y,\text{Carburator},x,\text{Engine})$ . Since both  $d_1$  and  $d_2$  are transitive, we maintain that transitivity holds within any chain of direct functional dependence relations in a conceptual model.

Now, take for instance the relationship depicted in figure 6 below.



**Fig. 6.** Example of indirect functional part of type 1(from Human Heart to Musician)

Every human heart necessitates a person, and every person necessitates a human heart, i.e., both  $d_1$  and  $d_2$  hold between direct instances of human heart and person. Moreover, every musician is a person. So, as any person, a musician necessitates a human heart, i.e.,  $d_2$  holds also between instances of human heart and musician. However, it is not the case that a *direct functional dependence* holds between human heart and musician. A human heart necessitates a person, but this person does not have to be a musician (this is made evident by the cardinality 0..1 of the inherited relation

between these two universals). This type of relationship is termed *indirect functional parthood* (1) in [2] and it is defined as follows:

**Definition 3 (Indirect Functional Part of type 1):**  $i_1(x, X, y, Y) \equiv (x < y) \wedge \text{IIFD}(x, X, y, Y)$ .  $\text{IIFD}(x, X, y, Y)$  is the relation of *individual indirect functional dependence* and is defined as (7).  $\text{IIFD}(x, X, y, Y) \equiv y::Y \wedge \exists Z (\text{Subtype}(Y, Z) \wedge \text{IFD}(x, X, y, Z))$ . ■

To put it in a simple way,  $x$  as an  $X$  is individually indirect functional dependent of  $y$  as a  $Y$  iff for  $x$  to function as an  $X$ ,  $y$  must function as a  $Z$ , whereas  $Z$  is a more general universal (subsuming that  $Y$ ) that  $y$  instantiates. Examples of  $i_1$  include handle-door (with “movable entity” for type subsuming “door”), door-house (with “wall, enclosure or building” subsuming “house”), engine-car (with “machine” subsuming “car”), brick-wall (with “construction” subsuming “wall”), valve-carburetor (with “fluid-holding device” subsuming carburetor), cell-heart (with “organ” subsuming “heart”), feather-canary (with “bird” subsuming “canary”).

Now, take the model depicted in figure 7 below. There are two potential parthood relations **A** and **B**. The relation **A** between Mitral Valve and Musician holds iff transitivity holds across (Mitral Valve  $\xrightarrow{d_1}$  Human Heart) and (Human Heart  $\xrightarrow{i_1}$  Musician), since in the other reading of these relations, i.e., (Mitral Valve  $\xrightarrow{d_2}$  Human Heart) and (Human Heart  $\xrightarrow{d_1}$  Musician), transitivity is already guaranteed by theorem (T2). To put it baldly, relation **A** is transitive in this case iff  $d_1(x, X, y, Y) \wedge i_1(y, Y, z, Z) \rightarrow i_1(x, X, z, Z)$  is a theorem. Likewise, relation **B** is transitive in this case iff  $i_1(x, X, y, Y) \wedge d_1(y, Y, z, Z) \rightarrow i_1(x, X, z, Z)$  is a theorem. As we show in the sequel,  $d_1(x, X, y, Y) \wedge i_1(y, Y, z, Z) \rightarrow d_1(x, X, z, Z) \vee i_1(x, X, z, Z)$  is a theorem (T3) while  $i_1(x, X, y, Y) \wedge d_1(y, Y, z, Z) \rightarrow d_1(x, X, z, Z) \vee i_1(x, X, z, Z)$  is not. Therefore, whilst **A** is a case of indirect functional parthood between Mitral Valve and Musician, relation **B** is not warranted and, hence, must not exist in figure 7.

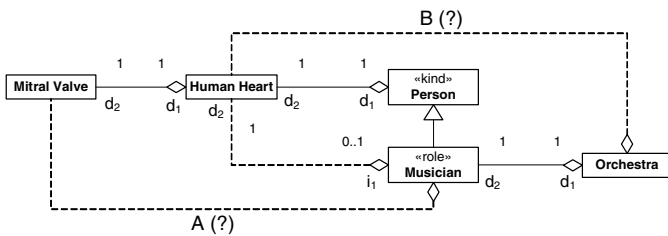


Fig. 7. Two candidate parthood relations due to transitivity

**(T3) Theorem 3:**  $d_1(x, X, y, Y) \wedge i_1(y, Y, z, Z) \rightarrow i_1(x, X, z, Z)$

**Proof:**

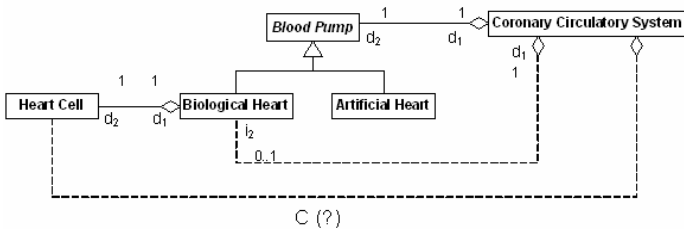
- |  |                 |
|--|-----------------|
| (1). $d_1(x, X, y, Y)$   | T3              |
| (2). $i_1(y, Y, z, Z)$   | T3              |
| (3). $(x < y) \wedge \text{IFD}(x, X, y, Y)$   | 1, Definition 1 |
| (4). $\text{GFD}(X, Y) \wedge x::X \wedge y::Y \wedge (F(x, X) \rightarrow F(y, Y))$ | 3, (6)          |

- (5).  $(y < z) \wedge \text{IIFD}(y, Y, z, Z)$  2, Definition 3
- (6).  $z::Z \wedge \exists W (\text{Subtype}(Z, W) \wedge \text{IFD}(y, Y, z, W))$  5, (7)
- (7).  $\text{GFD}(Y, W) \wedge y::Y \wedge z::W \wedge (F(y, Y) \rightarrow F(z, W))$  6, (6)
- (8).  $(x < z)$  3,5,TFP
- (9).  $\text{GFD}(X, W)$  4,7, (5)
- (10).  $(x::X) \wedge (z::W)$  4,7, EC
- (11).  $(F(x, X) \rightarrow F(z, W))$  4,7,TLI
- (12).  $\text{GFD}(X, W) \wedge (x::X) \wedge (z::W) (F(x, X) \rightarrow F(z, W))$  9,10,11,IC
- (13).  $\text{IFD}(x, X, z, W)$  12, (6)
- (14).  $z::Z \wedge \exists W (\text{Subtype}(Z, W) \wedge \text{IFD}(x, X, z, W))$  6,13,IC
- (15).  $\text{IIFD}(x, X, z, Z)$  14, (7)
- (16).  $(x < z) \wedge \text{IIFD}(x, X, z, Z)$  8,15, IC
- (17).  $i_1(x, X, z, Z)$  16, Definition 3 □

Let us now modify the model of figure 7 to depict a more realistic conceptualization. In this modified specification (figure 8) we have that every Blood Pump is part of a Circulatory System and necessitates a Circulatory System in order to work as such ( $d_1$ ). Likewise, every Circulatory System has as part a Blood Pump and necessitates the latter to work as such ( $d_2$ ). As any Blood Pump, a Biological Heart is part of a Circulatory System and necessitates a Circulatory System to work as such, i.e., *direct functional dependence* (2) is inherited by Biological Heart from the subsuming universal. The same obviously holds for Artificial Heart. However, it is not the case that a Circulatory System is directly functionally dependent of a Biological Heart specifically. To put it in an alternative way, a Circulatory System, in order to function as such, relies on the behavior of a Blood Pump, but this behavior does not have to be afforded in the specific way a Biological Heart does. In [2], this type of relationship between Biological Heart and Circulatory System is termed *indirect functional parthood* (2) and it is defined as follows:

**Definition 4 (Indirect Functional Part of type 2):**  $i_2(x, X, y, Y) \equiv (x < y) \wedge \text{IIFD}(y, Y, x, X)$ . ■

Examples of  $i_2$  include heart-circulatory system (with “blood pump” subsuming “heart”), brick-wall (with “construction material” subsuming “brick”).



**Fig. 8.** Example of an indirect functional parthood of type 2 (from Biological Heart to Coronary Circulatory System) and of a candidate parthood relationship (C) due to transitivity

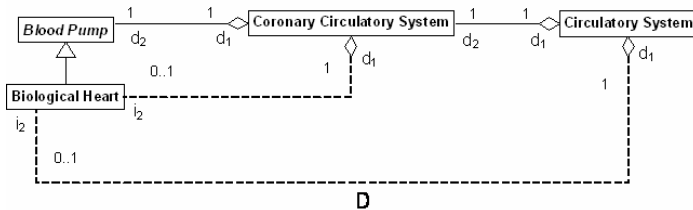
Once more, we have the question: does transitivity hold across (Heart Cell  $\xrightarrow{d2}$  Biological Heart) and (Biological Heart  $\xrightarrow{i2}$  Coronary Circulatory System)? In the other reading we have (Heart Cell Valve  $\xrightarrow{d1}$  Biological Heart) and (Biological Heart  $\xrightarrow{d1}$  Coronary Circulatory System), thus, relation **C** is warranted iff the question above is answered affirmatively. The answer in this case is negative, since  $d_2(x,X,y,Y) \wedge i_2(y,Y,z,Z) \rightarrow d_2(x,X,z,Z) \vee i_2(x,X,z,Z)$  cannot be shown to be a theorem in this theory. However, the following is a theorem:

**(T4) Theorem 4:**  $i_2(x,X,y,Y) \wedge d_2(y,Y,z,Z) \rightarrow i_2(x,X,z,Z)$

**Proof:**

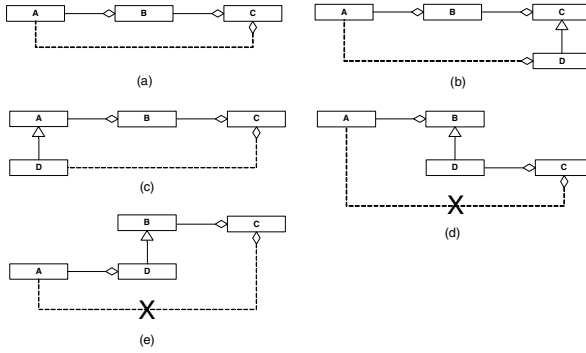
- (15).  $i_2(x,X,y,Y)$  T4
- (16).  $d_2(y,Y,z,Z)$  T4
- (17).  $(x < y) \wedge \text{IIFD}(y,Y,x,X)$  1, Definition 4
- (18).  $(y < z) \wedge \text{IFD}(z,Z,y,Y)$  2, Definition 2
- (19).  $x::X \wedge \exists W (\text{Subtype}(X,W) \wedge \text{IFD}(y,Y,x,W))$  3, (7)
- (20).  $\text{GFD}(Y,W) \wedge y::Y \wedge x::W \wedge (F(y,Y) \rightarrow F(x,W))$  5, (6)
- (21).  $\text{GFD}(Z,Y) \wedge z::Z \wedge y::Y \wedge (F(z,Z) \rightarrow F(y,Y))$  4, (6)
- (22).  $(x < z)$  3,4, TFP
- (23).  $\text{GFD}(Z,W)$  6,7, (5)
- (24).  $(z::Z) \wedge (x::W)$  6,7,EC
- (25).  $(F(z,Z) \rightarrow F(x,W))$  6,7,TLI
- (26).  $\text{GFD}(Z,W) \wedge (z::Z) \wedge (x::W) (F(z,Z) \rightarrow F(x,W))$  9,10,11,IC
- (27).  $\text{IFD}(z,Z,x,W)$  12,(6)
- (28).  $x::X \wedge \exists W (\text{Subtype}(X,W) \wedge \text{IFD}(z,Z,x,W))$  5,13,IC
- (29).  $\text{IIFD}(z,Z,x,X)$  14,(7)
- (30).  $(x < z) \wedge \text{IIFD}(z,Z,x,X)$  8,15,IC
- (31).  $i_2(x,X,z,Z)$  16, Definition 4 □

Due to this theorem we have that the relation **D** between Biological Heart and Circulatory System (depicted in figure 8 below) is warranted, since transitivity holds across (Biological Heart  $\xrightarrow{i2}$  Coronary Circulatory System) and (Coronary Circulatory System  $\xrightarrow{d2}$  Circulatory System) in this case.



**Fig. 8.** Example of an indirect functional parthood of type 2 due to transitivity (from Biological Heart to Circulatory System)





**Fig. 9.** The patterns of figures (a-c) represent cases in which a derived functional transitive parthood relation can be inferred. Intransitive cases are shown in figures (d) and (e).

We conclude this section by providing the following set of visual patterns that can isolate the scope of transitivity in conceptual models containing parthood relations between functional complexes (functional parthood). Transitivity can be guaranteed for these relations only in cases where the patterns of figures (9.a-c) occur. In summary, parthood relations between concrete functional complexes are neither transitive nor intransitive, but non-transitive relation (i.e., transitive in certain cases and intransitive in others). One of the main contributions of this paper is to provide a systematic engineering tool based on a solid theory to exactly inform the modeler which are the cases in which transitivity hold.

### 5 Final Considerations

The work presented here is part of a series of publications (e.g., [4,6,8]) in which we make use of Ontological theories for analyzing, re-designing and providing real-world semantics for conceptual modeling languages and models. Here we build on a formal theory of linguistic functional parthood presented in [2], and on an ontological theory of relationships presented [8] to provide a solution to one classical problem in conceptual modeling, namely, deciding on the transitivity of part-whole relations between the most common objects in conceptual models (functional complexes).

Despite being precise and ontologically well-founded, the theory presented here is of a substantial complexity, thus, demanding for its full understanding at least a basic notion of logics and an advanced understanding of formal ontology. For this reason, and with the intent to provide some methodological tools for helping the modeler in employing the results of this theory, we proposed a number of visual patterns that can be directly applied to diagrams to isolate the scope of transitivity of functional part-whole relations. We believe that these results contribute to the task of defining sound engineering tools and principles for the practice of conceptual modeling. It is important to emphasize that these patterns can be used to isolate the contexts of transitivity in a diagram regardless of the content of what is being represented there. As a consequence, fully automated tool support can be built for this task in a relatively simple

way, since the underlying algorithm merely has to check structural properties of the diagram and not the content of involved nodes. We are currently working on the implementation of prototype to do exactly that.

**Acknowledgement.** This work is partly funded by the Infra-Modela (FAPES) project.

## References

1. Winston, M.E., Chaffin, R., Herrman, D.: A taxonomy of part-whole relations. *Cognitive Science* 11, 417–444 (1987)
2. Vieu, L., Aurnague, M.: Part-of Relations, Functionality and Dependence. In: Aurnague, M., Hickmann, M., Vieu, L. (eds.) *Categorization of Spatial Entities in Language and Cognition*. John Benjamins, Amsterdam (2007)
3. Keet, M., Artale, A.: Representing and Reasoning over a Taxonomy of Part-Whole Relations. *Applied Ontology* 3(1-2), 91–110 (2008)
4. Guizzardi, G.: *Ontological Foundations for Structural Conceptual Models*, PhD Thesis, University of Twente, The Netherlands (2005)
5. Gerstl, P., Pribbenow, S.: Midwinters, End Games, and Bodyparts. A Classification of Part-Whole Relations. *Intl. Journal of Human-Computer Studies* 43, 865–889 (1995)
6. Guizzardi, G., Herre, H., Wagner, G.: Towards Ontological Foundations for UML Conceptual Models. In: Meersman, R., Tari, Z., et al. (eds.) *ODBASE 2002*. LNCS, vol. 2519. Springer, Heidelberg (2002)
7. Opdahl, A., Henderson-Sellers, B., Barbier, F.: Ontological Analysis of whole-part relationships in OO-models. *Information and Software Technology* 43, 387–399 (2001)
8. Guizzardi, G., Wagner, G.: What’s in a Relationship: An Ontological Analysis. In: Li, Q., Spaccapietra, S., Yu, E., Olivé, A. (eds.) *ER 2008*. LNCS, vol. 5231. Springer, Heidelberg (2008)
9. Heller, B., Herre, H.: Ontological Categories in GOL. *Axiomathes* 14, 71–90 (2004)
10. Thomasson, A.L.: *Ontological Categories and How to Use Them*, The Electronic Journal of Analytic Philosophy, Indiana University, USA (5) (Spring 1997)
11. Wieringa, R.J., de Jonge, W., Spruit, P.A.: Using dynamic classes and role classes to model object migration. *Theory and Practice of Object Systems* 1(1), 61–83 (1995)
12. Steimann, F.: On the representation of roles in object-oriented and conceptual modeling. *Data & Knowledge Engineering* 35, 1 (2000)

# Using UML as a Domain-Specific Modeling Language: A Proposal for Automatic Generation of UML Profiles

Giovanni Giachetti, Beatriz Marín, and Oscar Pastor

Centro de Investigación en Métodos de Producción de Software  
Universidad Politécnica de Valencia  
Camino de Vera s/n  
46022 Valencia, Spain  
{ggiachetti, bmarin, opastor}@pros.upv.es

**Abstract.** Nowadays, there are several MDD approaches that have defined Domain-Specific Modeling Languages (DSML) that are oriented to representing their particular semantics. However, since UML is the standard language for software modeling, many of these MDD approaches are trying to integrate their semantics into UML in order to use UML as DSML. The use of UML profiles is a recommended strategy to perform this integration allowing, among other benefits, the use of the existent UML modeling tools. However, in the literature related to UML profile construction; it is not possible to find a standardized UML profile generation process. Therefore, a process that integrates a DSML into UML through the automatic generation of a UML profile is presented in this paper. This process facilitates the correct use of UML in a MDD context and provides a solution to take advantage of the benefits of UML and DSMLs.

**Keywords:** UML Profile, UML, MDD, DSML.

## 1 Introduction

An appropriate modeling language is one of the most important elements for Model-Driven Development (MDD) approaches [22]. To obtain modeling languages that are adequate, different MDD approaches have defined their own Domain-Specific Modeling Languages (DSML) in order to represent their particular modeling needs. Two of the benefits that the use of DSMLs provide to MDD approaches are: (1) a correct and precise representation of the conceptual constructs related to the application domain, and (2) simplification of the implementation of tools oriented to improving the modeling tasks, development, and maintenance of generated software solutions.

However, since UML is seen as the standard language for software modeling purposes, many MDD approaches are integrating their modeling needs into UML in order to use UML as DSML. To perform this integration, the use of the extension mechanism defined in the UML specification, called *UML profile*, is the most suitable strategy. Therefore, the MDD approaches could achieve a larger market (greater number of potential users), take advantage of the existent UML technologies, and reduce the learning curve [2][12][23]. In addition, UML can be used as a mechanism to interchange ideas and theories among different research communities.

Currently, there are many definitions of UML profiles that are associated to MDD approaches [14]. Generally speaking, these profile definitions are manually elaborated in a straightforward way and without a standardized process because a standard that specifies how the UML extensions must be defined does not currently exist [6]. For this reason, many of the existent UML profiles are invalid or of poor quality [23]. In addition, the manual definition of a UML profile is an error-prone and time-consuming task [24]. These two risk factors (time and error) must be avoided, especially in a MDD industrial context, where time costs money and mistakes in implementation directly impact on customer satisfaction.

To avoid the risks described above, some works related to UML profile elaboration have defined proposals to achieve a semi-automated profile generation [12][24]. For the generation of the UML profile, these proposals use as input the metamodel that describes the *conceptual constructs* related to the DSML of an MDD approach (the *DSML Metamodel*). However, none of these proposals provide a sound solution for the automatic generation of a complete UML profile. This is because, in real MDD solutions, structural differences between the DSML metamodel and the UML Metamodel, which prevent the automated identification of the extensions that must be performed in UML, may be found.

This paper introduces a solution for a completely automated UML profile generation using as input the DSML Metamodel related to a MDD approach. This solution is part of an integration process that has been designed to introduce the modeling needs of MDD approaches into UML. In this process, the proposal presented in [8] is used to obtain a correct input for the generation of the UML profile.

Thus, this paper shows how the required UML extensions can be automatically identified and details the transformation rules to obtain the UML Profile that implements these extensions. This paper also shows the application of the integration process in an industrial MDD approach called *OO-Method* [19][20] in order to exemplify how this process can be used to integrate UML and DSML models in a unique MDD solution.

The rest of the paper is organized as follows: Section 2 shows the background related to UML profile generation. Section 3 introduces the proposed process. Section 4 details the automatic UML profile generation. Section 5 presents the application of the process. Finally, Section 6 presents our conclusions and further work.

## 2 Background

The UML profile extension mechanism is defined in the UML Infrastructure [16]. It defines the mechanisms used to adapt existing metamodels to specific platforms, domains, business objects, or software process modeling. In this work, the UML profiles are used to integrate the modeling needs of MDD approaches in UML [17]. Further details about UML Profiles and UML extensions can be consulted in [2][7][16].

In the literature related to the definition of UML profiles, two main working schemas can be observed: 1) the definition of the UML profile from scratch; and 2) the definition of the UML profile starting from the *DSML Metamodel* [23], which is the metamodel that describes the conceptual constructs required by a MDD approach. For

the process presented in this paper, the second working schema has been selected since it provides a methodological solution that has more automation possibilities.

One of the first proposals related to this working schema is the work presented by Fuentes-Fernández et al. in [7], who propose some basic guidelines for the UML profile definition. In [23], Selic proposes a systematic approach that takes into account the new UML profile extension features. In addition, this systematic approach establishes some guidelines to ensure a correct DSML metamodel specification and defines some criteria to obtain a UML profile by means of a mapping that identifies the equivalences between the DSML metamodel and the UML metamodel.

Nowadays, there are very few works related to the automation of the definition of a UML profile. One of these works is the Lagarde et al. approach [12], which can be partially automated through the identification of a set of specific design patterns. However, this approach requires the manual definition of an initial UML profile skeleton. Another interesting work is presented by Wimmer et al. [24]. This work proposes a semi-automatic approach that introduces a specific language to define the mapping between the DSML metamodel and the UML metamodel. This mapping allows an automated UML profile generation. However, this approach does not support all the possible mapping alternatives, for instance, the mapping M:M (many elements of the DSML metamodel mapped to many elements of the UML metamodel). As a consequence, the effective application in real MDD approaches is not possible.

In general, the analyzed works are only centered on representing those modeling elements of the DSML that do not exist in UML using the generated UML Profile. However, this focus is not enough to generate a correct UML profile because there are other elements that must be considered for a correct UML profile definition. These other elements are: 1) the representation of the differences that exist between elements of the DSML, and corresponding elements that already exist in UML, and 2) the definition of rules oriented to validate the correct use of the UML profile in order to produce correct conceptual models.

Even when these additional considerations are omitted, none of the works mentioned above provide a sound transformation process to automatically generate a complete UML profile solution. The main limitation of these approaches comes from the structural differences between the DSML metamodel and the UML metamodel. If these structural differences are solved, then the UML Profile generation can be automated. In Giachetti et al.[10], we propose a solution to solve these structural problems. This solution consists of the transformation of the DSML metamodel into a new metamodel. This new metamodel provides an adequate input to automate the integration of the abstract syntax that is represented in a DSML metamodel into the UML Metamodel. The automatic UML profile generation that is presented in the next section of this paper is based on this solution.

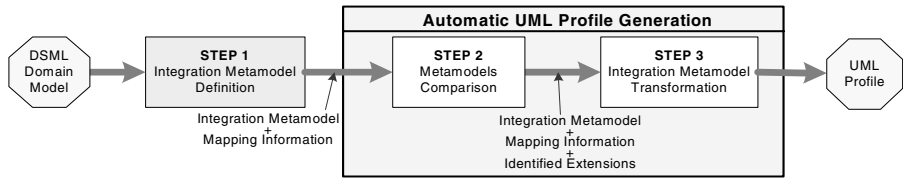
### **3 A Process to Integrate a DSML into UML**

This section presents the process that has been defined to integrate the modeling needs related to a specific MDD approach into UML by means of an automatically generated UML profile. These modeling needs are represented by the DSML

metamodel of the MDD approach. To elaborate this process, different works have been considered. Some of these works are: definition of profiles using DSML metamodels [7][12][23][24], correct use of metamodels in software engineering [11], UML profile implementations [14], interchange between UML and DSMLs [1][9], and new UML profile features introduced in UML [16].

The proposed process can be used in those MDD approaches where the conceptual constructs can be considered as a subset or extension of the UML constructs. This constraint comes from the limitation of the UML profile to change the reference metamodel and guarantees that the MDD approaches that use the proposed process are UML-Compliant. The process is composed of three steps (see Figure 1):

- *Step 1:* Definition of the Integration Metamodel from the DSML metamodel taking into account the UML Metamodel defined in the UML Superstructure [17].
- *Step 2:* Comparison between the Integration Metamodel and the UML Metamodel. This comparison identifies the extensions that must be defined in UML by using the equivalences identified in the *Step 1*.
- *Step 3:* Transformation of the Integration Metamodel according to a set of transformation rules in order to obtain the final UML profile.



**Fig. 1.** Integration Process Schema

The second and third steps of the integration process correspond to the *Automatic UML Profile Generation*. In this process, the original DSML metamodel is redefined to obtain the input required for the UML profile generation. This input is called the *Integration Metamodel*, and its main characteristics are described below.

### 3.1 Definition of the Integration Metamodel

The Integration Metamodel is a special DSML metamodel that has been defined to automate the integration of a DSML into UML. This metamodel is defined from the DSML metamodel, and it represents the same abstract syntax of the original metamodel. The main difference between the Integration Metamodel and the DSML metamodel is its structure since it is defined to obtain a mapping with the UML metamodel, which allows the automatic identification of the required UML extensions. This mapping information is included inside the Integration Metamodel definition. The UML metamodel selected for the definition of the Integration Metamodel is the metamodel presented in the UML Superstructure [17].

The Integration Metamodel is manually defined according to the systematic approach presented in [10]. In that work, the structural problems that can exist in a

DSML metamodel as well as the benefits of the Integration Metamodel are discussed. Summarizing, the Integration Metamodel has the following features:

- It is defined according to the EMOF modeling capabilities, which are defined in the MOF (Meta Object Facility) specification [15]. By using EMOF, the resultant metamodel properties do not have features that are not supported by UML profiles. Moreover, EMOF has a standardized XMI definition [18]. Thus, the UML profile generation can be automated by means of transformation rules that are implemented over the XMI definition of the Integration Metamodel. The EMOF definition also allows the implementation of specific model editors with tools such as Eclipse GMF [4].
- It is mapped to the UML Metamodel taking into account: Classes, Properties (Attributes and Associations), Enumerations, Enumeration Literals, and Data Types. The mapped elements are considered as *equivalent elements*, and the non-mapped elements are considered as *new elements* in the Integration Metamodel. This mapping complies with the following rules:
  - All the classes from the Integration Metamodel are mapped. This assures that the conceptual constructs of the DSML can be represented from the conceptual constructs of UML.
  - The mapping is defined between elements of the same type (classes with classes, attributes with attributes, and so on).
  - An element from the Integration Metamodel is only mapped to one element of the UML Metamodel. This rule also considers the possibility of have X:1 mappings ( $X \geq 0$ ); for instance, many classes of the Integration Metamodel can be mapped to one class of UML. In this example, the mapping rule is also accomplished because each class of the Integration Metamodel is only mapped to one UML class. It is important to note that the many-to-many mappings that may exist between the original DSML metamodel and the UML Metamodel are transformed into X:1 mappings during the generation of the Integration Metamodel.
  - If the properties (attributes and associations) of a class from the Integration Metamodel are mapped to properties of a UML class, then the class that owns the properties is mapped to this UML class (or a generalization of it).

## 4 Automatic Generation of the UML Profile

This section presents how a correct UML profile can be automatically generated from an Integration Metamodel. This automatic generation is comprised by two steps: 1) the comparison of metamodels to obtain the required UML extensions; and 2) The transformation of the Integration Metamodel into the corresponding UML profile. These steps are presented below.

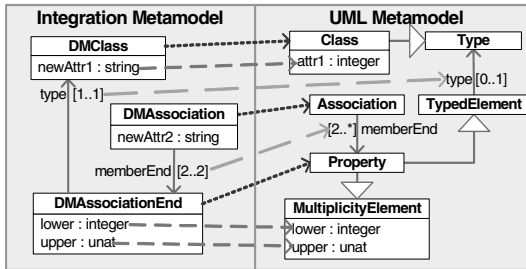
### 4.1 Comparison of Metamodels

The identification of the required UML extensions is performed through a comparison between the Integration Metamodel and the UML Metamodel. To perform this comparison, the mapping information defined in the Integration Metamodel is used.

The comparison between the Integration Metamodel and the UML Metamodel considers:

- The identification of *new elements*, which are the elements from the Integration Metamodel that are not equivalent to UML elements. These elements can be attributes, associations, enumerations, literal values, and data types.
- The identification of differences in type or cardinality of *equivalent properties* (attributes and associations).

Figure 2 shows an example of an Integration Metamodel that will be used to help understand how the UML extensions are identified. The metamodel presented in this figure represents a binary association between classes. In this metamodel, the attributes of the class *DMAssociationEnd* represent the cardinality related to each association end, and the attributes of the classes *DMClass* and *DMAssociation* represent generic attributes related to these classes.



**Fig. 2.** Integration Metamodel related to a binary association between classes

Table 1 shows the comparison result obtained from the Integration Metamodel presented in Figure 2. In this table, the column *Integration Metamodel* shows the *new elements* identified, or *equivalent elements* that differ in relation to the related UML elements. The *Difference* column shows what the differences are by indicating (when necessary) the values for the Integration Metamodel element (*I.M.*) and the UML element (*UML*).

**Table 1.** Metamodel comparison for the Integration Metamodel presented in Figure 2

Integration Metamodel	Difference
DMClass.newAttr1	Different type: I.M. = string; UML = integer
DMAssociation.memberEnd	Different upper bound: I.M. = 2; UML = *
DMAssociation.newAttr2	New attribute
DMAssociationEnd.type	Different lower bound: I.M. = 1; UML = 0 Different type: I.M. = DMClass; UML = Type



The mapping information defined in the Integration Metamodel allows the identification of type differences. For instance, in the case of *DMAssociationEnd.type* and *Property.type*, the type is different because *DMClass* is not equivalent to *Type*.

The cardinality differences are identified by analyzing the lower and upper bound of the equivalent properties and the referenced UML properties. This is the case of the equivalent properties *DMAssociation.memberEnd* and *DMAssociationEnd.type*.

Finally, the differences identified in the comparison are the extensions that must be introduced into the UML in order to correctly represent the modeling needs of the related MDD approach.

### 4.2 Transformation of the Integration Metamodel

The third step in the process defines a set of transformation rules to automatically generate a complete UML profile from the Integration Metamodel and the UML extensions previously obtained. These transformation rules are defined considering that the *new elements* and the differences between *equivalent elements* identified during the metamodel comparison must be represented in the generated UML profile. In addition, these rules take into account the automatic generation of the needed constraints in order to assure the correct application of the generated extensions.

In order to show how the Integration Metamodel can be transformed into the corresponding UML profile, the required transformation rules are described below. These transformation rules are separated by the different EMOF conceptual constructs. The possible modeling situations are analyzed for each construct, according to the Integration Metamodel features. A figure that exemplifies the application of the transformation rules in a generic way is also presented.

#### Classes

Rule 1: One Stereotype for each *equivalent class*. The stereotype extends the referenced UML class, and its name is equal to the *equivalent class* name. Figure 1 exemplifies this rule.

This first transformation rule is the most relevant because it involves the generation of the stereotypes, which are the main constructs of the UML profile. The rest of transformation rules are applied according to the results obtained by this first rule.

Validation: At the end of the UML profile generation, if there is only one stereotype that extends a UML class, then the stereotype extension must be defined as *required*. This constraint is defined because, in the DSML context, the UML class only has the semantics of the involved equivalent class (see *Class3* in Figure 3).

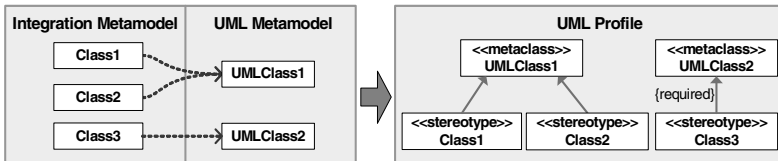


Fig. 3. Generic example for the transformation rule related to classes: Rule 1

## Properties

In EMOF, the properties represent attributes of a class (metaclass) or references (associations) between the classes. The main difference between an attribute and an association is that an attribute represents a data-valued property, while an association is an object-valued property. In other words, in an association, the type is given by another class of the model that represents the related class. These differences are taken into account in the definition of the involved transformation rules.

Rule 2: One tagged value for each *new property*. The tagged value must have the same type and cardinality as the *new property*. The name of the tagged value must be the name of the *new property*. In the case of an association, the tagged value must have the same aggregation kind as the *new property*. The application of this rule can be observed in Figure 4 for the association *Class1.rolClass2*.

Rule 3: One OCL constraint if the lower bound of an *equivalent property* is higher than the lower bound of the referenced UML property:

```
self.[property]->size() >= [newLowerBound]
```

Rule 4: One OCL constraint if the upper bound of an *equivalent property* is lower than the upper bound of the referenced UML property:

```
self.[property]->size() <= [newUpperBound]
```

As Figure 4 shows, rules 3 and 4 are applied to the *Class2.roleClass3* and *Class3.roleClass1*, respectively.

Validation: For rules 3 and 4, an OCL constraint is defined to validate that the corresponding stereotype is applied each time that the involved UML association is established. Thus, the type of the referenced UML association is restricted to the stereotype that represents the type of the *equivalent association*:

```
self.[equivalentAssociation]->isStereotyped1 ([newType])
```

This validation is also applied if the type of an *equivalent association* is changed by a specialization of the original type (see *Class2.rolClass3* in Figure 4).

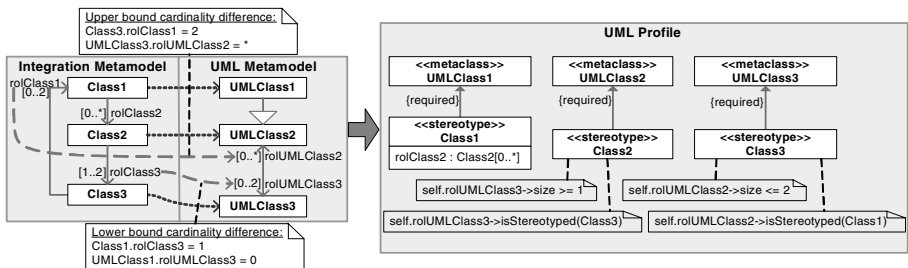


Fig. 4. Generic example for the transformation rules 2 to 4

<sup>1</sup> The OCL operation *isStereotyped* is not part of the OMG specification and is only used to simplify the OCL rules presented. In the application of the integration process, this operation must be implemented according to the target UML tool.

Even though an extension relationship represents a refinement of a class in a way similar to a generalization relationship, its semantics is represented as a special kind of association and not as a generalization. For this reason, a tagged value cannot redefine UML properties. Therefore, when the differences that exist between an *equivalent property* and the referenced UML property cannot be represented using OCL constraints, a tagged value that replaces the original UML property is created. In this case, the MDD process must only consider the tagged value and not the UML property.

Rule 5: One tagged value that replaces a UML property when one of the following conditions holds:

- The type of *equivalent property* is different than the type of the referenced UML property, and the new type is not a specialization of the original type or a stereotype that extends the original type (see *Class1.attr3* in Figure 5).
- The upper bound of the *equivalent property* is higher than the upper bound of the referenced UML property (see *Class1.attr2* in Figure 5).
- The lower bound of the *equivalent property* is lower than the lower bound of the referenced UML property (see *Class1.attr1* in Figure 5).

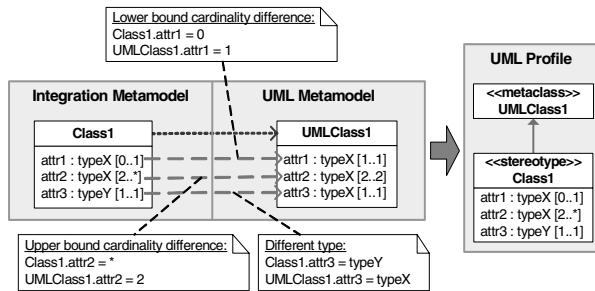


Fig. 5. Generic example for transformation rule 5

### Enumerations

The enumerations are used to specify a customized set of values that can be represented by an attribute of a class. Graphically, the enumerations are represented as a class. However, the enumeration is a specialization of a *Classifier* and not of a *Class*. This difference is considered in the following transformation rule.

Rule 6: One enumeration for each *new enumeration* or *equivalent enumeration* with new literal values. In the case of an *equivalent enumeration*, the generated enumeration replaces the original UML enumeration, and the involved *equivalent attributes* are considered as *new attributes* (Rule 2). In this case, since the UML enumeration is not a class, it cannot be extended with a stereotype in order to include the new literal values. Figure 6 shows the application of this rule for *Enum2* (*equivalent enumeration*) and *Enum3* (*new enumeration*).

Validation: One OCL constraint for each attribute whose type corresponds to an *equivalent enumeration* that has fewer alternatives (literal values) than the referenced UML enumeration (see *Class1.attr1* and *Enum1* in Figure 6).

```
self.[attribute] <> #[nonMappedLiteralValue]
```

This constraint avoids the use of invalid alternatives (non-referenced literal values) that are defined in the referenced UML enumeration.

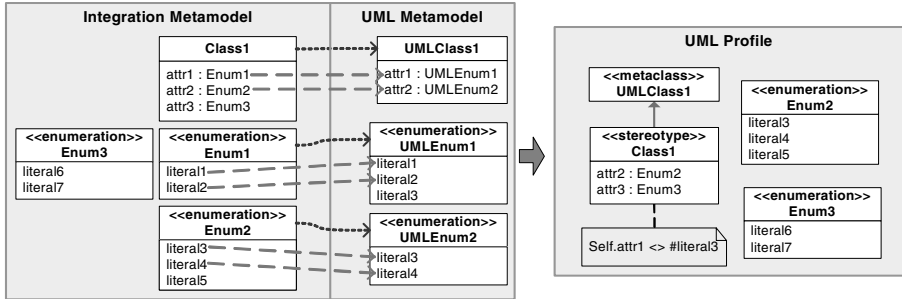


Fig. 6. Generic example for transformation rule 6.

## Generalizations

The generalization relationships have interesting features that must be considered in the generation of the related stereotypes. Two of the main features that must be considered are: 1) since stereotypes are a special kind of class, it is possible to define a generalization between stereotypes; and 2) since the extension association between a stereotype and its related class is a specialization of *Association*, the extension relationship can be inherited.

Rule 7: Define one generalization between two stereotypes that represent *equivalent classes* that are associated with a *new generalization* and that are referencing the same UML class. The extension related to the child stereotype is not defined since it is implicit in the generalization relationship. Figure 7 shows the application of this rule for the generalization defined between the classes *Class1* and *Class3*.

Rule 8: If there is a *new generalization* between two *equivalent classes* that are referencing different UML classes, the generalization relationship is not represented in the UML profile. In this case, the extensions of each stereotype to the corresponding UML class are defined, and the inherited properties (attributes and associations) are duplicated (see the generalization between classes *Class3* and *Class4* in Figure 7). If the generalization is represented, then the child stereotype will be able to extend the UML class that is extended by the father stereotype. This could produce a modeling error since, according to the mapping information, the child stereotype is referencing (extends) a different UML class.

Rule 9: If there is an *equivalent generalization* between two *equivalent classes*, the generalization relationship is not represented in the UML profile, and only the

extensions of each stereotype are defined to the corresponding UML class. In this way, the generalization defined in UML is used instead of the *equivalent generalization* (see the generalization between classes *Class1* and *Class2* in Figure 7).

Note that an *equivalent generalization* represents a generalization that already exists in the UML Metamodel. The *equivalent generalizations* are automatically identified through the participant classes of the Integration metamodel that are equivalent to the classes that participate in the UML generalization.

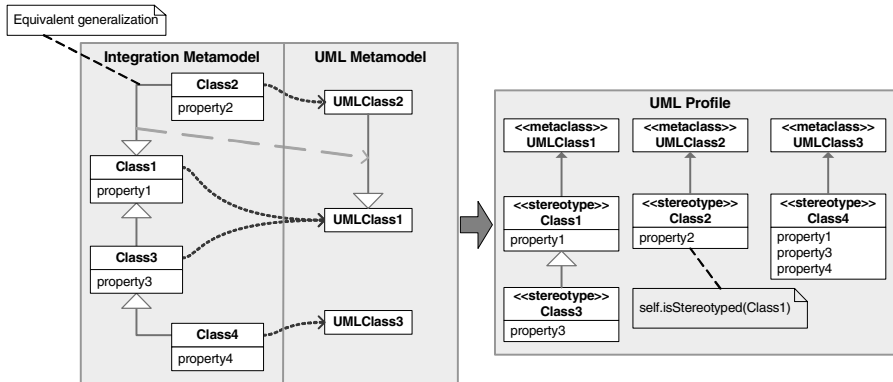


Fig. 7. Generic example for transformation rules 7, 8 and 9

## OCL Rules

The OCL rules defined in the Integration Metamodel manage the interactions between the different constructs of the DSML. Therefore, these rules must be included in the generated UML profile.

Rule 10: The OCL rules defined in the classes of the Integration Metamodel must be included in the stereotypes generated from these classes. The elements referenced in the rules must be changed by the corresponding UML classes and stereotypes.

## Data Types

Rule 11: The *new data types* defined in the Integration Metamodel are defined in a separate model library. This model library must be imported in each UML model that is designed using the generated UML profile.

The equivalent data types that have differences in relation to the referenced UML data types are considered as *new data types*. Since the data types are classifiers, they cannot be extended using stereotypes.

Validation: The UML data types that are not referenced by *equivalent data types* are not valid in the DSML context. For this reason, one OCL constraint is defined over the UML metaclass *TypedElement* to restrict the invalid UML data types:

```
self.type->oclIsTypeOf([invalidType]) = False
```

Rule 11 is the last rule defined for the transformation of the Integration Metamodel in the equivalent UML profile. Figure 8 presents the UML profile obtained after applying the proposed transformation rules to the example Integration Metamodel presented in Figure 2.

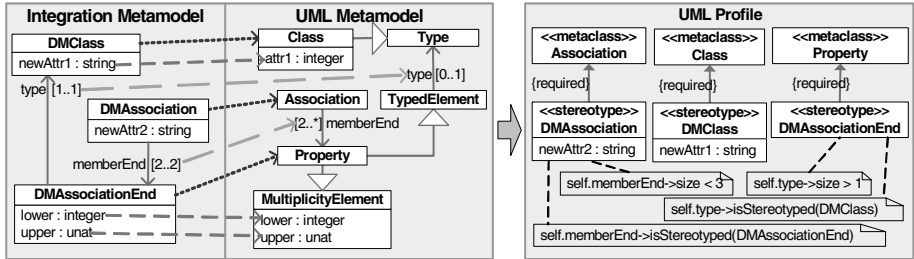


Fig. 8. UML profile generated for the example Integration Metamodel

In addition to the UML profile, the transformation of the Integration Metamodel also generates a new mapping that takes into account the generated UML profile elements (stereotypes, tagged values, etc.). This new mapping provides bidirectional equivalence between the Integration Metamodel and the UML Metamodel (extended with the generated UML profile). Figure 9 shows this new mapping information for the UML profile presented in Figure 8.

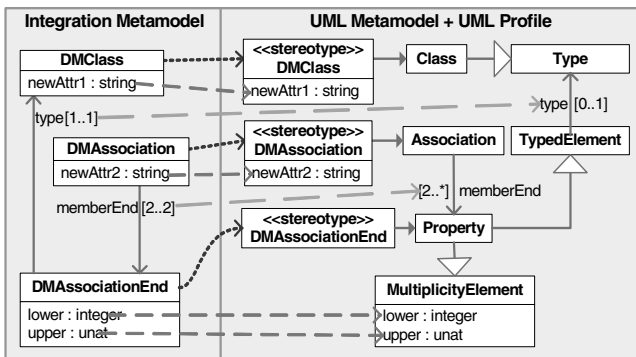


Fig. 9. New mapping information generated for the UML profile presented in Figure 8

The generated UML profile together with the new mapping definition can be used to interchange UML models and DSML models [9], in order to take advantage of these two modeling solutions. The following section shows how the proposed integration process has been applied to an industrial MDD approach [3], to integrate UML tools and the existent DSML-based tools.

## 5 Applying the Integration Process

In this section, the implementation schema used to apply the proposed integration process in the OO-Method industrial approach [19][20] is introduced. This schema (Figure 10) takes advantage of UML tools, without losing the benefits of the existent MDD technology based on the OO-Method DSML.

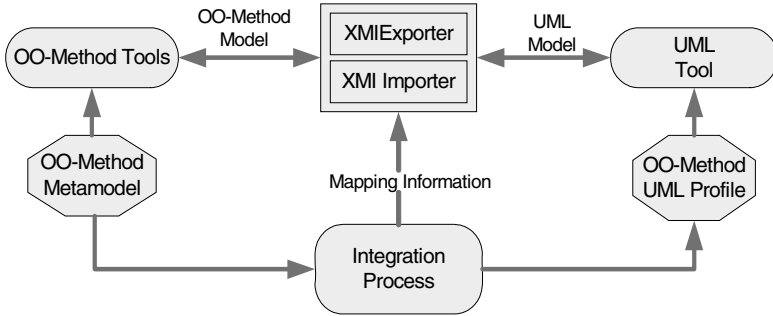


Fig. 10. Schema designed to apply the integration process into the OO-Method approach

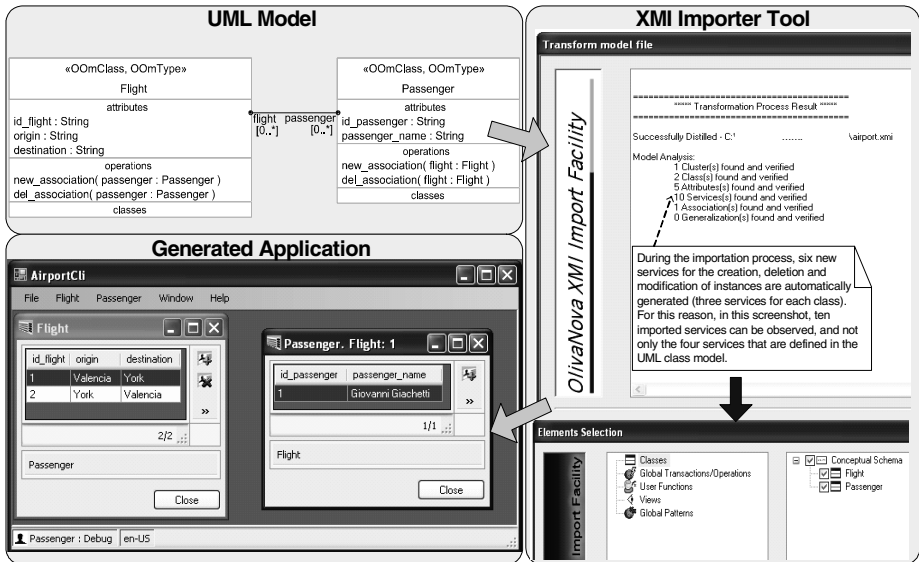


Fig. 11. Application of the OO-method compilation technology over a UML model

The core of the proposed schema is made up of two interchange tools called XMI importer and XMI exporter [3], which transform the UML models into DSML models, and vice versa. These tools are extended with the new mapping information obtained in the UML profile generation process in order to support the generated UML profile [9]. Figure 11 shows the application of the extended XMI importer tool to

automatically generate an application from a UML model extended with the OO-Method UML profile. This model has been defined using the Eclipse UML2 tool [5].

The schema proposed to apply the integration process in the OO-Method approach has three main benefits:

1. The technology, support, and commercial structure defined for the OO-Method development process can be used in a transparent way for UML users.
2. The different OO-Method tools, such as the OO-Method model compiler [21], and functional size measurement tools [8][13] can be used over UML models.
3. The customers can easily migrate from UML tools to OO-Method tools in order to take advantage of the improved functionalities that the OO-method tools provide for the management of OO-Method conceptual models.

## 6 Conclusions and Further Work

This paper presents a solution for the automatic generation of a UML profile from the metamodel that represents the DSML related to a MDD approach. This automatic generation is applied in an integration process in order to take advantage of the benefits provided by the use of UML and DSML based technologies.

The proposed solution tackles an important topic that has not yet received the required attention: the correct definition of UML profiles for MDD solutions. Even though the number of UML profile solutions has increased, the number of publications related to a correct UML profile definition is very limited [23]. In order to obtain this correct definition, the proposed transformations are focused on three main elements: 1) the generation of modeling elements defined in the DSML that are not present in UML; 2) the management of differences that could exist between elements of the DSML that are equivalent with UML elements; and 3) the generation of constraints to assure that the application of the generated UML profile follows the DSML specification.

It is important to note that the transformation rules defined in this paper are just one possible solution for the complete generation of a correct UML profile. Variations of these transformation rules can be defined depending on different design decisions. This paper also explains how this solution can be applied in MDD approaches, taking as example the application performed for the OO-Method approach in order to integrate UML tools and the existent OO-Method tools.

As further work, we plan to finish the implementation of a set of open-source tools that support the proposed integration process in order to provide a generic integration solution for different MDD approaches.

**Acknowledgments.** This work has been developed with the support of MEC under the project SESAMO TIN2007-62894 and co financed by FEDER.

## References

1. Abouzahra, A., Bézivin, J., Fabro, M.D.D., Jouault, F.: A Practical Approach to Bridging Domain Specific Languages with UML profiles. In: Best Practices for Model Driven Software Development (OOPSLA 2005) (2005)
2. Bruck, J., Hussey, K.: Customizing UML: Which Technique is Right for You? IBM (2007)



3. CARE-Technologies, <http://www.care-t.com/>
4. Eclipse: Graphical Modeling Framework Project, <http://www.eclipse.org/gmf/>
5. Eclipse: UML2 Project, <http://www.eclipse.org/uml2/>
6. France, R.B., Ghosh, S., Dinh-Trong, T., Solberg, A.: Model-driven development using uml 2.0: Promises and pitfalls. *IEEE Computer* 39(2), 59–66 (2006)
7. Fuentes-Fernández, L., Vallecillo, A.: An Introduction to UML Profiles. *The European Journal for the Informatics Professional (UPGRADE)* 5(2), 5–13 (2004)
8. Giachetti, G., Marín, B., Condori-Fernández, N., Molina, J.C.: Updating OO-Method Function Points. In: 6th IEEE International Conference on the Quality of Information and Communications Technology (QUATIC 2007), pp. 55–64 (2007)
9. Giachetti, G., Marín, B., Pastor, O.: Using UML Profiles to Interchange DSML and UML Models. In: Third International Conference on Research Challenges in Information Science, RCIS (2009)
10. Giachetti, G., Valverde, F., Pastor, O.: Improving Automatic UML2 Profile Generation for MDA Industrial Development. In: Song, I.-Y., et al. (eds.) *ER Workshops 2008*. LNCS, vol. 5232, pp. 113–122. Springer, Heidelberg (2008)
11. Henderson-Sellers, B.: On the Challenges of Correctly Using Metamodels in Software Engineering. In: 6th Conference on Software Methodologies, Tools, and Techniques (SoMeT), pp. 3–35 (2007)
12. Lagarde, F., Espinoza, H., Terrier, F., Gérard, S.: Improving UML Profile Design Practices by Leveraging Conceptual Domain Models. In: 22nd IEEE/ACM International Conference on Automated Software Engineering (ASE), pp. 445–448 (2007)
13. Marín, B., Giachetti, G., Pastor, O.: Automating the Measurement of Functional Size of Conceptual Models in an MDA Environment. In: Jedlitschka, A., Salo, O. (eds.) *PROFES 2008*. LNCS, vol. 5089, pp. 215–229. Springer, Heidelberg (2008)
14. OMG: Catalog of UML Profile Specifications
15. OMG: MOF 2.0 Core Specification
16. OMG: UML 2.1.2 Infrastructure Specification
17. OMG: UML 2.1.2 Superstructure Specification
18. OMG: XMI 2.1.1 Specification
19. Pastor, O., Gómez, J., Infrán, E., Pelechano, V.: The OO-Method Approach for Information Systems Modelling: From Object-Oriented Conceptual Modeling to Automated Programming. *Information Systems* 26(7), 507–534 (2001)
20. Pastor, O., Molina, J.C.: *Model-Driven Architecture in Practice: A Software Production Environment Based on Conceptual Modeling*, 1st edn. Springer, New York (2007)
21. Pastor, O., Molina, J.C., Iborra, E.: Automated production of fully functional applications with OlivaNova Model Execution. *ERCIM News* 57 (2004)
22. Selic, B.: The Pragmatics of Model-Driven Development. *IEEE Software* 20(5), 19–25 (2003)
23. Selic, B.: A Systematic Approach to Domain-Specific Language Design Using UML. In: 10th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC), pp. 2–9 (2007)
24. Wimmer, M., Schauerhuber, A., Strommer, M., Schwinger, W., Kappel, G.: A Semi-automatic Approach for Bridging DSLs with UML. In: 7th OOPSLA Workshop on Domain-Specific Modeling (DSM), pp. 97–104 (2007)

# Verifying Action Semantics Specifications in UML Behavioral Models

Elena Planas<sup>1</sup>, Jordi Cabot<sup>1</sup>, and Cristina Gómez<sup>2</sup>

<sup>1</sup> Estudis d'Informàtica, Multimèdia i Telecomunicacions, Universitat Oberta de Catalunya  
{eplanash, jcabot}@uoc.edu

<sup>2</sup> Dept. de Llenguatges i Sistemes Informàtics, Universitat Politècnica de Catalunya  
cristina@lsi.upc.edu

**Abstract.** MDD and MDA approaches require capturing the behavior of UML models in sufficient detail so that the models can be automatically implemented/executed in the production environment. With this purpose, Action Semantics (AS) were added to the UML specification as the fundamental unit of behavior specification. Actions are the basis for defining the fine-grained behavior of operations, activity diagrams, interaction diagrams and state machines. Unfortunately, current proposals devoted to the verification of behavioral schemas tend to skip the analysis of the actions they may include. The main goal of this paper is to cover this gap by presenting several techniques aimed at verifying AS specifications. Our techniques are based on the static analysis of the dependencies between the different actions included in the behavioral schema. For incorrect specifications, our method returns a meaningful feedback that helps repairing the inconsistency.

## 1 Introduction

One of the most challenging and long-standing goals in software engineering is the complete and automatic implementation of software systems from their initial high-level models [21]. This is also the focus of current MDD (Model-driven development) and MDA (Model-driven architecture) approaches.

Recently, the OMG itself has issued a RFP for the “Foundational Subset for Executable UML Models” [20], with the goal of reducing the expressivity of the UML to a subset that can be directly executable [17]. A key element in all executable UML methods is the use of Action Semantics (AS) to specify the fine-grained behavior of all behavioral elements in the model. Actions are the fundamental unit of behavior specifications. Their resolution and expressive power are comparable to the executable instructions in traditional programming languages. Higher-level behavioral formalisms of UML (as operations, activity diagrams, state machines and interactions diagrams) are defined as an additional layer on top of the predefined set of basic actions (e.g. creation of new objects, removals of existing objects, among others) [19].

Given the important role that actions play in the specification of the behavioral aspects of a software system, it is clear that their correctness has a direct effect on the quality of the final system implementation. As an example, consider the class diagram

of Fig. 1 including the operations *changeAddress* and *addPerson*. Both operations are incorrect, since *changeAddress* tries to update an attribute which does not even exist in the class diagram and *addPerson* can never be successfully executed (i.e. every time we try to execute *addPerson* the new system state violates the minimum ‘1’ cardinality constraint of the *department* role in *WorksIn*, since the created person instance *p* is not linked to any department). Besides, this operation set is not complete, i.e. through these operations users cannot modify all elements of the class diagram, e.g. it is not possible to create and destroy departments. These errors must be fixed before attempting to generate the system implementation.

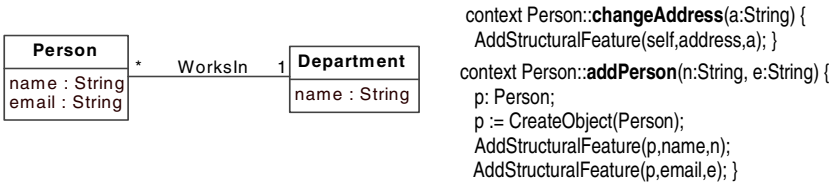


Fig. 1. A simple example of a class diagram with two operations

The main goal of this paper is to provide a set of lightweight techniques for the verification of correctness properties (*syntactic correctness*, *weak executability* and *completeness*) of action-based behavior specifications at design time. Due to space limitations, we will focus on the verification of AS specifications used to define the effect of the operations included in the class diagram (as the example above). Nevertheless, the techniques presented herein could be similarly used to verify action sequences appearing in other kinds of UML behavior specifications.

Roughly, given an operation *op*, our method (see Fig. 2) proceeds by first, analyzing the syntactic correctness of each action  $ac \in op$ . Then, the method analyzes *op* to determine all its possible execution paths. Executability of each path *p* is determined by performing a static analysis of the dependencies among the actions in *p* and their relationship with the structural constraints (as cardinality constraints) in the class diagram. Next, our method analyses the completeness of the whole operation set. For each detected error, possible corrective procedures are suggested to the designer as a complementary feedback. After our initial analysis, model-checking based techniques could also be used to get more information (e.g. incorrect execution traces) on the operations.

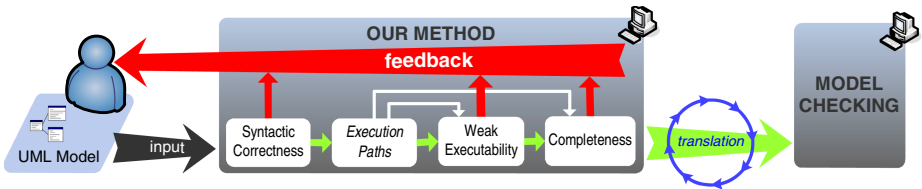


Fig. 2. Method overview

The rest of the paper is structured as follows. The next section describes basic AS concepts. Section 3 focuses on the operations' syntactic correctness. Section 4 explains how to determine the different execution paths in an operation and Section 5 determines their executability. Section 6 study the completeness of an operation set. In Section 7, we compare our method with the related work and, in Section 8, we present the conclusions and further work.

## 2 Action Semantics in the UML

The UML standard [19] defines the actions that can be used in behavioral specifications. In this paper, we will focus on the following write actions<sup>1</sup> (actions that modify the system state) since they are the ones that can compromise the system consistency:

1. *CreateObject(class:Classifier):InstanceSpecification*: Creates a new object that conforms to the specified classifier. The object is returned as an output parameter.
2. *DestroyObject(o:InstanceSpecification)*: Destroys the object  $o$ . We assume that links in which  $o$  participates are not automatically destroyed.
3. *AddStructuralFeature(o:InstanceSpecification, at:StructuralFeature, v:ValueSpecification)*: Sets the value  $v$  as the new value for the attribute  $at$  of the object  $o$ . We assume that multi-valued attributes are expressed (and analyzed) as binary associations between the class and the attribute data type.
4. *CreateLink(as:Association, p<sub>1</sub>:Property, o<sub>1</sub>:InstanceSpecification, p<sub>2</sub>:Property, o<sub>2</sub>:InstanceSpecification)*: Creates a new link in the binary association  $as$  between objects  $o_1$  and  $o_2$ , playing the roles  $p_1$  and  $p_2$ , respectively.
5. *DestroyLink(as:Association, p<sub>1</sub>:Property, o<sub>1</sub>:InstanceSpecification, p<sub>2</sub>:Property, o<sub>2</sub>:InstanceSpecification)*: Destroys the link between objects  $o_1$  and  $o_2$  from  $as$ .
6. *ReclassifyObject(o:InstanceSpecification, newClass:Classifier[0..\*], oldClass:Classifier[0..\*])*: Adds  $o$  as a new instance of classes in  $newClass$  and removes it from classes in  $oldClass$ . We consider that classes in  $newClass$  may only be direct superclasses or subclasses of classes in  $oldClass$ .
7. *CallOperation(op:Operation, o:InstanceSpecification, arguments:List(LiteralSpecification))*: *List(LiteralSpecification)*: Invokes  $op$  on  $o$  with the  $arguments$  values and returns the results of the invocation.

These actions can be accompanied with several read actions (e.g. to access the values of attributes and links of the objects). Read actions do not require further treatment since they do not affect the correctness properties we define in this paper.

Additionally, UML defines that actions can be structured to coordinate basic actions in action sequences, conditional blocks or loops (*do-while* or *while-do* loops).

As an example, we have defined three operations: *endOfReview*, *submitPaper* and *dismiss* (Fig. 4) for the class diagram of Fig. 3, aimed at representing part of a conference management system. The first operation reclassifies a paper as rejected or accepted depending on the *evaluation* parameter. The second one creates a new “under review” paper and links the paper with its authors. The last one deletes the *WorksIn* link between a person and his/her department.

<sup>1</sup> UML provides an abstract syntax for these actions [19]. Our concrete syntax is based on the names of the action metaclasses. For structured nodes we will use an ASL-based syntax [17].

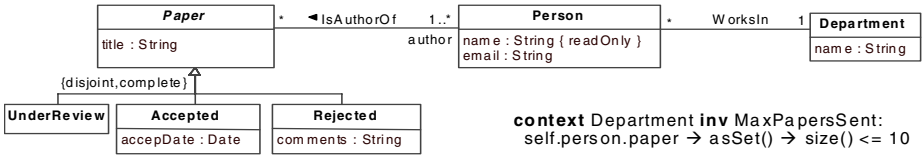


Fig. 3. Excerpt of a conference management system class diagram

```

context Paper::endOfReview(com:String,d:Date,
evaluation:String) {
  if self.oclsTypeOf(UnderReview) then
    if evaluation = 'reject' then
      ReclassifyObject(self,[Rejected],[ ]);
      AddStructuralFeature(self,comments,com);
    else
      ReclassifyObject(self,[Accepted],[ ]);
      AddStructuralFeature(self,accepDate,d);
    endif
  endif
}

context Paper::submitPaper(tit:String, authors:Person[1..*]) {
  i: Integer := 1;
  p: Paper;
  p := CreateObject(UnderReview);
  AddStructuralFeature(p,title,tit);
  while i <= authors->size() do
    CreateLink(IsAuthorOf,author,authors[i],paper,p);
    i := i+1;
  endwhile
}

context Person::dismiss() {
  DestroyLink(WorksIn,person,self,department,self.department);
}
  
```

Fig. 4. Specification of *endOfReview*, *submitPaper* and *dismiss* operations

### 3 Syntactic Correctness

The UML metamodel includes a set of constraints (i.e. well-formedness rules (WFRs)) that restrict the possible set of valid (or well-formed) UML models. Some of these WFRs are aimed at preventing syntactic errors in action specifications. For instance, when specifying a *CreateLink* action *ac* over an association *as*, the WFRs ensure that the type and number of the input objects in *ac* are compatible with the set of association ends defined for *as*.

An operation is syntactically correct when all the actions included in the operation satisfy the WFRs. Unfortunately, our analysis of the WFRs relevant to the Action Packages has revealed several flaws (see the detected errors in [22]). Besides, several required WFRs are missing. For instance, in actions of type *WriteStructuralFeature* we should check that the type of the input object (i.e. the object to be modified) is compatible with the classifier owning the feature (in OCL: *context WriteStructuralFeature inv: self.value.type = self.structuralFeature.type*). Also, in *CreateObject*, the input classifier cannot be the supertype of a covering generalization set (in a covering generalization, instances of the supertype cannot be directly created). Similar WFRs must be defined to restrict the possible *newClassifiers* in the *ReclassifyObject*. For instance, we should check that the *newClassifiers* set and the *oldClassifiers* set are disjoint sets. Additional rules are needed to check that values of *readOnly* attributes are not updated after their initial value has been assigned and so forth. These WFRs must be added to the UML metamodel to ensure the syntactic correctness of action specifications.

After this initial syntactic analysis, we proceed next with a more semantic verification process that relates the specified actions with other model elements.

## 4 Computing the Execution Paths

The correctness properties that will be presented in the next sections are based on an analysis of the possible *execution paths* allowed by the structured group of actions that define the operation effect. An execution path is a sequence of actions that may be followed during the operation execution. For trivial groups of actions (e.g. with neither conditional nor loop nodes) there is a single execution path but, in general, several ones will exist.

To compute the execution paths, we propose to represent the actions in the operation as a model-based control flow graph (MBCFG), that is, a control flow graph based on the model information instead of on the program code, as traditional control flow graph proposals. MBCFGs have been used to express UML sequence diagrams [9]. Here we adapt this idea to express the control flow of action-based operations.

For the sake of simplicity, we will assume that the group of actions defining the operation behavior is defined as a structured *SequenceNode* (see the metamodel excerpt in Fig. 5) containing an ordered set of *ExecutableNodes*, where each executable node can be either one of the basic modification actions described in Section 2 (other types of actions are skipped since they do not affect the result of our analysis), a *ConditionalNode*, a *LoopNode* or an additional nested *SequenceNode*. We also use two “fake” nodes, an initial node (representing the first instruction in the operation) and a final node (representing the last one). These two nodes do not change the operation effect but help in simplifying the presentation of our MBCFG.

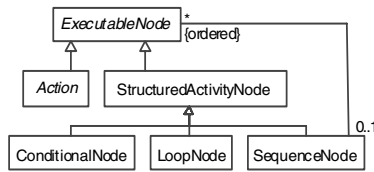


Fig. 5. Fragment of UML metamodel

The digraph  $MBCFG_{op} = (V_{op}, A_{op})$  for an operation  $op$  is obtained as follows:

- Every executable node in  $op$  is a vertex in  $V_{op}$ .
- An arc from an action vertex  $v_1$  to  $v_2$  is created in  $A_{op}$  if  $v_1$  immediately precedes  $v_2$  in an ordered sequence of nodes.
- A vertex  $v$  representing a conditional node  $n$  is linked to the vertices  $v_1 \dots v_n$  representing the first executable node for each *clause* (i.e. the *then* clause, the *else* clause, ...) in  $n$ . The last vertex in each clause is linked to the vertex  $v_{next}$  immediately following  $n$  in the sequence of executable nodes. If  $n$  does not include an *else* clause, an arc between  $v$  and  $v_{next}$  is also added to  $A_{op}$ .
- A vertex  $v$  representing a loop node  $n$ , is linked to the vertex representing the first executable node for  $n.bodyPart$  (returning the list of actions in the body of the loop) and to the vertex  $v_{next}$  immediately following  $n$  in the node sequence. The last vertex in  $n.bodyPart$  is linked back to  $v$  (to represent the loop behavior).

- A vertex representing an *OperationCall* action is replaced by the sub-digraph corresponding to the called operation *c* like follows: (1) the initial node of *c* is connected with the node that precedes the *OperationCall* node in the main operation, (2) the final node of *c* is connected with the node/s that follow the *OperationCall* node and (3) the parameters of *c* are replaced by the arguments in the call.

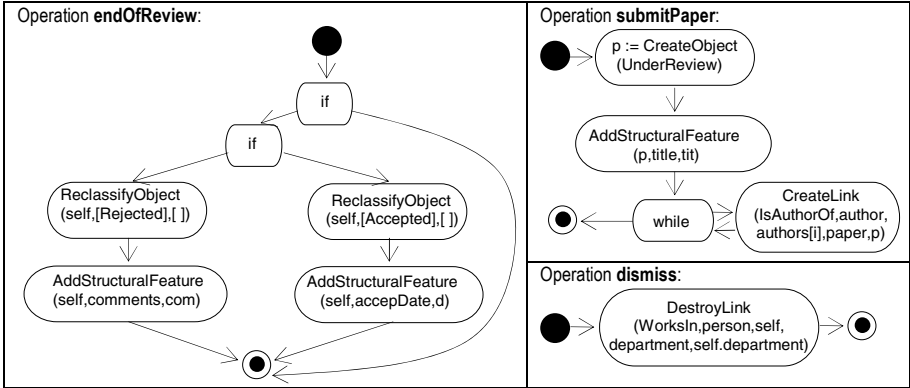


Fig. 6. MBCFG of *endOfReview*, *submitPaper* and *dismiss* operations for the example

Fig. 6 shows the MBCFGs for the operations in Fig. 4. Test conditions of conditional and loop nodes are not shown since they are not part of our analysis<sup>2</sup>.

Given a  $MBCFG_{op}$  graph  $G$ , the set of execution paths  $ex_{op}$  for  $op$  is defined as  $ex_{op} = allPaths(MBCFG_{op})$  where  $allPaths(G)$  returns the set of all paths in  $G$  that start at the initial vertex (the vertex corresponding to the initial node), end at the final node and does not include repeated arcs (these paths are also known as *trails* [2]).

Each path in  $ex_{op}$  is formally represented as a sequence of  $\langle number, action \rangle$  node tuples where *number* indicates the number of times that the action *action* is executed in that node. Vertices representing other types of executable nodes are discarded.

The *number* in the tuple is only relevant for actions included in loop nodes. For other actions the number value is always ‘1’. For an action *ac* within a loop, *number* is computed as follows: (1) each *while-do* loop in the graph is assigned a different variable  $N, \dots, Z$  representing the number of times the loop may be executed. *Do-while* loops are assigned the value  $1+N, \dots, 1+Z$  to express that the body is executed at least once and (2) the *number* of *ac* is defined as the multiplication of the variable values of all loop nodes we find in the path between *ac* and the initial vertex, i.e. *ac* will be executed  $N$  times if *ac* is in a top-level loop,  $N * M$  if *ac* is part of a single nested loop, and so forth. Fig. 7 shows the execution paths for the graphs in Fig. 6.

<sup>2</sup> Detection of infeasible paths due to unsatisfiable tests conditions is out of scope of this paper. This SAT-problem could be tackled with UML/OCL verification tools [3] adding the test condition as an additional constraint and checking if the extended model is still satisfiable.

```

endOfReview:
  p1 = 0
  p2 = [ <1,ReclassifyObject(self,[Rejected],[ ])>, <1,AddStructuralFeature(self,comments,com)> ]
  p3 = [ <1,ReclassifyObject(self,[Accepted],[ ])>, <1,AddStructuralFeature(self,accepDate,d)> ]

submitPaper:
  p = [ <1,p:=CreateObject(UnderReview)>, <1,AddStructuralFeature(p,title,tit)>,
        <N,CreateLink(IsAuthorOf,author,authors[i],paper,p)> ]

dismiss:
  p = [ <1,DestroyLink(WorksIn,person,self,department,self.department)> ]

```

Fig. 7. Execution paths of *endOfReview*, *submitPaper* and *dismiss* operations

## 5 Weak Executability

An operation is *weakly executable* when there is a chance that a user may successfully execute the operation, that is, when there is at least an initial system state and a set of arguments for the operation parameters for which the execution of the actions included in the operation evolves the initial state to a new system state that satisfies all integrity constraints. Otherwise, the operation is completely useless: every time a user tries to execute the operation (and regardless of the input values provided to the operation) an error will arise because some integrity constraint will become violated. We define our *executability* property as *weak executability* since we do not require all executions of the operation to be successful, which could be defined as *strong executability*. Obviously, weak executability is a prerequisite for strong executability. So, designers could check first our weak executability and then, if they think it is necessary, they could apply other techniques (see the related work) to determine the stronger property.

As an example, consider again the operations of Fig. 4. Clearly, *dismiss* is not executable since every time we try to delete a link between a person  $p$  and a department  $d$ , we reach an erroneous system state where  $p$  has no related department, a situation forbidden by the minimum ‘1’ multiplicity in the *WorksIn* association. As we will see later, in order to dismiss  $p$  from  $d$  we need to either assign a new department  $d'$  to  $p$  or to remove  $p$  itself within the same operation execution. Instead, *submitPaper* is weakly executable since we are able to find an execution scenario where the new paper can be successfully submitted (e.g. when submitting a paper whose authors belong to a department that has not previously submitted any other paper). Note that, as discussed above, classifying *submitPaper* as weakly executable does not mean that every time this operation is executed the new system state will be consistent with the constraints. For instance, if a person  $p$  passed as a value for the *authors* parameter belong to a department with already 10 submissions, then, the operation execution will fail because the constraint *MaxPapersSent* will not be satisfied by the system state at the end of the operation execution.

The weak executability of an operation is defined in terms of the weak executability of its execution paths: an operation is weakly executable if at least one of its paths is weakly executable<sup>3</sup>. Executability of a path  $p$  depends on the set of

<sup>3</sup> It is also important to detect and repair all non-executable paths. Otherwise, all executions of the operation that follow one of those paths will irremediably fail.



actions included in the path. The basic idea is that some actions require the presence of other actions within the same execution path in order to leave the system in a consistent state at the end of the execution. Therefore, to be executable, a path  $p$  must satisfy all action dependencies for every action  $ac$  in  $p$ . Dependencies for a particular action are drawn from the structure and constraints of the class diagram and from the kind of modification performed by the action type. For example, the *dismiss* operation is not weakly executable because its single path (see Fig. 7) is not executable since the action *DestroyLink(WorksIn, person, p, department, d)* must be always followed by *CreateLink(WorksIn, person, p, department, d')* or *DestroyObject(p)* to avoid violating the minimum multiplicity. The single path includes none of these actions and thus it is not executable.

To determine if a path  $p$  is weakly executable, we proceed by (1) computing the action dependencies for each action in  $p$  and (2) checking that those dependencies are satisfied in  $p$ . If all dependencies are satisfied, then, we may conclude that  $p$  is weakly executable. In the following, we explain in detail these two steps and provide an algorithm that combines them to determine the executability of a path.

### 5.1 Computing the Dependencies

A dependency from an action  $ac_1$  (the *depender* action) to an action  $ac_2$  (the *dependee*) expresses that  $ac_2$  must be included in all execution paths where  $ac_1$  appears to avoid violating the constraints of the class diagram. It may happen that  $ac_1$  depends on several actions (AND-composition) or that we have different alternatives to keep the consistency of the system after executing  $ac_1$  (OR-composition; as long as one of the possible dependee actions appears in the path, the dependency is satisfied).

Table 1 provides the rules to compute the dependencies for each kind of action, linked with the AND and OR operators, if necessary. These rules are adapted from [4]. The third column (*Shareable*) determines, for each dependency, if two or more dependee actions can be mapped (i.e. share) to the same depender action in the path.

As an example, according to the table 1, a *CreateLink* action needs (when the rule condition is true) a *DestroyLink*, a *CreateObject* or a *ReclassifyObject* action in the same execution path. The first dependency is not shareable, since each *CreateLink* needs a different *DestroyLink* to keep the system consistent. Instead, the alternative dependency *CreateObject* (*ReclassifyObject*) is shareable since several create links may rely on the same new (reclassified) object to satisfy the cardinality constraints.

Note that, to determine the dependencies we just take into account cardinality constraints and disjoint and complete generalization constraints. Other constraints do not affect the weak executability property, since we can always find a system state and/or a set of arguments for which the execution of an action results in a consistent state with respect to those constraints. For instance, constraints restricting the value of the attributes of an object may be satisfied when passing the appropriate arguments as parameters for the action (and similarly with constraints restricting the relationship between an object and related objects). As seen before, *MaxPapersSent* constraint (Fig. 3) does not affect the weak executability of *submitPaper*. It certainly restricts the set of people that may be passed as authors for the submitted paper but it is easy to see that there are many system states (and many possible values for the *authors* parameter) over which the operation can be successfully executed.

**Table 1.** Dependencies for modification actions.  $Min(c_i, as)$  and  $max(c_i, as)$  denote the minimum (maximum) multiplicity of  $c_i$  in  $as$  (for reflexive associations we use the role name).

<b>Depender Action</b>	<b>Dependee Actions</b>	<b>Share-able</b>
$o := CreateObject(c)$	$AddStructuralFeature(o, at, v)$ for each non-derived and mandatory attribute $at$ of $c$ or of a superclass of $c$	No
	<b>AND</b> $\langle min(c, as), CreateLink(as, p, o, p_2, o_2) \rangle$ for each non-derived association $as$ where $c$ or a superclass of $c$ has mandatory participation	No
$DestroyObject(o:c)$	$\langle min(c, as), DestroyLink(as, p, o, p_2, o_2) \rangle$ for each non-derived $as$ where $c$ or a superclass of $c$ has a mandatory participation	No
$CreateLink(as, p_1, o_1:c_1, p_2, o_2:c_2)$ (when $min(c_1, as) = max(c_1, as)$ ) to be repeated for the other end	$DestroyLink(as, p_1, o_1, p_3, o_3)$ (if $min(c_2, as) \langle \rangle max(c_2, as)$ )	No
	<b>OR</b> $CreateObject(o_1)$	Yes
	<b>OR</b> $ReclassifyObject(o_1, [c_1], [])$	Yes
$DestroyLink(as, o_1:c_1, o_2:c_2)$ (when $min(c_1, as) = max(c_1, as)$ ) to be repeated for the other end	$CreateLink(as, p_1, o_1, p_3, o_3)$ (if $min(c_2, as) \langle \rangle max(c_2, as)$ )	No
	<b>OR</b> $DestroyObject(o_1)$	Yes
	<b>OR</b> $ReclassifyObject(o_1, [], [c_1])$	Yes
$AddStructuralFeature(o, at, v)$	-	-
$ReclassifyObject(o, [nc], [oc])$	$AddStructuralFeature(o, at, v)$ for each non-derived and mandatory attribute $at$ of each class $c \in nc$	No
	<b>AND</b> $\langle min(c, as), CreateLink(as, p, o, p_3, o_3) \rangle$ for each $c \in nc$ and for each non-derived association $as$ where $c$ has a mandatory participation	No
	<b>AND</b> $\{ReclassifyObject(o, [], [c_1])$ <b>OR</b> $ReclassifyObject(o, [], [c_n])\}$ for each $c \in nc$ such that $c$ is a subclass in a disjoint and complete generalization $G(superclass, c, c_1, \dots, c_n)$ and not $\exists i \mid c_i \in nc$	Yes
	<b>AND</b> $\langle min(c, as), DestroyLink(as, p, o, p_3, o_3) \rangle$ for each $c$ in $oc$ and for each non-derived association $as$ where $c$ has a mandatory participation	No
	<b>AND</b> $\{ReclassifyObject(o, [c_1], [])$ <b>OR</b> $\dots ReclassifyObject(o, [c_n], [])\}$ for each $c \in oc$ such that $c$ is a subclass in a disjoint and complete generalization $G(superclass, c, c_1, \dots, c_n)$ and not $\exists i \mid c_i \in oc$	Yes

## 5.2 Mapping the Dependencies

Each single dependency  $d = \langle \text{number}, \text{action} \rangle$  computed for a path must be satisfied. Otherwise,  $d$  must be returned as a feedback to the user to help him/her to repair the inconsistency. A dependency is satisfied if it can be successfully mapped to one of the actions in the path.

A dependency  $d$  can be mapped onto a node  $n$  in the path when the following conditions are satisfied: (1)  $d.\text{action}$  and  $n.\text{action}$  are the same (e.g. both are *CreateLink* actions), (2) the model elements referenced by the actions coincide (e.g. both create new links for the same association), (3) all instance-level parameters of  $d.\text{action}$  can be bound to the parameters in  $n.\text{action}$  (free variables introduced by the rules may be bound to any parameter value in  $n.\text{action}$ , while fixed ones must have the same identifier in  $d$  and  $n$ ) and (4)  $d.\text{number} \leq 1$  (for actions that are shareable) or  $d.\text{number} \leq n.\text{number}$  (for non-shareable actions). This comparison may include positive integer abstract variables (when  $n$  is part of a loop, see Section 4). In those cases,  $d$  can be mapped iff there is a possible instantiation of the abstract variables that satisfies the inequality comparison  $d.\text{number} - n.\text{number} \geq 0$ . This can be easily expressed (and solved) as a constraint satisfaction problem [16].

## 5.3 Algorithm to Determine the Weak Executability of a Path

In the following, we present an algorithm for determining the weak executability of an execution path  $path$  on a class diagram  $cd$ . For non-executable paths, the algorithm returns a set of possible repair action alternatives (output parameter *requiredActions*) that could be included in the path to make it executable<sup>4</sup>.

```

function weakExecutability (
in: path: List(<number:Integer, action:Action>),
in: cd: <Set(Class), Set(StructuralFeature), Set(Association),
Set(GeneralizationSet), Set(Constraint)>,
out: requiredActions: Set(List<number:Integer, action:Action>)): Boolean
{
  node: <number:Integer, action:Action>;
  depLists: Set(List<number:Integer, action:Action>):=∅;
  //Loop 1: Computing the dependencies
  i: Integer:=1;
  while i ≤ path->size() do
    node:=getNode(path,i);
    updateDependencies(node,cd,depLists); i:=i+1;
  endwhile
  //Loop 2: Determining the required actions
  executable: Boolean:=false; i:=1;
  while i ≤ depLists->size() and ¬executable do
    requiredActions[i]:=mapping(depLists[i],path);
    if (requiredActions[i] = ∅) then executable:=true;
    else i:=i+1;
    endif
  endwhile
  return executable;}

```

<sup>4</sup> Extending the path with this sequence is a necessary condition but not a sufficient one to guarantee the executability of the path. Actions in the sequence may have, in its turn, additional dependencies that must be considered as well.

Roughly, the algorithm works by executing two loops<sup>5</sup>. The first loop uses the *updateDependencies* function to compute the dependencies for each action in the input *path*. This function updates the variable *depLists* as follows: (1) computes the dependencies for the action in *node.action* as stated in Table 1 (2) multiplies the *number* value in each dependency by the value of *node.number* and (3) adds the dependencies to the end of all lists in *depLists* (if all dependencies for *node* are AND-dependencies) or forks all lists and adds to the end of each cloned list a different dependency (in case of OR-dependencies) to represent the different alternatives we have to satisfy the dependencies.

The second loop tries to map each dependency *d* onto the actions in *path*. The *mapping(depLists[i],path)* function copies in *requiredActions[i]* the actions of *depLists[i]* that either do not map in the path or that map with an insufficient *number* value. In this latter case, the dependency is added indicating the additional number of actions that are needed. In the former, *number* is directly extracted from *d.number*.

If at least one of the lists in *depLists* is fully satisfied the *path* is determined as weakly executable. Otherwise, the algorithm returns in *requiredActions* a list of repair actions for each possible way of satisfying the dependencies.

The execution of the *executability* function for the *submitPaper* and *dismiss* operations (Fig. 4) is shown in Tables 2 and 3. *EndOfReview* is detailed in [22].  $v_1 \dots v_n$  represent free variables introduced by the rules.

The only path of *submitPaper* operation is executable since all dependencies in *depLists[1]* are satisfied by the path (when *N* takes the value 1, the last dependency can be mapped to the first node in the path). Thus, the operation is weakly executable.

**Table 2.** Weak Executability for the *submitPaper* operation

<i>submitPaper</i>	<i>Input path</i>	$p = [ \langle 1, p := \text{CreateObject}(\text{UnderReview}) \rangle, \langle 1, \text{AddStructuralFeature}(p, \text{title}, \text{tit}) \rangle, \langle N, \text{CreateLink}(\text{IsAuthorOf}, \text{author}, \text{authors}[i], \text{paper}, p) \rangle ]$
	<i>Dependencies</i>	$\text{depLists}[0] = [ \langle 1, \text{AddStructuralFeature}(p, \text{title}, v_1) \rangle, \langle 1, \text{CreateLink}(\text{IsAuthorOf}, \text{person}, v_2, \text{paper}, p) \rangle, \langle N, \text{DestroyLink}(\text{IsAuthorOf}, \text{person}, \text{authors}[i], \text{paper}, v_3) \rangle ]$ $\text{depLists}[1] = [ \langle 1, \text{AddStructuralFeature}(p, \text{title}, v_1) \rangle, \langle 1, \text{CreateLink}(\text{IsAuthorOf}, \text{person}, v_2, \text{paper}, p) \rangle, \langle N, p := \text{CreateObject}(\text{UnderReview}) \rangle ]$
	<i>Output</i>	$\text{requiredActions} = \emptyset$ ( $\text{depLists}[1]$ maps correctly with input path <i>p</i> ) $\text{executability} = \text{TRUE}$

**Table 3.** Weak Executability for the *dismiss* operation

<i>dismiss</i>	<i>Input path</i>	$p = [ \langle 1, \text{DestroyLink}(\text{WorksIn}, \text{person}, \text{self}, \text{department}, \text{self}, \text{department}) \rangle ]$
	<i>Dependencies</i>	$\text{depLists}[0] = [ \langle 1, \text{CreateLink}(\text{WorksIn}, \text{person}, \text{self}, \text{department}, v_1) \rangle ]$ $\text{depLists}[1] = [ \langle 1, \text{DestroyObject}(\text{self}) \rangle ]$
	<i>Output</i>	$\text{requiredActions}[0] = [ \langle 1, \text{CreateLink}(\text{WorksIn}, \text{person}, \text{self}, \text{department}, v_1) \rangle ]$ $\text{requiredActions}[1] = [ \langle 1, \text{DestroyObject}(\text{self}) \rangle ]$ $\text{executability} = \text{FALSE}$

<sup>5</sup> We could also mix both loops by checking partial satisfiability of *depLists* after each node (more efficient but with a poorer feedback since only part of the required actions would be returned).

This path is not executable (and thus, neither the *dismiss* operation, since this is its only path), because removing the link violates the multiplicity ‘1’ of *WorksIn*. Adding a new link to the dangling object (with *CreateLink(WorksIn,...)*) or destroying it (with *DestroyObject(self)*) would make the path executable, as reported by our method.

## 6 Completeness

Users evolve the system state by executing the set of write actions defined in the behavior elements of the UML model (the operations in the class diagram in our case). Intuitively, we say that the set of actions in an UML model is *complete* when, all possible changes (inserts/updates/deletes/...) on all parts of the system state can be performed through the execution of those actions. Otherwise, there will be parts of the system that users will not be able to modify since no behavioral elements address their modification. For instance, the set of actions in the operations defined in Fig. 4 is incomplete since actions to remove a person or to create and remove departments are not specified, forbidding users to perform such kind of changes on the data.

We feel this property is important to guarantee that no behavioral elements are missing in the model. Clearly, it may happen that a class diagram contains some elements that designers do not want the users to modify but then those elements should be defined, for instance, as derived information or read-only elements.

More formally, an operation set  $set_{op} = \{op_1, \dots, op_n\}$  is complete when, for each modifiable element  $e$  in the class diagram and each possible action  $ac$  modifying the population of  $e$ , there is at least a weak executable path in some  $op_i$  that includes  $ac$ .

A simple function for checking the completeness of  $set_{op}$  is the following:

```
function completeness (in: cd: <Set(Class), Set(StructuralFeature),
Set(Association), Set(GeneralizationSet)>, in: op: Set(Operation), out:
feedback: Set(Action)): Boolean
{
  requiredActionsSet, existingActionsSet: Set(Action):=∅;
  action: Action; feedback:=∅;
  existingActionsSet:=getExistingActions(op);
  requiredActionsSet:=getRequiredActions(cd);
  for each action ∈ requiredActionsSet do
    if action∉existingActionsSet then feedback:=feedback ∪ {action};
  endif
endfor
return (feedback = ∅);}
```

The parameters of the *completeness* function are the model elements of the class diagram. The result indicates whether the set of operations is complete. For incomplete operations sets, the output parameter *feedback* contains the set of actions that should be included in some operation to satisfy the completeness property. *GetExistingActions* simply retrieves all different actions of weak executable paths of the operations set (*op* parameter). *GetRequiredActions* computes the set of actions that the software system should provide to its users in order to be able to modify all parts of the system state, depending on the structure and properties of the class diagram.

The set of actions returned by *getRequiredActions* is computed by first determining the modifiable model elements in the class diagram (i.e. the elements whose value or population can be changed by the user at run-time) and then deciding, for each modifiable element, the possible types of actions that can be applied on it.

A class is modifiable as long as it is not an abstract class and it is not the supertype of a complete generalization set (instances of such supertypes must be created/deleted through their subclasses). An attribute is modifiable when it is not derived<sup>6</sup>. An association is modifiable if none of its member ends are derived.

For each modifiable class  $c$ , users must be provided with the actions *CreateObject(c)* and *DestroyObject(o:c)*<sup>7</sup> to create and remove objects from  $c$ . For each modifiable attribute  $at$  the action *AddStructuralFeature(o,at,v)* is necessary. For each modifiable association  $as$ , we need the actions *CreateLink(as,p<sub>1</sub>,o<sub>1</sub>,p<sub>2</sub>,o<sub>2</sub>)* and *DestroyLink(as,p<sub>1</sub>,o<sub>1</sub>,p<sub>2</sub>,o<sub>2</sub>)*. For generalizations, we need a set of actions *ReclassifyObject(o,nc,oc)* among the classes involved in them to specialize (generalize) the object  $o$  to (from) each subclass of the generalization. As an example, the result of *getRequiredActions* for our running example is provided in [22].

## 7 Related Work

There is a broad set of research proposals devoted to the verification of behavior specifications in UML, focusing on state machines [15], [14], [18], interaction diagrams [1], sequence diagrams [11], activity diagrams [8] or on the consistent interrelationship between them and/or the class diagram [13], [5], [10], [26], [25], [23], [6], among others. Nevertheless, many of these methods target very basic correctness properties (basically some kind of well-formedness rules between the different diagrams) and/or restrict the expressivity of the supported UML models. Most of the methods above do not accept the specification of actions in the behavior specifications (a relevant exception is [24]), which is exactly the focus of our method.

Another major difference is the formalism used to perform the verification. To check the executability of a behavior specification (or, in general, any property that can be expressed as a Linear Temporal Logic formula - LTL [7]) previous approaches rely on the use of model-checking techniques [12]. Roughly, model checkers work by generating and analyzing all the potential executions at run-time and evaluating if for each (or some) execution the given property is satisfied.

When compared with model-checking methods, our approach presents several advantages. First of all, our analysis is static (no animation/simulation of the model is required) and, thus, our method is more efficient. Model-checking methods suffer from the state-explosion problem (i.e. the number of potential executions to analyze grows exponentially in terms of the size of the model, the domains of the parameters,...) even though a number of optimizations are available (as partial order reduction or state compression). Therefore, in general, it is not possible to explore all possible executions. This implies that results provided by these methods are not conclusive, i.e. absence of a solution cannot be used as a proof: an operation classified as not weakly executable may still have a correct execution outside the search space explored during the verification. Another advantage of our method is the kind of feedback provided to the designer when a property is not satisfied. Model-checking based proposals provide example execution traces that do (not) satisfy the property. In

---

<sup>6</sup> Read-only attributes are considered modifiable because users must be able to initialize their value (and similar for read-only associations).

<sup>7</sup> Or a generic operation *DestroyObject(o:OclAny)* to remove objects of any class.

contrast, our method provides a more valuable feedback (for our correctness analysis) since it suggests how to change the operation specification in order to repair the detected inconsistency.

As a trade-off, our method is unable to verify arbitrary LTL properties. In this sense, we believe our method could be used to perform a first correctness analysis, basic to ensure a basic quality level in the actions specification. Then, designers could proceed with a more detailed analysis adapting current approaches presented above to the verification of behaviors specified with AS. For instance, example execution traces that lead to an error state would help designers to detect particular scenarios not yet appropriately considered.

Finally, we would like to remark that, to the best of our knowledge, our method is the first one considering the completeness and syntactic analysis of action specifications.

## 8 Conclusions and Further Work

We have presented an efficient method for the verification of the correctness of AS specifications. In particular, we have focused on the verification of actions specified as part of the definition of the effect of imperative operation specifications, one of the key elements in all MDD methods. Our approach can be easily extended to cope with other kinds of behavioral specifications since all of them use AS for a fine-grained behavior specification.

Our method is based on a static analysis of the dependencies among the actions; an animation/simulation is not required. Thus, our method does not suffer from the state-explosion problem as current model-checking methods. As a trade-off, our method is not adequate for evaluating general LTL properties. Moreover, the feedback provided by our method helps designers to correct the detected errors since our method is able to suggest a possible repair procedure instead of just highlighting the problem.

Therefore, we believe that the characteristics of our method make it especially suitable for its integration in current CASE and code-generation tools, as part of the default consistency checks that those tools should continuously perform to assist designers in the definition of software models.

As a further work, we would like to complement our techniques by providing an automatic transformation between the UML AS specification and the input language of a model-checker tool (as the PROMELA language [12]) so that, after an initial verification with our techniques (simpler and which would efficiently provide a first correctness result), designers may get a more fine-grained (though partial) analysis by means of applying more complex model checking techniques. Also, we also plan to implement/integrate these techniques into a CASE tool and validate them with a more complex case study. In addition, we plan to empirically evaluate the computational cost of each technique and compare them.

**Acknowledgements.** Thanks to the anonymous referees and the people of the GMC group for their useful comments to previous drafts of this paper. This work has been partly supported by the Ministerio de Ciencia y Tecnología under TIN2008-00444 project, Grupo Consolidado.

## References

1. Baker, P., Bristow, P., Jervis, C., King, D., Thomson, R., Mitchell, B., Burton, S.: Detecting and Resolving Semantic Pathologies in UML Sequence Diagrams. *ESEC/SIGSOFT FSE*, 50–59 (2005)
2. Bollobas, B.: *Modern graph theory*. Springer, Heidelberg (2002)
3. Cabot, J., Clarisó, R., Riera, D.: UMLtoCSP: a tool for the formal verification of UML/OCL models using constraint programming. *ASE*, 547–548 (2007)
4. Cabot, J., Gómez, C.: Deriving Operation Contracts from UML Class Diagrams. In: Engels, G., Opdyke, B., Schmidt, D.C., Weil, F. (eds.) *MODELS 2007*. LNCS, vol. 4735, pp. 196–210. Springer, Heidelberg (2007)
5. Gallardo, M.M., Merino, P., Pimentel, E.: Debugging UML Designs with Model Checking. *Journal of Object Technology* 1(2), 101–117 (2002)
6. Egyed, A.: Instant Consistency Checking for the UML. In: *ICSE*, pp. 381–390 (2006)
7. Emerson, E.A.: Temporal and Modal Logic. *Handbook of Theoretical Computer Science* 8, 995–1072 (1990)
8. Eshuis, R.: Symbolic Model Checking of UML Activity Diagrams. *ACM Transactions on Soft. Eng. and Methodology* 15(1), 1–38 (2006)
9. Garousi, V., Briand, L., Labiche, Y.: Control Flow Analysis of UML 2.0 Sequence Diagrams. In: Hartman, A., Kreische, D. (eds.) *ECMDA-FA 2005*. LNCS, vol. 3748, pp. 160–174. Springer, Heidelberg (2005)
10. Graw, G., Herrmann, P.: Transformation and Verification of Executable UML Models. *Electronic Notes in Theoretical Computer Science* 101, 3–24 (2004)
11. Grosu, R., Smolka, S.A.: Safety-Liveness Semantics for UML 2.0 Sequence Diagrams. In: *ACSD*, pp. 6–14 (2005)
12. Holzmann, G.J.: *The spin model checker: Primer and reference manual*. Addison-Wesley Professional, Reading (2004)
13. Knapp, A., Wuttke, J.: Model checking of UML 2.0 interactions. In: Kühne, T. (ed.) *MODELS 2006*. LNCS, vol. 4364, pp. 42–51. Springer, Heidelberg (2007)
14. Latella, D., Majzik, I., Massink, M.: Automatic Verification of a Behavioural Subset of UML Statechart Diagrams using the SPIN Model-Checker. *Formal Aspects of Computing* 11(6), 637–664 (1999)
15. Lilius, J., Paltor, I.P.: Formalising UML State Machines for Model Checking. In: France, R.B., Rumpe, B. (eds.) *UML 1999*. LNCS, vol. 1723, pp. 430–445. Springer, Heidelberg (1999)
16. Marriott, K., Stuckey, P.J.: *Programming with Constraints: An Introduction*. MIT Press, Cambridge (1998)
17. Mellor Stephen, J., Balcer Marc, J.: *Executable UML: A foundation for model-driven architecture*. Addison-Wesley, Reading (2002)
18. Ober, I., Graf, S., Ober, I.: Validating Timed UML Models by Simulation and Verification. *Int. Journal on Software Tools for Technology Transfer* 8(2), 128–145 (2006)
19. Object Management Group (OMG): *UML 2.0 Superstructure Specification*. OMG Adopted Specification (ptc/07-11-02) (2007)
20. Object Management Group (OMG): *Semantics of a Foundational Subset for Executable UML Models RFP (ad/2005-04-02)* (2005)
21. Olivé, A.: Conceptual Schema-Centric Development: A Grand Challenge for Information Systems Research. In: Pastor, Ó., Falcão e Cunha, J. (eds.) *CAiSE 2005*. LNCS, vol. 3520. Springer, Heidelberg (2005)



22. Planas, E., Cabot, J., Gómez, C.: Verifying Action Semantics Specifications in UML Behavioral Models (Extended Version). LSI-09-6-R LSI Research Report, UPC (2008)
23. Rasch, H., Wehrheim, H.: Checking Consistency in UML Diagrams: Classes and State Machines. In: Najm, E., Nestmann, U., Stevens, P. (eds.) FMOODS 2003. LNCS, vol. 2884, pp. 229–243. Springer, Heidelberg (2003)
24. Turner, E., Treharne, H., Schneider, S., Evans, N.: Automatic Generation of CSP  $\parallel$  B Skeletons from xUML Models. In: Fitzgerald, J.S., Haxthausen, A.E., Yenigun, H. (eds.) ICTAC 2008. LNCS, vol. 5160, pp. 364–379. Springer, Heidelberg (2008)
25. Van Der Straeten, R., Mens, T., Simmonds, J., Jonckers, V.: Using Description Logic 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)
26. Xie, F., Levin, V., Browne, J.C.: Model Checking for an Executable Subset of UML. ASE, 333–336 (2001)

# Using Macromodels to Manage Collections of Related Models

Rick Salay, John Mylopoulos, and Steve Easterbrook

Department of Computer Science, University of Toronto  
Toronto, ON M5S 3G4, Canada  
{rsalay, jm, sme}@cs.toronto.edu

**Abstract.** The creation and manipulation of multiple related models is common in software development, however there are few tools that help to manage such collections of models. We propose a framework in which different types of model relationships -- such as *submodelOf* and *refinementOf* -- can be formally defined and used with a new type of model, called a *macromodel*, to express the required relationships between models at a high-level of abstraction. Macromodels can be used to support the development, comprehension, consistency management and evolution of sets of related models. We illustrate the framework with a detailed example from the telecommunications industry and describe a prototype implementation.

**Keywords:** Modeling, Metamodeling, Macromodeling, Relationships, Mappings.

## 1 Motivation

In Software Engineering, it is common to model systems using multiple interrelated models of different types. A typical modeling paradigm provides a collection of domain specific modeling languages, and/or multi-view languages such as UML. The modeling languages are typically defined through a metamodel, however only very limited support is typically provided for expressing the relationships between the models.

Existing approaches to supporting model relationships tend to focus on how the contents of specific model instances are related (e.g. [7, 10]), rather than on the models themselves. Those approaches that do provide abstractions for expressing relationship between models typically concentrate on a limited set of relationship types required to support model transformation (e.g. [2, 4]) and/or traceability (e.g. [1]). In contrast, we argue that a rich, extensible set of relationship types for relating models is needed, to capture the intended meaning of the models and the various ways that models in a collection can constrain one another.

UML suffers from a similar limitation. The UML metamodel defines a number of diagram types as projections of a single UML model. However, developers rarely manipulate the underlying UML model directly - they work with the diagrams, and the intended purpose of each new diagram is only partially captured in the UML metamodel. For example, a developer might create an object diagram showing

instantiations of a subset of the classes, chosen to be relevant to a particular use case scenario. The relationships of these objects to the classes in the model is captured in the metamodel, but the relationship between this particular object diagram and the set of behaviour models (e.g. sequence diagrams) that capture the scenario is left implicit.

The key point is that each model in a collection is created for a specific purpose, for example to capture important concepts, or to elaborate and constrain other models. Each model therefore has an *intended purpose* and a set of *intended relationships* with other models. Such relationships might include submodels, refinements, aspects, refactorings, transformations, instantiations, and so on. A precise definition of these intended relationships is needed to fully capture the intended meanings of the models.

We propose a framework for systematically extending a modeling paradigm to formally define model relationship types between model types and we introduce a new type of model, called a *macromodel*, for managing collections of models. A macromodel consists of elements denoting models and links denoting intended relationships between these models with their internal details abstracted away. The framework provides a number of benefits:

1. Understandability. When large models are decomposed into smaller ones, the representation of relationships is essential for understandability [9], thus, making the underlying relational structure of a set of models explicit helps comprehension.
2. Specifying constraints on models, even for models yet to be created. When constituent models change, they are still expected to maintain the relationships expressed in the macromodel, because the macromodel captures the intentions of the modelers on how the set of models should be structured.
3. Consistency checking. The macromodel can be used to assess an evolving set of models for conformance with modeler intentions even when mappings between models are incomplete.
4. Model evolution. A change to the macromodel can be taken as a specification for change in the collection of models it represents – either involving the removal of some models, changes in model relationships, or additions of new models. Thus, a macromodel can be used to guide model evolution.

In a short paper [14], we introduce the basic concepts of the framework. In this paper, we elaborate the details, provide a formal semantics for macromodels and describe a new implementation that integrates the framework with the model finder Kodkod [16] to support automated model management activities such as consistency checking and model synthesis.

The structure of this paper is as follows. Section 2 introduces the framework informally and then Section 3 provides a formal treatment. In Section 4, we describe a prototype tool implementation and Section 5 describes the application of the framework to a detailed example from the telecommunications domain. In Section 6 we discuss related work and finally in Section 7 we state some conclusions and discuss future work.

## 2 Framework Description

In the framework we assume that at the detailed level, the relationship between two (or more) models is expressed as a special kind of model that contains the mapping

relating the model elements. Furthermore, these relationships can be classified into types and that they can be formalized using metamodels. For example, Figure 1 shows an instance of the *objectsOf* relationship type that can hold between a sequence diagram and object diagram in UML. Each object symbol in the sequence diagram is mapped to an object symbol in the object diagram that represents the same object (via the identity relation *id*) and each message in the sequence diagram is mapped to the link in the object diagram over which the message is sent (via the relation *sentOver*). In addition, the mapping is constrained so that both *id* and *sentOver* are total functions and the mapping must be consistent in the sense that the endpoint objects of a message should be the same as the endpoint objects of the link to which it is mapped.

The benefit of defining different relationship types such as *objectsOf* is that their instances express the relationships between models at two levels of abstraction. At the detail level, it shows how the elements of the models are related. At the aggregate, or *macro* level, it expresses a fact about how the models are related as a whole and conveys information about how a collection of models is structured. Thus, we can use these to express meaningful macromodels such as the one in Figure 2 that shows the relationships between some of the models and diagrams of a hypothetical library management system. Here we have added the relationship types *caseOf* that holds between a sequence diagram and its specializations, *actorsOf* that holds between a model and an organization UML chart and *diagramsOf* that holds between a collection of UML diagrams and the UML model they are diagrams of.

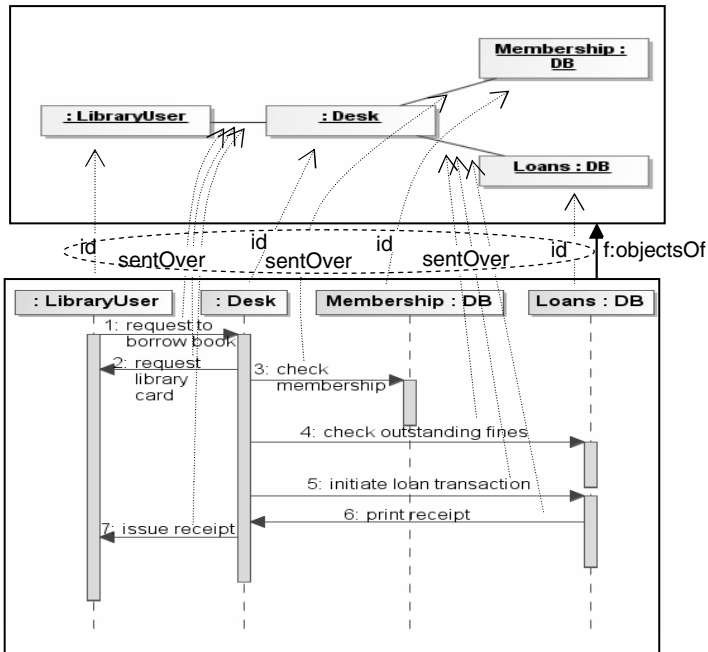
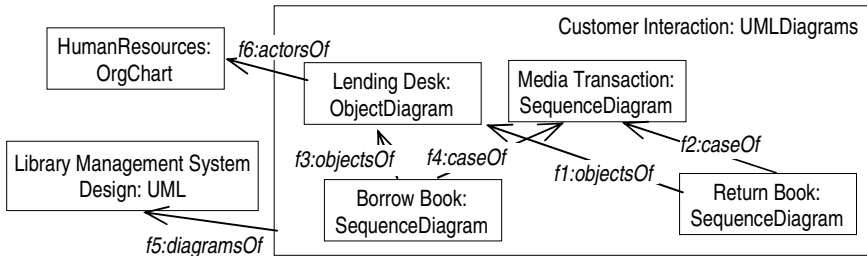


Fig. 1. A relationship between a sequence diagram and an object diagram



**Fig. 2.** Partial macromodel of a library system specification

A macromodel also has a metamodel and can contain well-formedness constraints on the valid configurations of models. For example, in Figure 2, a macromodel of type `UMLDiagrams` can only contain symbols denoting UML diagrams.

The symbols in a macromodel are realized by actual models and mappings. As the development of the library management system proceeds and as these artifacts evolve we expect the relationships expressed in the macromodel to be maintained. Thus, the macromodel provides a new level of specification in which the intentions of the modelers are expressed at the macroscopic level as models that must exist and relationships that must hold between models. Since these constraints are formalized using the metamodels of the relationship types involved, they can be leveraged by automated support for model management activities such as consistency checking and change propagation. We illustrate this in Section 5 with the prototype MCAST (Macromodel Creation and Solving Tool).

The application of the framework to a particular modeling paradigm involves the following steps:

1. The relationship types that are required for relating model types are defined using metamodels.
2. The metamodel for macromodels is defined in terms of the model and relationship types.
3. For a given development project within the paradigm, an initial macromodel is created expressing the required models and their relationships. As the project continues, both the macromodel and the constituent realizations of the models and relationships (i.e. mappings) continue to evolve. During the project, the macromodel is used as a way of supporting the comprehension of the collection of models in a project, specifying extensions to it and for supporting model management activities with tools such as MCAST.

We describe the formal basis for these steps below.

### 3 Formalization

The problem of how to express relationships between models has been studied in a number of different contexts including ontology integration [6], requirements engineering [10, 11] and model management [3]. Bernstein [3] defines the

relationship in terms of the semantics of the models: two models are related when the possible interpretations of one model constrain the possible interpretations of the other model. Thus, it is a binary relation over the sets of interpretations of the models.

At the syntactic level, this relationship can be expressed by embedding the models within a larger *relator* model that contains the mapping showing how the elements of the models are related. Figure 3 shows how the *objectsOf* relationship type between sequence and object diagrams shown in Figure 1 is defined using metamodels. Note that these are simplified versions of portions of the UML metamodel that correspond to the content of these diagrams. In order to formally relate these metamodels, we exploit the following similarity between metamodels and logical theories: a metamodel can be taken to consist of a pair  $\langle \Sigma, \Phi \rangle$  where  $\Sigma$  defines the types of elements used in a model and is called its *signature* while  $\Phi$  is a set of logical sentences over this vocabulary that define the well-formedness constraints for models. Thus, a metamodel is a logical theory and the set of finite models of this theory, in the model-theoretic sense, is also the set of models that the metamodel specifies. We designate this set  $\text{Mod}(\Sigma, \Phi)$ .

Institution theory [5] provides a general way to relate logical theories by mapping the signatures of the theories in such a way that the sentences are preserved. We take a similar approach for metamodels and define the notion of a *metamodel morphism* between two metamodels as follows: a metamodel morphism  $f: \langle \Sigma_A, \Phi_A \rangle \rightarrow \langle \Sigma_B, \Phi_B \rangle$  is a homomorphism of the signatures  $f_\Sigma: \Sigma_A \rightarrow \Sigma_B$  such that  $\Phi_B \models f(\Phi_A)$  - where we have abused the notation and have used  $f$  as a function that translates sentences over  $\Sigma_A$  to sentences over  $\Sigma_B$  according to the mapping  $f_\Sigma: \Sigma_A \rightarrow \Sigma_B$ . Thus, by establishing a metamodel morphism from metamodel A to metamodel B we both map the element types and set up a proof obligation that ensures that any B-model contains an A-model that can be “projected” out of it.

In the example of Figure 3, the metamodel morphisms  $p_{OD}$  and  $p_{SD}$  map in the obvious way into the relator metamodel of *objectsOf* and the fact that they are metamodel morphisms ensures that every well-formed instance of *objectsOf* contains a well-formed instance of *ObjectDiagram* and of *SequenceDiagram* that it relates. In particular, an instance of *objectsOf* constrains the *SequenceDiagram* to have a subset of the objects of the *ObjectDiagram*, and it contains a functional association  $\text{sentOver}: \text{Message} \rightarrow \text{Link}$  that satisfies the consistency constraint that when it maps a message to a link then the endpoint objects of the message must be endpoint objects of the link. Thus,

$$\begin{aligned} \Phi_{\text{objectsOf}} &= p_{OD}(\Phi_{\text{ObjectDiagram}}) \cup p_{SD}(\Phi_{\text{SequenceDiagram}}) \cup \\ &\{ \forall m: \text{Message}. \\ &\quad (\text{linkStart}(\text{sentOver}(m)) = \text{messageStart}(m) \wedge \\ &\quad \quad \text{linkEnd}(\text{sentOver}(m)) = \text{messageEnd}(m)) \vee \\ &\quad (\text{linkEnd}(\text{sentOver}(m)) = \text{messageStart}(m) \wedge \\ &\quad \quad \text{linkStart}(\text{sentOver}(m)) = \text{messageEnd}(m)) \} \end{aligned}$$

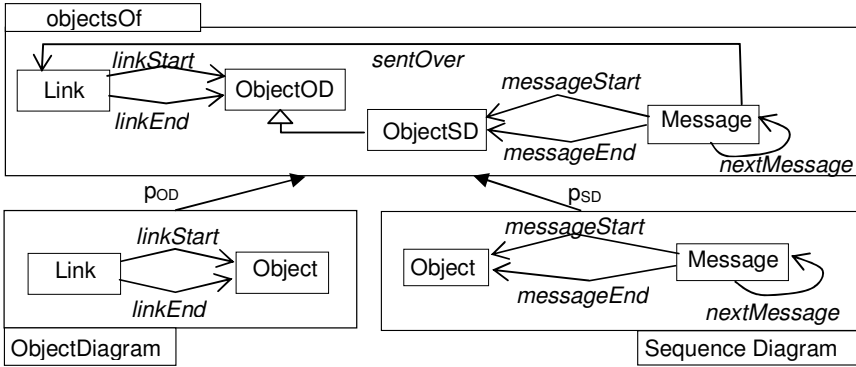


Fig. 3. Defining the *objectsOf* relationship type

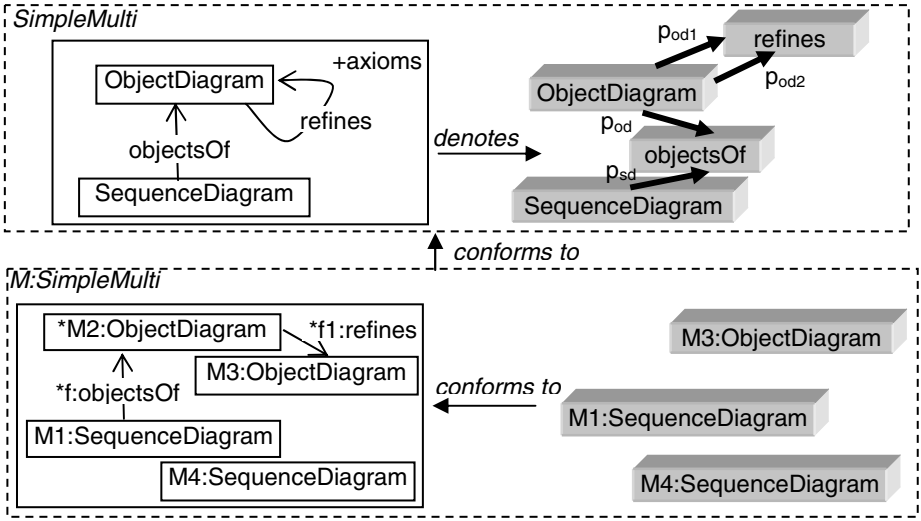
The key benefit of using metamodel morphisms for relating metamodels is that the approach can be formulated in a way that is independent of the metamodeling language since each metamodeling language can define its own type of metamodel morphism. Furthermore, institution theory provides a formal means to relate different logics using *Institution morphisms* [5] and thus our approach extends similarly to multiple metamodeling formalisms; however, we do not pursue this direction further in this paper as it is secondary to our interests here.

In addition to “custom defined” relationship types such as *objectsOf* there are a variety of useful generic parameterized relationship types that can be automatically constructed from the metamodels for the associated model types. We discuss two of these briefly as they are used in Section 5.

Given any model type  $T$ , we can define the relationship type  $eq[T]$  where  $eq[T](M_1, M_2)$  holds iff there is an isomorphism between  $M_1$  and  $M_2$  and where we interpret corresponding elements as being semantically equal – i.e. they denote the same semantic entities. A second parameterized type is  $merge[T]$ . Given a collection of models  $K$  consisting of  $T$ -models,  $merge[T](K, M)$  holds iff  $M$  is the smallest  $T$ -model that contains all of the models in  $K$  as submodels. Note that a unique merge  $M$  may not always exist.

### 3.1 Macromodels

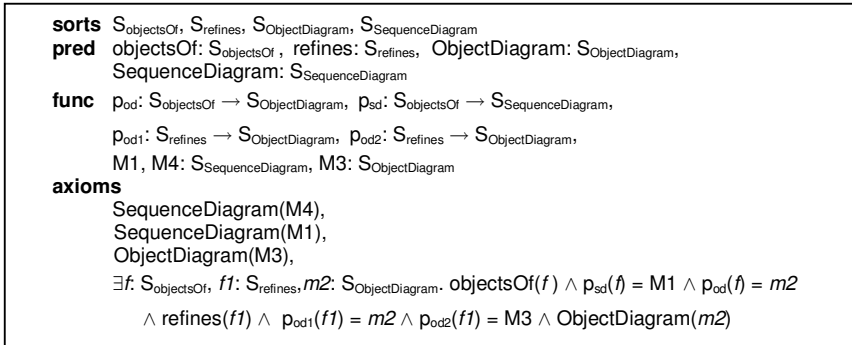
Now that we have an approach for defining relationship types we can characterize a *macromodel type* in terms of the model types and relationship types it contains. A macromodel is a hierarchical model whose elements are used to denote models that must exist and the relationships that must hold between them. Like model and relationship types, a macromodel type is defined using a metamodel. Figure 4 depicts a simple example of this. The upper part shows a macromodel metamodel *SimpleMulti* (left side) and the set of metamodels for the model and relationship types it can depict (right side). The axioms of the macromodel metamodel limit the possible well-formed collections of models and relationships that the macromodel can represent. The notation for macromodels expresses binary model relationship types as directed arrows between model types however they should be understood as



**Fig. 4.** A macromodel metamodel and an instance of it

consisting of a relator model and two metamodel morphisms. The simple illustration in Figure 4 does not depict hierarchy but Figure 9 shows a more complex one that does that we used in the example of Section 5.

The lower part of Figure 4 depicts a particular macromodel  $M:SimpleMulti$  and a collection of models that conform to it. The asterisk preceding  $f:objectsOf$ ,  $M2:ObjectDiagram$  and  $f1:refines$  indicate that they are “unrealized” models and relationships. This implies that there is no corresponding instance for these in the collection of models specified by the macromodel and they are just placeholders used to express more complex constraints. The macromodel in Figure 4 expresses the fact that the collection should contain models M1, M3 and M4 and these must satisfy the constraint that M4 is a sequence diagram and the object diagram corresponding to sequence diagram M1 is a refinement of object diagram M3. Translated into first order logic we get the set of sentences shown in Figure 5.



**Fig. 5.** Example translation



Here we are using projection functions with the same name as their corresponding metamodel morphisms to associate relator models with the models they relate. Each connected set of unrealized elements in the macromodel is translated to an existential sentence with the unrealized models and relationships as existentially quantified variables. We will use this translation approach for defining the semantics of macromodels in general, below.

### 3.2 Macromodel Syntax and Semantics

We now define the syntax and semantics of macromodels formally. Figure 6 shows the abstract syntax and well-formedness rules of the macromodel language<sup>1</sup>. The notation of macromodels is summarized as follows:

- A *Model* element is represented as a box containing the model name and type separated by a colon. When the name is preceded with an asterisk then it is has the *realized* attribute set to **false**.
- A *Relation* element is represented with an arrow, if binary, or as a diamond with *n* legs, if *n*-ary. It is annotated with its name, type and with optional *Role* labels. When the name is preceded with an asterisk then it is has the *realized* attribute set to **false**.
- A sub-*Macromodel* element is represented as a box containing its name and type. It optionally can show the sub-macromodel as the contents of the box.
- A *Macrorelation* element is represented with an arrow, if binary, or as a diamond with *n* legs, if *n*-ary. It is annotated with its name, type and with optional *Macrorole* labels. It can optionally show its contents as a dashed oval linked to the main the arrow or diamond symbol.

Assume that we have a macromodel *K* which has metamodel *T*. To define the formal semantics of macromodels we proceed by first translating *T* to a first order signature  $\Sigma_T$  reflecting the different model and relationship types in it and then translating *K* to the theory  $\langle \Sigma_T \cup \Sigma_K, \Phi_K \rangle$  where  $\Sigma_K$  consists of a set of constants

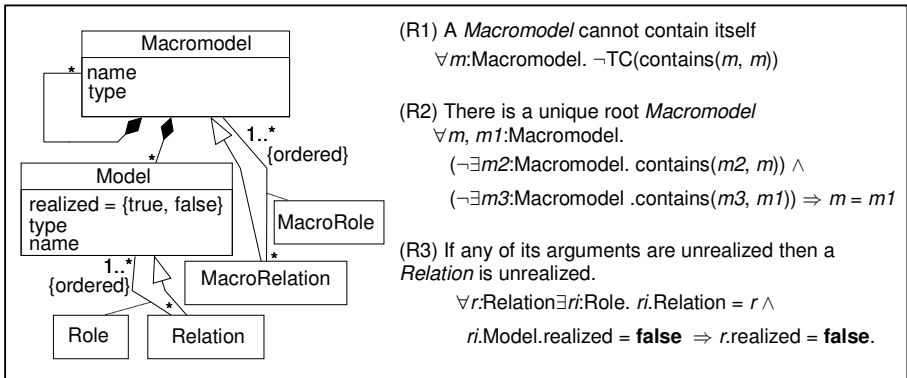


Fig. 6. Abstract syntax and well-formedness constraints of macromodels

<sup>1</sup>  $\text{TC}(\text{pred}(x, y))$  denotes the transitive closure of  $\text{pred}(x, y)$ .

The translation algorithm for  $T$  is as follows:

$\Sigma_T$  is initially empty, then,

- For each metamodel  $X = \langle \Sigma_X, \Phi_X \rangle$  denoted by a *Model* or *Relation* element, add a sort symbol  $S_X$  and a unary predicate symbol  $X:S_X$  to  $\Sigma_T$
- For each metamodel morphism  $p: \langle \Sigma_X, \Phi_X \rangle \rightarrow \langle \Sigma_Y, \Phi_Y \rangle$  denoted by a *Role* element, add a function symbol  $p: S_Y \rightarrow S_X$  to  $\Sigma_T$

The interpretation  $J_T$  is constructed as follows:

- To each sort symbol  $S_X$  assign the set  $\text{Mod}(\Sigma_X, \emptyset)$
- To each predicate symbol  $X:S_X$  assign the unary relation defined by the set  $\text{Mod}(\Sigma_X, \Phi_X)$
- To each function symbol  $p: S_Y \rightarrow S_X$  assign the function  $p: \text{Mod}(\Sigma_Y, \emptyset) \rightarrow \text{Mod}(\Sigma_X, \emptyset)$  induced by the signature morphism  $p_\Sigma: \Sigma_X \rightarrow \Sigma_Y$

The translation algorithm for  $K$  is as follows:

$\Sigma_K$  and  $\Phi_K$  are initially empty, then,

- For each realized *Model* or *Relation* element  $M$  of type  $X$  add the constant  $M:S_X$  to  $\Sigma_K$  and the axiom ' $X(M)$ ' to  $\Phi_K$
- For each *Role* element of type  $p$  from realized relation  $R$  to a realized model  $M$ , add the axiom ' $p(R) = M$ ' to  $\Phi_K$
- For each connected set  $S = \{M_1, \dots, M_n\}$  of unrealized *Model* and *Relation* elements, add the axiom ' $\exists m_1, \dots, m_n. \phi_S$ ' to  $\Phi_K$  where  $\phi_S$  is a conjunction constructed as follows:
  - $\phi_S$  is initially empty, then,
  - For each element  $M_i$  of type  $X$  add the conjunct ' $X(m_i)$ ' to  $\phi_S$ .
  - For each *Role* element of type  $p$  from relation  $M_i$  to realized model  $M$ , add the conjunct ' $p(m_i) = M$ ' to  $\phi_S$ .
  - For each *Role* element of type  $p$  from relation  $M_i$  to model  $M_j$ , add the conjunct ' $p(m_i) = m_j$ ' to  $\phi_S$ .
  - For each *Role* element of type  $p$  from realized relation  $R$  to model  $M_i$ , add the conjunct ' $p(R) = m_i$ ' to  $\phi_S$ .

**Fig. 7.** Semantic interpretation algorithms

corresponding to the realized models and relationships in  $K$  and  $\Phi_K$  is a set of axioms. Figure 5 is the result of performing the translation to the example in Figure 4. We then construct a “universal” interpretation  $J_T$  of  $\Sigma_T$  that consists of all possible models and relationships using these types. Any collection  $M$  of models and relationships conforms to  $K$  iff it is an assignment of the constants in  $\Sigma_K$  to elements of the appropriate types in  $J_T$  such that  $\Phi_K$  are satisfied.

Figure 7 shows the algorithms involved. Note that in the translation algorithm for  $K$ , the connected sets of unrealized *Model* and *Relation* elements are obtained by treating the macromodel as a graph and forming the maximally connected subgraphs consisting of unrealized elements.

## 4 Prototype Implementation: MCAST

The prototype implementation MCAST is built in Java on the Eclipse-based Model Management Tool Framework (MMTF) described in [12] and leverages the Eclipse Modeling Framework (EMF) and related components. Figure 8 shows the architecture

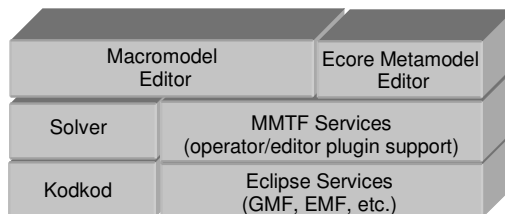
of MCAST. MMTF already provides a simplified version of a macromodel called a Model Interconnection Diagram (MID) used as an interface for invoking the model manipulation operators and editors that can be plugged into the framework. MCAST extends this to a full macromodel editor and provides the Solver module that utilizes the Kodkod [16] model finding engine to solve model management problems expressed using annotated macromodels. We now revisit the three steps for utilizing the framework as described in Section 2 and describe how these steps are implemented using MCAST.

**Step 1: Defining relationship types.** In the formal treatment of section 3, relationship types are expressed using a relator metamodel plus metamodel morphisms. In the implementation we exploit the fact that EMF metamodels (i.e. Ecore) can directly reference other metamodels and thus rather than replicate the endpoint model types within the relator metamodel they are referenced as external metamodels. Axioms are expressed using a textual representation of Kodkod's relational logic language. Metamodels for model types are also expressed in this way.

**Step 2: Defining a macromodel metamodel.** MCAST allows metamodels for macromodels to be defined as Ecore metamodels that extend the base metamodel shown in Figure 6. Each model and relationship type is given as subclass of classes Model and Relation, respectively. In order to implement the mapping in the top part of Figure 4, these are annotated with references to the Ecore metamodels they denote.

**Step 3. Managing the evolution of model collections.** The Solver takes as input, a macromodel with a subset of the model and relationship elements annotated with references to existing models and relationships (i.e. relator models). It then transforms this into a Kodkod model finding problem and uses it to find solutions that assign the remainder of the elements to new models and relationships in such a way that the constraints expressed by the macromodel are satisfied. This can be used in two ways:

- **Simple Conformance Mode:** If the input consists of all of the realized elements assigned to existing models and relationships then a solution exists to the Kodkod problem iff this is a conformant collection and hence the Solver can be used for conformance checking.
- **Extensional Conformance Mode:** If simple conformance mode yields the result that the collection is non-conformant, some of the assigned models and relationships can be marked as “incomplete” and the Solver will allow these to be extended in order to find a conformant solution.



**Fig. 8.** MCAST Architecture

**Table 1.** Example Solver scenarios

Case	Input	Output
1	As in Figure 1, all marked <i>complete</i> .	Conformant
2	As in case 1 but <i>sentOver</i> relations removed from mapping <i>f:objectsOf</i> .	Non-conformant violates constraint that <i>sentOver</i> is a function.
3	As in case 2 but <i>f:objectsOf</i> marked <i>incomplete</i>	<i>f:objectsOf</i> can be uniquely extended to conformance
4	As in case 3 but the link from <i>:Desk</i> to <i>Loans:DB</i> removed.	<i>f:objectsOf</i> cannot be extended due to violation of the endpoint preservation axiom.

In extensional conformance mode, when new models and relationships are constructed as part of finding a conformant solution they are guaranteed to be consistent with the existing models/relationships but of course, this does not mean they are necessarily correct because there may be many possible consistent extensions. When the solution is unique, however, then it must be correct and hence this provides a way to do model synthesis. On the other hand, if a solution cannot be found, this indicates that there is no way to consistently extend the incomplete models/relationships and so this provides a way to do consistency checking with incomplete information.

Note that since Kodkod finds solutions by exhaustively searching a finite bounded set of possible solutions, the above results are valid only within the given bounds. Fortunately, there are common cases in which it is possible to compute upper bounds for model extension that are optimal in the sense that if a conformant extension cannot be found within the bounds then one does not exist. MCAST allows a metamodel to specify such bounds computations using special annotations within the metamodel.

We illustrate both usage modes using a macromodel consisting of the models and *objectsOf* mapping in Figure 1. Table 1 shows four cases in which we applied Solver. In case 1 we passed the models and mapping of Figure 1 (all marked as *complete*) and Solver determined that they satisfied the constraints and hence were conformant. In case 2 we removed all *sentOver* relation instances from the mapping and Solver found the models to be non-conformant because the constraint that *sentOver* is a total function from *Message* to *Link* was violated. Case 3 is the same except that the mapping was marked as *incomplete* and hence we use extensional conformance mode. In this case, we used an upper bound based on the fact that every *objectsOf* relationship is bounded by the models it relates and these would not be extended (i.e. they are marked *complete*). Solver responded by generating an extension of the mapping that filled in the missing *sentOver* links. In this case, Solver identified it as the unique extension that satisfied the constraints, thus it must be the correct one and so it had automatically synthesized the missing part of the mapping. In case 4, we modified the object diagram to remove the link from *:Desk* to *Loans:DB*. Now Solver could not find a conformant extension of the mapping and we can conclude that the models (and partial mapping) are inconsistent with the constraint that the *objectsOf*

relationship holds between the models. Cases 3 and 4 showed that it is possible to work usefully with incomplete (or even non-existent) mappings. This is significant because the creation of mappings is often given little attention in the modeling process and is considered to be overhead.

### 5 A Detailed Example

As a more detailed illustration of the framework we applied it<sup>2</sup> to a design project taken from a standards document for the European Telecommunications Standards Institute (ETSI) [8]. The example consists of three UML models: a context model (4 diagrams), a requirements model (6 diagrams) and a specification model (32 diagrams) and details the development of the Private User Mobility dynamic Registration service (PUMR) – a simple standard for integrating telecommunications networks in order to support mobile communications. More specifically, it describes the interactions between Private Integrated Network eXchanges (PINX) within a Private Integrated Services Network (PISN). The following is a description from the document:

“Private User Mobility Registration (PUMR) is a supplementary service that enables a Private User Mobility (PUM) user to register at, or de-register from, any wired or wireless terminal within the PISN. The ability to register enables the PUM user to maintain the provided services (including the ability to make and receive calls) at different access points.” [pg. 43]

Figure 9 shows the macromodel metamodel *UMLMulti* that we constructed and Figure 10 shows part of the macromodel that we used it to create it. Note that our macromodels include models as well as “diagrams” of these models. We treat a diagram as a special type of model that identifies a submodel of the model for which it is a diagram. This allows both the diagram structure within a UML model and the relational structure across UML models to be expressed within a macromodel.

Due to lack of space we do not show the definitions of the relationship types of *UMLMulti*. Please see [13] for further details. The diagram in Figure 10 shows two sub-macromodels representing the diagrams of the context model and a subset of the diagrams of the specification model.

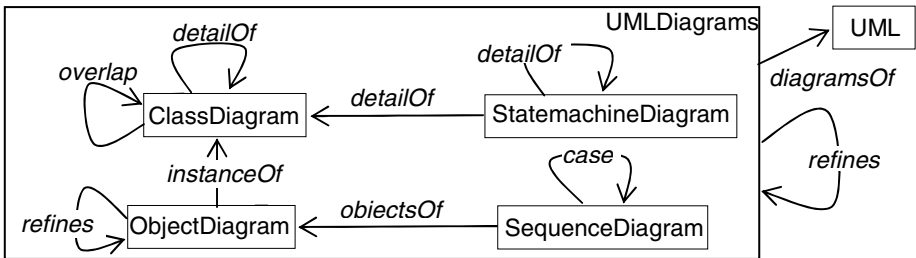


Fig. 9. UMLMulti

<sup>2</sup> Note that this example application was performed by hand – as future work we intend to implement it using MCAST.

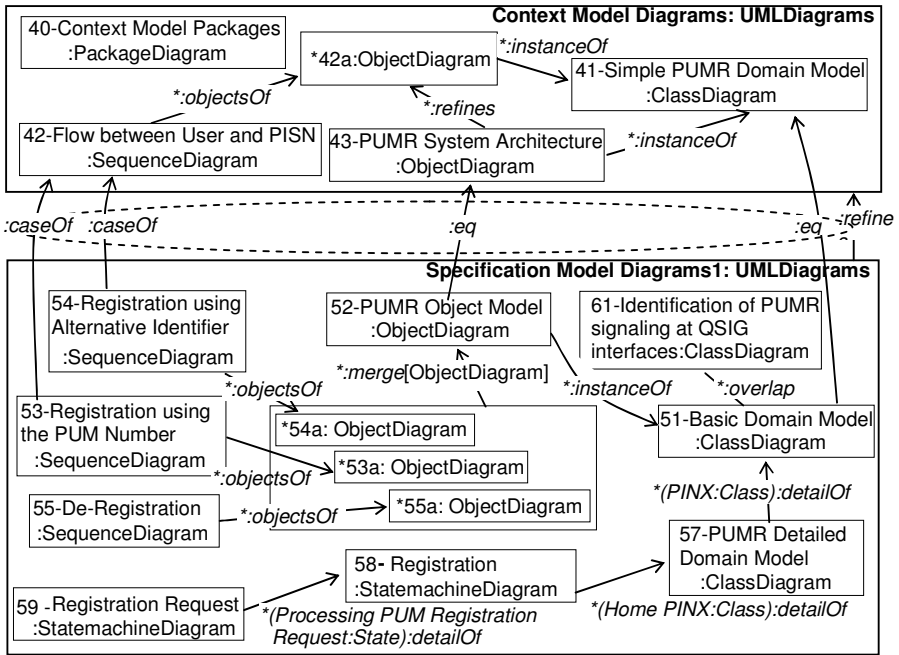


Fig. 10. PUMR Macromodel

Relationships are shown both among the diagrams within each UML model and also between the diagrams across the models. In the latter case, these are aggregated within the *refines* relationship that holds between the two collections of diagrams.

We found two interesting cases where we needed to express complex relationships using unrealized models. The relationship between sequence diagram 42 and class diagram 41 is expressed using the unrealized object diagram \*42a and this is also used to show that object diagram 43 is a refinement of the objects in diagram 42. Another example is the one between the three sequence diagrams 53, 54, 55 and the object diagram 52. The macromodel shows that 52 is the smallest superset (i.e. the merge) of the object diagrams corresponding to each of these sequence diagrams.

Even without understanding the details of the PUMR domain, it should be clear how the expression of the relationships helps to expose the underlying structure in this collection of models and diagrams. In the process of constructing the macromodel, we observed that it significantly helped us to understand how the collection of diagrams contributed toward creating an overall model of the PUMR domain. Unfortunately, since this example involved an existing completed collection, we were not able to assess the hypothesis that the macromodel can be used throughout the development lifecycle to assess conformance and guide development. In order to do this, we are planning to do a more in depth case study that uses our framework from project inception through to completion.

## 6 Related Work

Existing work on dealing with multiple models has been done in a number of different areas. The ViewPoints framework [10] was an influential early approach to multiview modeling. Our approach differs from this work in being more formal and declarative rather than procedural. Furthermore we treat relationships as first class entities and provide support for typing of relationships.

More recently, configurable modeling environments have emerged such as the Generic Modeling Environment (GME) [7]. None of these approaches provide general support for expressing model relationships or their types; hence, they have limited support for defining and expressing interrelated collections of models. Furthermore, the focus of these approaches is on the detail level (i.e. the content of particular models) rather than at the macroscopic level.

Process modeling approaches like the Software Process Engineering Metamodel (SPEM) [15] bear some similarity to our notion of a macromodel metamodel; however, our main focus is in the use of a macromodel at the instance level to allow fine-grained control over the ways in which particular models are related rather than the activities that consume and produce them. However, we believe that macromodels could complement process models by providing a means for specifying pre and post conditions on process activities.

The emerging field of Model Management [3] has close ties to our work but our focus is different in that we are interested in supporting the modeling process whereas the motivation behind model management is primarily model integration.

The term “megamodel” as representing models and their relationships at the macroscopic level emerged first in the work of Favre [4] and also later as part of the Atlas Model Management Architecture (AMMA) [2]. Macromodels bear similarity to these two kinds of megamodels, but the intent and use is quite different – to express the modeler’s intentions in a development process.

Finally, the work on model traceability also deals with defining relationships between models and their elements [1]; however, this work does not have a clear approach to defining the semantics of these relationships. Thus, our framework can provide a way to advance the work in this area.

## 7 Conclusions and Future Work

By its very nature, the process of software development is an activity involving many interrelated models. Much of the research and tools for modeling is focused on supporting work with individual models at the detail level. Working with collections of models creates unique challenges that are best addressed at a macroscopic level of models and their inter-relationships.

In this paper we have described a formal framework that extends a modeling paradigm with a rich set of model relationship types and uses macromodels to manage model collections at a high level of abstraction. A macromodel expresses the relationships that are intended to hold between models within a collection. We have focused on two main ways that macromodels can support modeling. Firstly, they are tools for helping the comprehension of the collection by revealing its intended

underlying structure. Secondly, macromodels can be used to help maintain the model relationships as a collection evolves. In this capacity they are used to guide the development process by ensuring that modelers intentions are satisfied.

Finally, we described the prototype implementation MCAST that integrates the Kodkod model finding engine [16] as a way to support model management activities using macromodels. As part of future work, we are exploring other ways to use a macromodel to manipulate collections of models.

## References

1. Aizenbud-Reshef, N., Nolan, B.T., Rubin, J., Shaham-Gafni, Y.: Model Traceability. *IBM Systems Journal* 45(3), 515–526 (2006)
2. ATLAS MegaModel Management website, <http://www.eclipse.org/gmt/am3/>
3. Bernstein, P.: Applying Model Management to Classical Meta Data Problems. In: *Proc. Conf. on Innovative Database Research*, pp. 209–220 (2003)
4. Favre, J.M.: Modelling and Etymology. *Transformation Techniques in Software Engineering* (2005)
5. Goguen, J.A., Burstall, R.M.: Institutions: Abstract Model Theory for Specification and Programming. *J. ACM* 39(1), 95–146 (1992)
6. Kalfoglou, Y., Schorlemmer, M.: Ontology mapping: the state of the art. *The Knowledge Engineering Review* 18(1), 1–31 (2003)
7. Ledeczi, A., Maroti, M., Bakay, A., Karsai, G., Garrett, J., Thomason IV, C., Nordstrom, G., Sprinkle, J., Volgyesi, P.: The Generic Modeling Environment. In: *Workshop on Intelligent Signal Processing* (2001)
8. Methods for Testing and Specification (MTS); Methodological approach to the use of object-orientation in the standards making process. ETSI EG 201 872 V1.2.1 (2001-2008), [http://portal.etsi.org/mbs/Referenced%20Documents/eg\\_201\\_72.pdf](http://portal.etsi.org/mbs/Referenced%20Documents/eg_201_72.pdf)
9. Moody, D.: Dealing with ‘Map Shock’: A Systematic Approach for Managing Complexity in Requirements Modelling. In: *Proceedings of REFSQ 2006, Luxembourg* (2006)
10. Nuseibeh, B., Kramer, J., Finkelstein, A.: A Framework for Expressing the Relationships Between Multiple Views in Requirements Specifications. *IEEE TSE* 20(10), 760–773 (1994)
11. Sabetzadeh, M., Easterbrook, S.: An Algebraic Framework for Merging Incomplete and Inconsistent Views. In: *13th IEEE RE Conference, Paris, France* (2005)
12. Salay, R., Chechik, M., Easterbrook, S., Diskin, Z., McCormick, P., Nejati, S., Sabetzadeh, M., Viriyakattiyaporn, P.: An Eclipse-Based Tool Framework for Software Model Management. In: *ETX 2007 at OOPSLA 2007* (2007)
13. Salay, R.: Macro Support for Modeling in Software Engineering. Technical Report, University of Toronto, <http://www.cs.toronto.edu/~rsalay/tr/macrosupport.pdf>
14. Salay, R., Mylopoulos, J., Easterbrook, S.: Managing Models through Macromodeling. In: *Proc. ASE 2008*, pp. 447–450 (2008)
15. Software Process Engineering Metamodel V1.1. Object Management Group, <http://www.omg.org/technology/documents/formal/spem.htm>
16. Torlak, E., Jackson, D.K.: A Relational Model Finder. In: Grumberg, O., Huth, M. (eds.) *TACAS 2007*. LNCS, vol. 4424. Springer, Heidelberg (2007)



# A Case Study of Defect Introduction Mechanisms

Arbi Ghazarian

Department of Computer Science, University of Toronto, Canada  
arbi@cs.toronto.edu

**Abstract.** It is well known that software production organizations spend a sizeable amount of their project budget to rectify the defects introduced into the software systems during the development process. An in depth understanding of the mechanisms that give rise to defects is an essential step towards the reduction of defects in software systems. In line with this objective, we conducted a case study of defect introduction mechanisms on three major components of an industrial enterprise resource planning software system, and observed that external factors including incomplete requirements specifications, adopting new, unfamiliar technologies, lack of requirements traceability, and the lack of proactive and explicit definition and enforcement of user interface consistency rules account for 59% of the defects. These findings suggest areas where effort should be directed.

**Keywords:** Defects Sources, Defect Root Cause Analysis, Case Study.

## 1 Introduction

It has been frequently mentioned in the software engineering literature that maintenance activities are the dominant costs of developing software systems. Studies have shown that changes made to software systems account for 40 to 90 percent of the total development costs [4] [10] [3] [7] [6]. These changes can be corrective, adaptive, perfective, or preventive. Corrective changes deal with fixing the defects introduced into the software systems during the development process, and account for a significant portion of the maintenance costs; in their study of 487 data processing organizations, Lientz and Swanson [15] reported that, on the average, about 21% of the maintenance effort is allocated to the corrective maintenance. More recent studies have concluded that, in spite of the advances in software engineering in the past few decades, the maintenance problems have remained the same [18] [22]. These figures clearly indicate the potential economic value that can be gained from leveraging defect prevention techniques. A reduced rate of defects in the delivered software results in a reduction in the corrective maintenance activities, which in turn translates into a lower total development cost.

The importance of defect prevention has been emphasized by quality standards such as the Software Engineering Institute's Capability Maturity Model (SEI-CMM) [26], where defect prevention is a key process area for the optimizing maturity level (i.e., CMM Level 5). According to CMM [25]:

*“Defect prevention involves analyzing defects that were encountered in the past and taking specific actions to prevent the occurrence of those types of defects in the future. The defects may have been identified on other projects as well as in earlier stages or*

*tasks of the current project. Defect prevention activities are also one mechanism for spreading lessons learned between projects.”*

It goes without saying that an important first step to devising tools, techniques, and processes to counteract the mechanisms that give rise to software defects is to gain a profound understanding of these mechanisms. The work reported in this paper is an effort in this direction. However, it should be noted that due to the high variations among software projects in terms of the programming language, development process, design complexity, team size, organizational structure, application domain, individual team member characteristics, and many other factors, it is not possible to identify all possible defect introduction mechanisms in a single case study. Many studies of real-world systems are required to understand the full spectrum of defect introduction mechanisms.

We noticed that there are few published studies of defects on business software systems. Instead, most previous studies have been conducted on system-level software products in other software domains such as telecommunication, networking, real-time, and control systems. As a result, it is not clear to what extent results from these studies are applicable to business software systems. We, therefore, believe that there is a need for more studies of defects, similar to those performed in other domains, in the domain of business software systems. The importance of repeated studies by different researchers in establishing confidence in the results is well recognized by software engineering researchers. Hofer and Tichy [12] analyzed all the refereed papers that appeared in the Journal of Empirical Software Engineering from its first issue in January 1996 through June 2006, and observed that 26% of the papers describing an experiment were replications. Replication is needed to obtain solid evidence [23]. Moreover, it is only through conducting a multitude of similar studies and comparing the findings of these studies (i.e., looking for commonalities, differences, and patterns in defects in various projects) that we can gain deeper insights into questions such as:

- Which defect introduction mechanisms are common in all types of systems?
- Which defect introduction mechanisms are specific to, or occur more frequently in certain application types or domains? For instance, are the types of defects in business software applications different from those of scientific software systems?
- On average, what percentage of defects in software systems are pure logic mistakes on the programmers’ part, and what percentage of defects are rooted outside the code and in other external sources such as incorrect or incomplete specifications?

Eldh et al. [8] emphasize that it is important to regularly collect and report findings about software defects from real industrial and commercially used systems to keep information in tune with development approaches, software and faults. They identify the lack of recent industry data for research purposes as the key problem. This view is supported by Mohagheghi et al. [17] who argue that there is a lack of published empirical studies on industrial systems and that many organizations gather large volumes of data on their software processes and products, but either the data are not analyzed properly, or the results are kept inside the organization. This situation hinders the spreading of lessons learned between projects in various organizations. The case study reported in this paper is a response to this need for more empirical studies on industrial systems. The main purpose of the study is to collect empirical evidence to answer the following two research questions:

1. What are the mechanisms that gave rise to defects in the case under study?
2. How large a role each identified mechanism has played in introducing defects in the case under study?

To answer our research questions, we performed root cause analysis (RCA) on 449 defects from a commercial Enterprise Resource Planning (ERP) software system. Our analysis, backed up by evidence drawn from project data including the defect reports in the defect tracking system, source code, requirements specifications, and test cases, as well as group sessions and individual interviews with the project team members identified a number of defect categories and their root causes, along with their frequencies.

Overall, we found that in our case, the causes for 59% of the reported defects were rooted in external factors including incomplete requirements specifications, adopting new, unfamiliar technologies, lack of requirements traceability, and the lack of proactive and explicit definition and enforcement of user interface consistency rules.

The rest of this paper is organized as follows: in Section 2, we report on our empirical study of defect introduction mechanisms. In Section 3, we discuss the related work. The conclusion and directions for future work follow in Section 4.

## 2 Case Study

### 2.1 Context of the Case Study

The organization where the case study was conducted is a manufacturer of telecommunication devices and an information technology company. For confidentiality reasons, we keep the organization anonymous. For the past five years, the IT department has been actively involved in developing a web-based ERP system comprised of various subsystems<sup>1</sup> including Bookkeeping, Inventory Management, Human Resources, Administration, Manufacturing, Procurement, and Workflow Management. The implementation of the ERP system is carried out in Java programming language, and the software developers involved in the project have an average of 7 years of industry experience.

The IT department follows a customized development process, which borrows concepts from both Rational Unified Process (RUP) and Extreme Programming (XP) methodologies. For instance, most programming is performed in pairs, which is an XP practice, whereas the requirements and analysis phases are conducted through a more traditional RUP-like process using use cases. At the time of this study, the project team was comprised of 28 individuals in various roles including a development manager, 5 system analysts, 13 developers, 4 testers, 2 graphics designers, and 3 marketing representatives. As with most long-term software projects, a number of individuals have left the team during the project, while a few others have joined the team. The team has had 35 members in its largest size. The development of the Bookkeeping, Human Resources, Administration, and Inventory Management subsystems has been completed, while the remaining subsystems are still under development. The final product is estimated to contain 200,000 lines of code.

---

<sup>1</sup> Throughout this paper, we use the terms system, subsystem, component, and module (source-code file) as units of system decomposition from largest to smallest, respectively.

The company uses the following process for handling defects. When a defect is detected, a defect report is registered in a defect tracking software tool. Each defect report captures a set of information including a unique defect identifier, a summary of the defect, a detailed description, the date the defect report is created, the subsystem to which the defect is attributed, the reporter of the defect, the assignee of the defect, priority, status, resolution, any number of additional comments by team members, and the date the defect report is last updated.

During system testing, all detected defects are assigned to a single team member (i.e., a point of contact between the testing and development teams), who in turn reviews the reported defects and further assigns them to the appropriate developers (i.e., the developer who has introduced the defect into the system) to be fixed. The idea behind this practice is that the developer who implements a feature is also the most qualified team member to rectify the defects reported on that feature since he/she is considered to be the most knowledgeable team member about the implementation details of that feature and therefore should be able to rectify the defect more reliably and in less time.

All team members, periodically or upon request by a team member, review all or some of the reported defects, and if they have any specific information or comments that can facilitate the correction of the defects, add them to the defect reports in the defect tracking system. Typical information in the added comments includes the cause of the defects, the location of the defects in the source code, how to fix the defects, and comments to clarify the descriptions of the defects. Developers assigned to the defects then use this information to correct the reported defects.

## 2.2 Description of the Case

All ERP subsystems in the studied organization are built on top of a shared infrastructure layer, which is composed of a set of reusable libraries and frameworks. The components in the infrastructure layer are either developed in-house or acquired as open-source software. All subsystems follow a three-tier layered architectural style, which is comprised of user interface, application logic, and data access layers. Each layer is considered a distinct component and as such each subsystem is divided into three major components. Our case study concerns the three components of the Book-keeping subsystem. The first five columns in Table 1 present the characteristics of the target components in terms of the types of the modules (i.e., source-code files) in each

**Table 1.** Characteristics and Metrics of the Target Components

Component	Module Type	Size (LOC)	Module Count	Avg. Module Size (LOC)	Incorrect Impl.	Missing Impl.	Defect Count	Defect Density
User Interface	JSP	7802	41	191	180	150	330	38.606
	Javascript	678	1					
	XML	126	3					
Application Logic	Java	9434	24	393	22	41	63	6.677
Data Access	Java	4602	59	78	56	0	56	12.168
<b>Total</b>		<b>22642</b>	<b>128</b>		<b>258</b>	<b>191</b>	<b>449</b>	

component, the size of each component, the number of modules in each component, and average module size. The sizes of the components and modules are reported in non-commented and non-blank lines of code (LOC).

### 2.3 Case Study Process and Data Collection

At the time of this study, there were 482 defect reports in the defect tracking system, assigned to the target subsystem that we used in our case study. The time span between the first and the last defect was 17 months. Of these, 4 defects were labeled as “Duplicate”, and 29 defects were labeled as “Not a Bug”. The defects labeled as “Duplicate” were reported twice in the defect tracking system, whereas the ones labeled as “Not a Bug” were not actually defects and were initially reported as defects as a result of the incorrect usage of the subsystem or the unfamiliarity of the reporters of these defects with the correct behavior of the subsystem. We excluded these 33 defect reports, which left us 449 unique defects to study.

To analyze the distribution of the defects over the target components, the author of the paper and a member of the studied organization, who had a thorough understanding of the system, independently followed the analysis process described below to analyze each of the defect reports and attribute them to their corresponding components. This analysis of the distribution of the defects over the three target components was required since this information was not readily available; records in the defect tracking system included a data field that captured the attribution of the defects to the subsystems, but no data fields were available to capture the attribution of the defects to the lower-level units such as components and modules. The results from this analysis were required to compute the defect count and defect density measures for each of the components.

We used the following process to analyze the distribution of the reported defects over the three target components. We started our analysis by checking the information recorded in the “Defect Summary”, “Detailed Description”, and all of the available “Additional Comments” fields for each of the defect reports. Based on the information recorded in these fields, a group of the defects could be directly attributed to their corresponding components in the source code. For the remaining group of the defects, where the attribution of the defects to the components could not be derived from the information available in the defect reports, we recovered this information through conducting a series of defect review sessions. Each session was attended by two people: a research investigator who facilitated the session and recorded the recovered information, and a developer who had been involved in fixing the defects. During each defect review session, all defects fixed by the participant developer were discussed and attributed to their corresponding component. Where necessary, the system’s source code was consulted to locate the components related to the defects. In addition to the attribution of the defects to their corresponding components, where possible, for each defect, we identified the internal cause of the defect in the source code (e.g. an incorrect database query statement or missing source code statements). We also determined whether the defect was caused as a result of a missing or incorrect implementation.

To ensure the quality of the collected information, the abovementioned analysis process was conducted twice and independently. The results of the two analyses were in close agreement (Cohen’s Kappa inter-rater agreement of 0.79 ), which is an indication

of the objectivity of the analyses performed. The cases where there was a difference between the two analyses were jointly reexamined to reach a consensus.

We then measured the sizes of the target components in non-blank and non-commented lines of code using a software tool, and computed the defect density for each of the target components. The results are summarized in the last two columns in Table 11 which present the Defect Count and Defect Density (in defects per KLOC) for each of the target components, respectively.

We followed our data collection and analysis process by performing root cause analysis of the reported defects through conducting a series of group sessions and interviews with the team members. During these sessions, we used input from project team members and, for each defect, identified the external factor that underlay the internal cause of the defect (e.g., the unfamiliarity of the developers with the new query language underlying the incorrect query statements, or incomplete requirements specification documents underlying the missing source code statements). To ensure the correctness of the identified root causes, the team members frequently consulted the project data including defect information in the defect tracking system, requirements documents, test cases, and the source code, as well as the results of the analysis of the attribution of the defects to the components and the collected project metrics (size, defect count, and defect density). As a result of these sessions, we traced each defect back to its origin, which led to the classification of the defects based on their root causes. Unfortunately, the available data was not detailed enough to allow us to calculate the cost and effort spent on rectifying the defects.

## 2.4 Results

Based on a detailed analysis of the data collected during the study, we make several observations about the mechanisms that gave rise to defects in the studied subsystem. We discuss the impact of each identified mechanism on defect rate of the subsystem.

**The Impact of Adopting New, Unfamiliar Technologies on Defect Rate.** The data from our study show that of the 56 defects attributed to the data access component (see Table 11), 47 (roughly 84%) were caused by incorrect database query statements. We discussed this finding with the development team and found that the unexpected number of query-related defects in the data access component was due to the adoption of a new database query technology. The development team had adopted an unfamiliar query language to implement the data access component. Since there had been no previous experience and expertise on the newly adopted query language, the team had encountered many problems with this new technology. Only 16% of the defects attributed to the data access component were caused by the code written in the Java language, which constitutes the bulk of the code in the data access component and serves as the host language for the embedded queries.

We excluded the defects directly caused by the introduction of the new technology and recalculated a defect density of 1.955 for the data access component. Comparing this new calculated value for the defect density in the data access component with its current value from Table 11 (12.168) clearly demonstrates the negative impact of adopting the new, unfamiliar technology in increasing the defect rate in this component. As

a result of adopting the unfamiliar technology, the defect density of the data access component has increased by a factor of 6.22. The data from our study suggest that:

*The adoption of new, unfamiliar technologies into a software component is a risk factor that has adverse effects on the component's quality in terms of its defect rate.*

This observation is not surprising. There is an intuitive consensus in the software engineering literature on the correctness of this proposition. However, empirical evidence taken from industrial software systems, like ours, to support it can strengthen our beliefs in this proposition.

**The Impact of Incomplete Requirements Specifications on Defect Rate.** As part of our data analysis, we classified the reported defects under two broad categories of incorrect implementation and missing implementation, and observed that about 57.5% of the defects (258 cases) were caused as a result of incorrect implementations of the requirements in the code, whereas the remaining 42.5% of the defects (191 cases) were a result of missing implementations from the code. Columns 6 and 7 in Table 1 present the distribution of defects classified as incorrect versus missing implementation over the three components. A chi square test of independence, at the significance level of 0.05, revealed that the type of defect (incorrect or missing implementation) is dependent on component type (user interface, application logic, or data interface). We further observed that the majority of the defects classified under the missing implementation category were related to missing business rules and data validations. Defects caused by missing implementations (i.e., the code that is necessary is missing) have also been referred to as "*faults of omission*" in the literature [8].

Our inspection of the requirements specification documents revealed that in 156 cases (roughly 82% of the defects in the missing implementation category), the system analysis team had not explicitly included the requirements in the requirements specification documents, and were consequently missing from the system's implementation as well. In a series of defect root cause analysis group session with the team members, we reviewed all of the 191 defects in the missing implementation category, and confirmed that the reason for the 156 missing implementation cases which didn't have corresponding requirements was actually the missing requirements.

We know for a fact that the requirements specification documents were the main means through which the requirements of the system were communicated to developers. The introduction of this group of defects into the system's source code can be directly traced back to the incomplete requirements specifications. This conclusion is consistent with the results of the interviews conducted with the development team members. When asked about the relatively large number of defects related to missing implementations, the team members expressed their opinions in statements such as "what is obvious for the business analysis team is not clear to anyone else in the development team. Consequently, if they fail to communicate some of the business requirements in an explicit manner, we are highly likely to miss these requirements in our implementations" and "when we start the development of a new use case of the system, we are not provided with all the details. What we initially receive from the business side includes a description of the use case including the main and alternative flows, and some of the major related business requirements, but the documents are not comprehensive enough

to cover all aspects of the use cases including some of the data validations and less obvious business rules. Therefore, a number of missing requirements are detected during the system testing”.

An interesting aspect of this observation is that it puts into question the comprehensiveness of the traditional view of a defect as any characteristic of the system that does not comply to its predetermined (proactively and explicitly documented) specification. In our case, we observed that there were no predetermined specifications for a significant number of functionalities in the system and yet the absence of these functionalities from the system were reported as defects, whose rectification were required for the correct operation of the system. An interview with the testing team revealed that they had partly relied on their implicit knowledge of the system domain to test the software system. For instance, while the requirements documents lacked some of the data validation rules, the testing team, relying on their knowledge of the system, had identified and incorporated some of these missing rules into their test cases. A consequence of this phenomenon is that a part of software requirements are documented outside the requirements specification documents, and inside the test cases and defect records. This practice, over time, can lead to the loss of parts of the system knowledge as a part of requirements are buried within test cases and defect reports. Our data suggest that:

*Incomplete requirements specifications (i.e., some requirements are not explicitly stated in the specifications) are a mechanism for introducing defects into software systems in the form of missing implementations.*

As mentioned earlier, in our case, the development team was following a traditional document-centric requirements process. This means that the majority of communication between the business analysis and development teams was taking place through requirements documents. A direct consequence of this reliance on documentation as the main form of communicating system requirements is that the performance of the system developers (e.g., in terms of the number of defects related to missing implementations introduced into the system) becomes partly dependent on the quality of the documents produced by the requirements team (e.g., in terms of the completeness of the requirements). The results could be different in projects with an agile requirements process, where the project team mostly relies on verbal communication of requirements.

**The Impact of the Lack of Requirements Traceability on Defect Rate.** An interesting observation was that in the remaining 35 defects classified under the missing implementation category (roughly 18% of the defects in this category), the corresponding requirements were existing somewhere in the requirements documents and were somehow overlooked by developers.

Our inspection of the requirements documents in conjunction with group sessions with the team members revealed that these cases were related to the business concepts or data items and their associated business rules and data validations that were defined in one requirements document and implicitly referred to in other requirements documents. For instance, consider the case where a step in one of the system use cases states that the user shall enter a data item into the system as part of the data entry for that use case, without explicitly making references to the other use cases of the system where the data validation rules for this data item have been specified. Since there were no explicit



links between the parts of a document that referred to external business concepts or data items and the parts of the other documents that actually specified the requirements for that concept (requirement-requirement traceability link), developers either did not realize that some of the requirements pertaining to the feature are defined somewhere else, or had to manually navigate between the various requirements documents to capture a complete view of the requirements pertaining to the feature under development. This process of moving from a requirements document to another in order to collect all the relevant requirements can be problematic when a document has many points of jump (i.e., items mentioned in a document are specified in other documents, but not explicitly linked to those external documents). In other words, the requirements related to a feature under development were scattered across different documents without an explicit mechanism for cross-referencing, and the human errors involved in the process of navigating between the various documents and collecting the complete set of requirements had led to the introduction of a group of defects into the system in the form of missing implementations. Since, other than system testing, there was no other mechanism in place to verify the completeness of the implementation of a feature under development with regards to its specified requirements, these missing implementation defects were remained hidden until system testing. Our data suggest that:

*The lack of traceability between requirements specification documents, when the requirements pertaining to a feature of the system are scattered across multiple documents, plays a role in the occurrence of the cases where the requirements are existing in the requirements documents but missing from the implementation.*

**The Impact of Not Proactively Defining and Enforcing the User Interface Consistency Rules on Defect Rate.** Another observation is that the cause of 8% of the defects in the user interface component (or 6% of the total number of defects in the subsystem under study) can be directly traced back to inconsistencies in the user interface. A close examination of this group of defects in conjunction with a group session with developers revealed that in the absence of explicit consistency rules for the unification of the system's user interface behavior, developers had made individualistic and ad hoc decisions in their implementations, which led to inconsistencies in the user interface of the system. The descriptions given for these defects in the defect tracking system refer to various types of inconsistencies in the user interface of the system including inconsistencies in the screen layouts, user interface navigation methods, fonts, and data formats in various screens and reports displayed by the system. In the subsystem under study, these forms of inconsistencies were considered to be detrimental to the usability of the application and as such any occurrences of such inconsistencies in the user interface were reported as defects. The data from our study suggest that:

*The lack of explicit and proactive definition and enforcement of implementation consistency rules leads to defects in cases, such as the user interface, where inconsistent implementations are considered defects.*

**The Impact of Software Size on Defect Rate.** The relationship between defect measures such as the defect count and defect density, and software size has been the subject of many studies in the literature. For example, [9], [19], [2], [21], [24], [1], [11], [20],

and [16] are some of the studies that have been conducted in this area. Table 4 in the related work section of this paper summarizes the key results from these nine studies. Some of the previous studies report a relationship between these parameters, while others do not. What makes the situation complicated is that the results from the studies where a relationship between these two parameters have been observed are conflicting. Some of these studies report a rising trend of defects as the size increases, while other studies report a declining trend of defects as the size grows. Others have tried to explain these rising and declining trends.

The data from our study does not suggest any noticeable dependence between defect rate and component size. 73.5% of all reported defects are attributed to the user interface component. In contrast to this high defect concentration, the business logic, and data access components have a share of only 14% and 12.5% of the total reported defects, respectively. Obviously, this noticeable difference between the defect distributions over the user interface component and the other two components cannot be attributed to the sizes of the components. Although the user interface component is almost the same size as the business logic component, and only twice the size of the data access component, it has at least five times more defects compared to each of the other two components. Since our data come from only three components, we cannot draw any conclusions about the dependence between the defect rate and component size.

We would like to study the relationship between defect measures and module size (as opposed to component size). Unfortunately, no data were available on the distribution of defects over the modules. An attempt to collect these data after the fact would be extremely time consuming and error prone.

## 2.5 Summary of the Results

Based on the analysis of the collected data, we can now answer the two research questions posed in the introduction.

1. What are the mechanisms that gave rise to defects in the case under study?
2. How large a role each identified mechanism has played in introducing defects in the case under study?

Table 2 summarizes the answers to these questions. The major conclusions and contributions of our study are presented below. These findings suggest areas where effort should be directed.

**Table 2.** Defect Introduction Mechanisms Identified in the Subsystem Under Study

Defect Introduction Mechanism	Defect Count	% of Defects
Incorrect implementations not linked to external causes	184	41
Incomplete requirements specifications	156	34.7
Adopting new, unfamiliar technology	47	10.5
Lack of traceability between requirements specifications	35	7.8
Lack of consistency in the user interface	27	6
<b>Total</b>	<b>449</b>	<b>100</b>

- A significant portion of the defects originate outside the source code. This finding is supported by the evidence that 59% of the reported defects were propagated into the considered subsystem from external sources including incomplete requirements specifications, adopting new, unfamiliar technologies, lack of requirements traceability, and the lack of proactive and explicit definition and enforcement of user interface consistency rules.
- Specification-related defects represent the largest category of defects (42.5%, of which 34.7% were caused by incomplete requirements specifications, and 7.8% were a result of a lack of traceability between various requirements specifications).

### 2.6 Implications of the Findings

Given the profile of defect sources identified in our case study, in Table 3, we propose a set of mitigation strategies that software development organizations can employ to counteract these defect introduction mechanisms.

**Table 3.** Defect Mitigation Strategies

Defect Source	Defect Mitigation Strategy
Incomplete requirements specifications	<ul style="list-style-type: none"> <li>• Improving the requirements process in terms of the completeness of the requirements specifications</li> <li>• Explicit documentation of all business rules and data validations</li> </ul>
Adopting new, unfamiliar technology	<ul style="list-style-type: none"> <li>• Thorough evaluation of new technologies before adopting them</li> <li>• Provision of sufficient training when a decision is made to adopt a new technology</li> </ul>
Lack of traceability between requirements	<ul style="list-style-type: none"> <li>• Practicing requirements traceability</li> <li>• Automated support for requirements process including tools to help the project team to keep track of the relationships between requirements and requirements status</li> </ul>
Lack of consistency in the user interface	<ul style="list-style-type: none"> <li>• Proactive definition and enforcement of user interface design rules to unify the implementation of user interface look and behavior</li> </ul>

### 2.7 Threats to Validity

Several factors potentially affect the validity of our findings. We discuss these factors under standard types of validity threats in empirical studies.

**Construct Validity.** The construct validity criterion questions whether the variables measured and studied truly reflect the phenomenon under study. We use defect count and defect density as surrogate measures for the quality of the software components. These measures are widely used in software engineering studies. All component and module sizes were measured in lines of non-commented and non-blank code. The purpose of the study was to identify some of the mechanisms that give rise to defects in software systems. Our observations involve the calculated measures of size, defect count (for various categories of defects), and defect density. Therefore, the variables measured and studied, truly reflect the purpose of the study.

**Internal Validity.** The internal validity of a study is concerned with distinguishing true causal relationships from those effected by confounding variables. In our study, the incorrect attribution of the defects to the components could be a threat to the internal validity of our findings. To minimize the potential effect of this confounding factor, the author of the paper and a member of the development team, who had a detailed knowledge of the system and had been actively involved in fixing the reported defects, independently analyzed the defects and attributed them to their corresponding components. For the great majority of the defects, we were able to accurately assign the defects to their corresponding components. The accuracy of the assignment of defects to the components was evident from the closely matching results of the two analyses. The cases where there was a difference in the results of the two analyses were jointly reexamined and resolved. We are highly confident that the attribution of defects to their corresponding components was accomplished accurately.

We studied the entire population of the reported defects in the subsystem under study. This effectively eliminated any potential sampling bias, which can be a problem for studies where a selected sample of the population is included in the study. Furthermore, the inclusion of the complete set of defects in our study not only helped us to obtain a complete picture of the defects and their root causes, but also increased the reliability of the conclusions drawn from the analysis of the data.

In studies where components developed in multiple languages are involved, an *equivalent code* must be calculated for the components, to make the comparison of the component sizes meaningful. The software tool used in our study to measure the component sizes provides an equivalent code size in a hypothetical third generation programming language. To calculate the equivalent sizes, the software tool multiplies the component size in Java with 1.36, Javascript with 1.48, JSP with 1.48, and XML with 1.90. We recalculated all the metric data collected in our study using the equivalent sizes. The results did not change any of our findings or conclusions.

To further validate our findings, we discussed them with the team members during our interviews and group sessions. They believe that the five findings of the study, presented in Section 2.4 of this paper, capture an accurate portrayal of their situation.

**External Validity.** The external validity of a study is concerned with the extent to which the findings of the study can be generalized. Our dataset was taken from one product of a development organization, which can be a limit to the generalizability of our findings. The results of our study can be considered as early evidence for some of the mechanisms that give rise to defects in software systems. Gaining more confidence in the results of the present study requires further replication of the study with other systems within and outside the considered organization. This is planned as future work.

### 3 Related Work

The work of Leszak et al. [13] is similar to ours in intent. They conducted a root cause analysis study of defects in a large transmission network element software, and based on the findings of the study devised countermeasures to either prevent the defects or detect them earlier in the development process. Results from our study contradict their

findings. They concluded that the majority of defects do not originate in early phases. They report that in the system they studied defects were introduced into the system predominantly (71%) within the component-oriented phases of component specification, design, and implementation. In contrast to our study results, in their case, requirements-related defects did not have a significant contribution to the total number of defects.

Eldh et al. [8] studied and classified the failures in a large complex telecommunication middleware system. They concluded that faults related to unclear specifications (46.6%) dominates among the software faults. In their case, faults of omission and spurious faults accounted for 38.3% and 8.3% of the total faults, respectively. Our study agrees with their finding. In our case, we also observed that specification-related defects represent the largest category of defects (42.5%). Our work is different from theirs both in its motivation and the focus of the study. Our goal is to identify the causes of defects so that appropriate actions can be initiated to prevent the sources of defects. To fulfill our goal, we performed root cause analysis of the defects and traced the defects to their sources outside the code and into the external factors (e.g., specifications, processes, and decisions). In contrast, the main motivation behind the classification of faults in Eldh et al.'s work is to investigate software test techniques through injecting the identified classes of faults into code. As a result, in contrast to our focus on the external root causes of the defects, their focus is on the static origin of the faults within the code.

From a research methodology point of view, our data selection approach is different from both Eldh et al. [8] and Leszak et al.'s [13] study. In our study, the entire population of defects in the considered subsystem was included for analysis. In comparison, the data used in Eldh et al.'s [8] study were selected by convenience sampling. they selected their sample data set from the defects whose labels in the configuration management system allowed them to trace the failures back to their origins in the system's source code. Leszak et al.'s [13] used a combination of manual and random sampling. Another distinction between our study and the two previous studies is that our data is taken from an ERP system, which is a business software system, whereas both previous studies collected data from system-level software products namely, telecommunication middleware software and transition network element software.

Chillarege et al. [5] propose a semantic classification of defects called Orthogonal Defect Classification (ODC), which is comprised of eight distinct defect types, each capturing the meaning of a type of defect fix. The distribution of defects over the ODC

**Table 4.** Summary of Observations from the Previous Studies on Defect Measures and Size

Study	Results
Fenton and Ohlsson [9]	(a) No significant relation between fault density and module size. (b) A weak correlation between module size and the number of pre-release faults. (c) No correlation between module size and the number of post-release faults.
Ostrand and Weyuker [19]	(a) Fault density slowly decreases with size. (b) Files including a high number of faults in one release, remain high-fault in later releases. (c) Newer files have higher fault density than older files.
Basili and Perricone [2]	Larger modules are less error prone, even when they are more complex in terms of cyclomatic complexity.
Shen et al. [21]	(a) Of 108 modules studied, for 24 modules with sizes exceeding 500 LOC, the size does not influence the defect density. (b) For the remaining 84 modules, defect density declines as the size grows.
Withrow [24]	A minimum defect density for modules with sizes between 161 and 250 LOC, after which the defect density starts increasing with module size.
Banker and Kemerer [1]	Proposed a hypothesis that for any given environment, there is an optimal module size. For lesser sizes, there is rising economy, and for greater sizes, the economy declines due to rising number of communication paths.
Hatton [11]	(a) For sizes up to 200 LOC, the total number of defects grows logarithmically with module size, giving a declining defect density. (b) For larger modules, a quadratic model is suggested.
Rosenberg [20]	Argued that the observed phenomenon of a declining defect density with rising module sizes is misleading.
Malaiya and Denton [16]	(a) Proposed that there are two types of defects: module-related defects, and instruction-related defects. (b) Module-related defects decline with growing module size. (c) The number of instruction-related defects rises with growing module size. (d) An optimal module size for minimum defect density is identified.

classes changes with time, which provides a measure of the progress of the product through the process. The main motivation for ODC is to provide feedback to developers during the development process. In contrast to ODC's focus on providing in-process feedback, the classes of defects identified by our study, along with the observed distribution of defects over these classes can provide after-the-fact feedback to developers, which can be used to improve the development of the next subsystems within the organization. In this sense, a study like ours can serve as a means for spreading lessons learned between projects. Given the qualitative nature of performing root cause analysis of defects, the resources required to perform the analysis are significant, which might make it impractical for providing in-process feedback.

There have been several studies about the relationship between defect-based measures such as defect count and defect density, and software size. Table 4 summarizes the findings of these studies.

## 4 Conclusion and Future Work

Defect prevention is a possible way to reduce software maintenance costs. However, to devise tools, techniques, and processes to support defect prevention requires an understanding of the mechanisms that give rise to defects during the development process. Case studies of real-world industrial systems are a systematic approach towards gaining such an insight. The study reported in this paper is meant to serve such a purpose.

Our case study identified four possible defect introduction mechanisms including incomplete requirements specifications, adopting new, unfamiliar technologies, lack of traceability of requirements, and the lack of explicit definition of user interface consistency rules that collectively account for 59% of the defects in the subsystem under study. These four defect introduction mechanisms are all well understood, which suggests that the cause of a significant portion of defects in industrial software projects is not a lack of knowledge, but rather a lack of application of existing knowledge.

In our future work, we intend to replicate this study with the other subsystems in the considered organization, as well as with ERP systems in other organizations in order to determine if patterns of defect introduction mechanisms exist among ERP systems. This should also give us insights into the factors affecting the magnitude of the defects introduced by each identified mechanism.

For the remaining 41% of the reported defects, no external contributing factors could be found. We intend to investigate this group of defects in future work. We plan to analyze historical data from the revision control system to track and analyze the changes made to the modules to fix the defects in order to understand the nature of these defects.

## References

1. Banker, R.D., Kemerer, C.F.: Scale Economics in New Software Development. *IEEE Transactions on Software Engineering*, 1199–1205 (October 1989)
2. Basili, V.R., Perricone, B.R.: Software Errors and Complexity. *Communications of ACM* 27, 42–45 (1984)
3. Bersoff, E., Henderson, V., Siegel, S.: *Software Configuration Management*. Prentice-Hall, Englewood Cliffs (1980)

4. Boehm, B.W.: Software and its Impacts: A Quantitative Assessment. *Datamation* 9, 48–59 (1973)
5. Chillarege, R., Bhandari, I.S., Chaar, J.K., Halliday, M.J., Moebus, D.S., Ray, B.K., Wong, M.: Orthogonal Defect Classification - A Concept for In-Process Measurements. *IEEE TSE* 18(11), 943–956 (1992)
6. Cleland-Huand, J., Chang, C.K., Christensen, M.: Event-Based Traceability for Managing Evolutionary Change. *IEEE TSE* 29(9), 1226–1242 (2003)
7. Devanbu, P., Brachman, R.J., Selfridge, P.G., Ballard, B.W.: LaSSIE: A Knowledge-Based Software Information System. *Com. of ACM* 34(5), 34–49 (1991)
8. Eldh, S., Punnekkat, S., Hansson, H., Jonsson, P.: Component Testing Is Not Enough - A Study of Software Faults in Telecom Middleware. In: Petrenko, A., Veanes, M., Tretmans, J., Grieskamp, W. (eds.) *TestCom/FATES 2007*. LNCS, vol. 4581, pp. 74–89. Springer, Heidelberg (2007)
9. Fenton, N.E., Ohlsson, N.: Quantitative Analysis of Faults and Failures in a Complex Software System. *IEEE Transactions on Software Engineering* 26(8), 797–814 (2000)
10. Fjelstad, R.K., Hamlen, W.T.: Application Program Maintenance Study - Report to Our Respondents. Technical Report, IBM Corporation, DP Marketing Group (1986)
11. Hatton, L.: Reexamining the Fault Density-Component Size Connection. *IEEE Software*, 89–97 (March 1997)
12. Hofer, A., Tichy, W.F.: Status of Empirical Research in Software Engineering. In: *Empirical Software Engineering Issues*, pp. 10–19. Springer, Heidelberg (2007)
13. Leszak, M., Perry, D.E., Stoll, D.: A Case Study in Root Cause Defect Analysis. In: *Proceedings of International Conference on Software Engineering, ICSE 2000*, pp. 428–437 (2000)
14. Lientz, B.P., Swanson, E.B., Tompkins, G.E.: Characteristics of Application Software Maintenance. *Communications of the ACM* 21(6), 466–471 (1978)
15. Lientz, B.P., Swanson, E.B.: *Software Maintenance Management*. Addison-Wesley, Reading (1980)
16. Malaiya, K.Y., Denton, J.: Module Size Distribution and Defect Density. In: *Proceedings of ISSRE 2000*, pp. 62–71 (2000)
17. Mohagheghi, P., Conradi, R., Killi, O.M., Schwarz, H.: An Empirical Study of Software Reuse vs. In: *Defect-Density and Stability*. In: *Proceedings of ICSE 2004*, pp. 282–292 (2004)
18. Nozek, J.T., Palvia, P.: Software Maintenance Management: Changes in the Last Decade. *Journal of Software Maintenance: Research and Practice* 2(3), 157–174 (1990)
19. Ostrand, T.J., Weyuker, E.J.: The Distribution of Faults in a Large Industrial Software System. In: *Proceedings of ISSTA 2002*, pp. 55–64 (2002)
20. Rosenberg, J.: Some Misconceptions about Lines of Code. In: *Proceedings of the International Software Metrics Symposium, November 1997*, pp. 137–142 (1997)
21. Shen, V.Y., Yu, T., Thebut, S.M.: Identifying Error-Prone Software- An Empirical Study. *IEEE Transactions on Software Engineering* 11, 317–324 (1985)
22. Vliet, H.V.: *Software Engineering: Principles and Practices*. John Wiley & Sons, Chichester (2000)
23. Tichy, W.F.: Should Computer Scientists Experiment More? *IEEE Computer* 31(5), 32–40 (1998)
24. Withrow, C.: Error Density and Size in Ada Software. *IEEE Software*, 26–30 (1990)
25. Whitney, R., Nawrocki, E., Hayes, W., Siegel, J.: Interim Profile: Development and Trial of a Method to Rapidly Measure Software Engineering Maturity Status. Technical Report, CMU/SEI-94-TR-4, ESC-TR-94-004, March 26-30 (1994)
26. <http://www.sei.cmu.edu/index.html>

# Measuring and Comparing Effectiveness of Data Quality Techniques

Lei Jiang<sup>1</sup>, Daniele Barone<sup>2</sup>, Alex Borgida<sup>1,3</sup>, and John Mylopoulos<sup>1,4</sup>

<sup>1</sup> Dept. of Computer Science, University of Toronto

<sup>2</sup> Dept. of Computer Science, Università di Milano Bicocca

<sup>3</sup> Dept. of Computer Science, Rutgers University

<sup>4</sup> Dept. of Information Engineering and Computer Science, University of Trento

**Abstract.** Poor quality data may be detected and corrected by performing various quality assurance activities that rely on techniques with different efficacy and cost. In this paper, we propose a quantitative approach for measuring and comparing the effectiveness of these data quality (DQ) techniques. Our definitions of effectiveness are inspired by measures proposed in Information Retrieval. We show how the effectiveness of a DQ technique can be mathematically estimated in general cases, using formal techniques that are based on probabilistic assumptions. We then show how the resulting effectiveness formulas can be used to evaluate, compare and make choices involving DQ techniques.

**Keywords:** data quality technique, data quality measure, data quality assurance.

## 1 Introduction

The poor quality of data constitutes a major concern world-wide, and an obstacle to data integration efforts. Data of low quality may be detected and corrected by performing various quality assurance activities that rely on techniques with different efficacy and cost under different circumstances. In some cases, these activities require additional data, changes in the database schema, or even changes in core business activities. For example, consider the relation schema *Person*(*sin*, *name*, *address*), which intends to record a person's social insurance number, name and address. Due to the decision to represent an address value as single string, no obvious integrity constraints or other automatically enforceable techniques can be specified on the components of the address value [1]. In particular, one cannot detect missing street, city, etc. using “not null” constraints, because nothing is said in the schema about the exact format of address values.

In [2], we proposed a goal-oriented database design process, and extended in [3] to handle data quality goals. The quality design process starts with a conceptual schema, which is then augmented by a set of high level *data quality goals* (e.g., “accurate student data”). These goals are gradually decomposed into concrete *data quality problems* to be avoided (e.g., no “misspelled student names”). For each such problem, a list of *risk factors* (i.e., potential causes) and *mitigation plans* (i.e., potential solutions) is presented. The main component of a mitigation plan is a *design proposal* consisting of a revised original schema and a set of data quality (DQ) techniques it supports.



In this paper, we take the next step of proposing a quantitative approach for measuring and comparing the effectiveness of DQ techniques used in quality assurance activities. The main contributions of this paper include: (i) the definitions of effectiveness measures for DQ techniques, based on the well-established notions of precision, recall and F-measure; (ii) formal techniques for estimating the expected effectiveness scores for a technique (on a wider range of possible instances of a database), based on probabilistic assumptions about the occurrence of errors in data values and confounding factors; these techniques result in effectiveness formulas parametrized by variables introduced by these assumptions; (iii) analysis and comparison of DQ techniques and their respective strengths in terms of the subranges of values of the parameters in the effectiveness formulas.

The rest of the paper is organized as follows. We first discuss briefly the main concepts in our approach in Section 2. We then present the definitions of our effectiveness measures in Section 3, and show examples of calculating effectiveness scores when a database instance is available. Next, a general pattern is identified for the formal estimation of the expected effectiveness scores based on the probabilistic assumptions, and is applied to several DQ techniques in Section 4.1. The resulting effectiveness formulas provide input for the what-if analysis in Section 4.2 in which we evaluate a single DQ technique and compare multiple ones under different scenarios. Finally, we review the related work in Section 5 and conclude and point out to our future work in Section 6.

## 2 Main Concepts

### 2.1 DQ Techniques

The core concept in our approach is a *DQ technique*, which is, broadly speaking, any automatic technique that can be applied to data in a quality assurance activity, in order to assess, improve and monitor its quality. This includes techniques to standardize data of different formats, to match and integrate data from multiple sources, and to locate and correct errors in data values [1]. In this paper, we focus on DQ techniques that automatically enforce a rule of the form “if *condition* then *action*”, where *condition* checks violation of some integrity constraint, and *action* produces either deterministic or probabilistic decisions regarding quality of data being examined (e.g., to mark values as possibly erroneous, to suggest possible corrections to erroneous values). Following example illustrates two simple rules.

**Example 1.** Consider the relation schema *Person* again. Suppose we are especially concerned with the quality of name values. In this case, we can modify this schema by adding a second name attribute as in *Person'*(*sin*, *name*, *address*, *name'*), with the intention of modifying the workflow so that names are entered twice (by the same or different persons), and then detect errors by comparing the two name entries. The revised schema makes it possible to specify and enforce following rule, “for each tuple *t* inserted in *Person'*, if  $t.name \neq t.name'$  then mark the tuple  $\langle t.name, t.name' \rangle$  as erroneous”. Another way to provide quality assurance for names, without changing the schema of *Person*, is to keep a table of valid names,  $L_{name}$ . This allows to specify and enforce the rule, “if  $\neg(t.name \in L_{name})$ , then mark *t.name* as erroneous”. □

## 2.2 Effectiveness of DQ Techniques

Different DQ techniques may have different efficacy and cost under different circumstances. The effectiveness of a DQ technique is determined both by the nature of the technique and the particular values and errors in the data being examined. Following example explains the concept of effectiveness in the context of DQ techniques.

**Example 2.** Consider a simple conditional functional dependency  $\phi = [\text{country-code} = 44, \text{area-code} = 131] \rightarrow [\text{city-name} = \text{Edinburgh}]$  [45]<sup>1</sup>. If it is the case that city name has much higher possibilities of having errors than country code and area code, violation of  $\phi$  is more likely an indication of erroneous city name values than others. In this case, a DQ technique may check for violation of  $\phi$ , and mark the city name value in a tuple as possibly erroneous whenever the tuple violates  $\phi$ . A set of tuples marked by this DQ technique then needs to be presented to a domain expert who will make the final decision. To minimize human effort, ideally a city name value is actually erroneous if and only if the tuple containing this value is in the returned set. However, this is unlikely to be true due to (comparably small amount of) errors in country code and area code values. Effectiveness of  $\phi$  measures its ability to produce “good” sets of tuples compared to the ideal set, for a given database instance or a range of instances.  $\square$

## 2.3 Effectiveness Measures, Scores and Formulas

With respect to a particular database instance, an *effectiveness score* is assigned to a DQ technique. To obtain such effectiveness scores, the first step is to adopt a set of *effectiveness measures*, such as precision, recall and F-measure from Information Retrieval. Then, the DQ technique is applied to a database instance (for which quality of data is already known, e.g., through manual assessment); and the effectiveness scores of the DQ technique is calculated by comparing its output with existing knowledge of the instance. There are several limitations for this approach. First, a database instance may not always be available (e.g., when designing a new schema) or only partially available (e.g., when modifying an existing schema). Second, the effectiveness scores only tell us how the DQ technique performs on one snapshot of the database.

Therefore, it is often necessary to consider how a DQ technique performs on average over a range of possible instances of the database. This leads to the *expected effectiveness score* of a DQ technique. To obtain expected effectiveness score of a DQ technique, one can first derive an *effectiveness formula* of the DQ technique. This requires making probabilistic assumptions about the occurrence of errors in data values and confounding factors. By *confounding factors*, we mean special events that may “confuse” a DQ technique, making it less effective. For example, the country name “Australia” may be misspelled as “Austria”, which is still a valid country name; such an error cannot be detected using a technique based on a country name lookup table. The resulting effectiveness formulas, can be evaluated and compared by fixing some parameters in formulas and allowing the others to vary.

---

<sup>1</sup> A conditional functional dependency is formally defined as a pair of a regular functional dependency and a pattern tableau. Here we are using an abbreviated notation.

### 3 Effectiveness Measures

In this section, we present the definitions for our effectiveness measures. We first explain the general idea and give the basic definitions for these measures, and then extend them to accommodate errors of different types and data values from multiple attributes.

#### 3.1 Basic Definition

Effectiveness represents the ability to produce a desired result (in order to accomplish a purpose). For DQ techniques, the purpose is to assess, improve, etc. quality of data. This gives rise to measures for assessability, improvability, etc. In what follows, we concentrate on assessability measures; we defer a detailed treatment of other types of effectiveness measures to a later report.

*Assessability* represents an DQ technique's ability to effectively detect erroneous data values. Normally, this ability can only be measured if we have access to the reality (i.e., is an erroneous value marked by a DQ technique really an error). When not accessible, we need an approximation of the reality, possibly obtained through some manual quality assurance activity. The precise meaning of "assessability" depends on what do we mean by "erroneous" and "data value". To begin with, we assume that each DQ technique assesses quality of data in a single attribute, and classifies the attribute values into two categories: those with some error, and those without. In Section 3.2, we relax these limitations.

Inspired by Information Retrieval, we define assessability measures in terms of precision, recall and F-measure [6]. More specifically, let  $S$  be a relation schema,  $A$  be an attribute in  $S$ , and  $I$  be an instance of  $S$ . Consider a DQ technique  $T$ , which is applied to  $I$  in order to assess quality of  $A$  values. Equation 1 and 2 defines the precision and recall [6] for  $T$  with respect to  $I$  and  $A$ ; these two measures are combined in Equation 3 into F-measure [6], where  $\beta$  is a constant that represents the importance attached to recall relative to precision.

$$precision(T, I, A) = \frac{TP(T, I, A)}{TP(T, I, A) + FP(T, I, A)} \quad (1)$$

$$recall(T, I, A) = \frac{TP(T, I, A)}{TP(T, I, A) + FN(T, I, A)} \quad (2)$$

$$F_{\beta}(T, I, A) = \frac{(1 + \beta^2) \times precision(T, I, A) \times recall(T, I, A)}{\beta^2 \times precision(T, I, A) + recall(T, I, A)} \quad (3)$$

The values  $TP$ ,  $FP$  and  $FN$  represent the number of true positives, false positives and false negatives respectively, and are explained more clearly below. Example 3 shows how the assessability scores can be calculated using the sampling approach.

- $TP(T, I, A)$  = the number of erroneous  $A$  values in  $I$ , correctly marked by  $T$  as being erroneous
- $FP(T, I, A)$  = the number of non-erroneous  $A$  values in  $I$ , incorrectly marked by  $T$  as being erroneous

- $FN(T, I, A)$  = the number of erroneous  $A$  values in  $I$ , not marked by  $T$  as being erroneous, but should have been

**Example 3.** Consider the *Person* schema again. Suppose from one of its instances,  $I_{Person}$ , 10 tuples are selected as the sample. After performing some manual quality assurance activity on the sample, 3 erroneous name values are identified and the correct values are obtained. Table 1(a) shows the result of this manual activity, where the name value is erroneous iff  $err = "1"$ ;  $name^{new}$  is used to record the suggested name values<sup>2</sup>.

Now consider a DQ design proposal  $P_1(Person'(sin, name, address, name'), T_{equal})$ , in which the *Person* schema is revised to *Person'*, and  $T_{equal}$  is a DQ technique that enforces the rule: “for each tuple  $t$  inserted in an instance of *Person'*, if  $t.name \neq t.name'$  then mark  $t.name$  as erroneous.” An instance  $I_{Person'}$  of *Person'* is generated by starting with data from  $I_{Person}$  and obtaining independent values for the new attribute  $name'$ .

Suppose we need to know how effective  $T_{equal}$  is in assessing *name* values in  $I_{Person'}$ . First, we select the same 10 tuples from  $I_{Person'}$  as the sample, and obtain the quality assessments on the sample using  $T_{equal}$ , as shown in column *err* of Table 1(b). By comparing Table 1(b) with Table 1(a), we obtain following numbers  $TP = 2$  (due to Tuple 006 and 009),  $FP = 2$  (due to Tuple 001 and 008), and  $FN = 1$  (due to Tuple 004). The assessability scores for  $T_{equal}$  on this sample (when  $\beta = 1$ ) are:  $precision(T_{equal}, I_{Person'}, name) = 0.5$ ,  $recall(T_{equal}, I_{Person'}, name) = 0.67$ , and  $F_1(T_{equal}, I_{Person'}, name) = 0.57$ .  $\square$

**Table 1.** Calculation of effectiveness scores using the sampling approach

(a) Quality of *name* values in  $I_{Person}$

sin	name	err	name <sup>new</sup>
001	Kelvin	0	
002	Michelle	0	
003	Jackson	0	
004	Alexander	1	Alexandre
005	Maria	0	
006	Tania	1	Tanya
007	Andrew	0	
008	Christopher	0	
009	Michale	1	Michael
010	Matthew	0	

(b) DQ annotation for *name* values in  $I_{Person'}$  using  $T_{equal}$

sin	name	name'	err
001	Kelvin	Kelvn	1
002	Michelle	Michelle	0
003	Jackson	Jackson	0
004	Alexander	Alexander	0
005	Maria	Maria	0
006	Tania	Tanya	1
007	Andrew	Andrew	0
008	Christopher	Christophor	1
009	Michale	Michael	1
010	Matthew	Matthew	0

(c) DQ annotation for *name* values in  $I_{Person'}$  using  $T_{equal-prob}$

sin	name	name'	err	err'
001	Kelvin	Kelvn	0.5	0.5
002	Michelle	Michelle	0	0
003	Jackson	Jackson	0	0
004	Alexander	Alexander	0	0
005	Maria	Maria	0	0
006	Tania	Tanya	0.5	0.5
007	Andrew	Andrew	0	0
008	Christopher	Christophor	0.5	0.5
009	Michale	Michael	0.5	0.5
010	Matthew	Matthew	0	0

## 3.2 Extensions

We may be interested in measuring the effectiveness of a DQ technique with respect to particular types of errors, instead of considering all possible ones. For example, a lookup-table based DQ technique is very effective in detecting syntactic but not semantic accuracy errors [11]. In this case, the assessability scores can be calculated using Equation 1 and 2 in the same way as before, except that we only consider errors of the specified types when counting  $TP$ ,  $FP$  and  $FN$ .

<sup>2</sup> The *address* values are omitted here and thereafter.

Equation 1 and 2 work for DQ techniques whose output involve a single attribute. In some case, the result of a DQ technique may involve values of a set  $X = \{A_1, \dots, A_n\}$  of attributes. There are two ways to look at this situation, which lead to two different solutions. In one view, we may treat a tuple  $t.X$  as a single value (i.e.,  $t.X$  is erroneous if any of  $t.A_1, \dots, t.A_n$  is). Then we can calculate assessability scores of a DQ technique using modified versions of Equation 1 and 2 where  $precision(T, I, A)$  and  $recall(T, I, A)$  are replaced with  $precision(T, I, X)$  and  $recall(T, I, X)$  respectively. In another view, we introduce the notion of uncertainty. This leads to a more general solution. When a DQ technique marks a tuple  $t.X$  as being erroneous, it essentially marks each individual value  $t.A_1, \dots, t.A_n$  in the tuple as being erroneous with certain probability. If those probabilities can be estimated, we can still treat each attribute individually, but allow the assessment result to be a number between 0 and 1. Example 4 illustrates the second view.

**Example 4.** Let us consider another DQ design proposal  $P_2(Person'(sin, name, address, name'), T_{equal-prob})$ , where  $T_{equal-prob}$  is same as  $T_{equal}$  in  $P_1$ , except that it marks the whole tuple  $t[name, name']$  as being erroneous when  $t.name \neq t.name'$ . Following the second view, if we assume that a  $name$  and  $name'$  value have the same probability of being wrong, Table 1(c) shows the output of  $T_{equal-prob}$  applied to the same sample of  $I_{Person'}$  (as in Example 3). Notice,  $err = "0.5"$  (respectively  $err' = "0.5"$ ) means the  $name$  (respectively  $name'$ ) value is marked as erroneous with the 0.5 probability.

In this case, a real erroneous  $name$  value, being marked as erroneous with 0.5 probability (e.g., Tuple 006), counts for 0.5 toward  $TP$  and 0.5 toward  $FN$ . By comparing this table with Table 1(a), we can obtain following numbers:  $TP = 1$  (due to Tuple 006 and 009),  $FP = 1$  (due to Tuple 001 and 008) and  $FN = 2$  (due to Tuple 004, 006 and 009). The assessability scores for  $T_{equal-prob}$  on this sample can then be calculated using this numbers.  $\square$

## 4 Estimating and Comparing Expected Effectiveness Scores

### 4.1 Formal Approach

The above examples show the calculation of assessability scores for a DQ technique on a particular database instance. In this section, we show how assessability scores can be estimated without applying the DQ technique to data. More specifically, we show how to obtain the expected assessability scores for a DQ technique based on probabilistic assumptions. This approach can be divided into four steps: (1) setting the stage, (2) making probabilistic assumptions, (3) calculating probabilities for the events of interests, and (4) formulating assessability scores. In what follows, we illustrated this approach on several DQ techniques.

**Introducing Redundancy.** Although duplicating an attribute as we have shown in previous examples may seem simplistic, the idea of using redundancy checks (e.g., checksum) to protect the integrity of data has long been practiced in computer communication, and also been proposed for detecting corruption in stored data [7]. More generally, partial redundancy is the basis of many integrity constraints (e.g., correlations between phone area codes and postal codes).

*Step 1: setting the stage.* In general, given a relation schema  $S$ , we are interested in DQ design proposals of the form  $P_{redundancy}(S', T_{B \neq f(X)})$ , where  $S'$  contains all attributes in  $S$  plus a new attribute  $B$ , and  $T_{B \neq f(X)}$  enforces the rule “for each tuple  $t$  inserted in an instance of  $S'$ , if  $t.B \neq f(t.X)$  then mark  $t.X$  as erroneous”; here  $X$  is a subset of attributes in  $S$ , and  $f$  represents some computable function. For example,  $X$  may contain a single attribute *birthdate* and  $B$  is the attribute *age*;  $f$  computes the current age from the date of birth<sup>3</sup>. In what follows, we illustrate the formal approach for the case where  $X$  contains a single attribute  $A$  and  $f$  is the identity function, i.e., for the DQ technique  $T_{B \neq A}$ . More general cases can be handled in a similar way.

*Step 2: making probabilistic assumptions.* The main factor that affects the assessability scores for  $T_{B \neq A}$  is the occurrence of errors in the attributes  $A$  and  $B$ . For the rest of the paper, we make several independence assumptions about values and errors in general: (i) the probability that a value will be wrong is independent of the value itself, and (ii) the probability of an error occurring in one attribute is independent of those of the other attributes.

To simplify the analysis here, we will assume that the probability of a  $A$  value or  $B$  value being incorrect is the same — denoted by  $p$ . If we use  $Err^{t.A}$  to name the event that the recorded value in  $t.A$  does not correspond to the real one and use  $Cor^{t.A}$  to mean the converse, this assumption can be stated symbolically as  $\text{pr}(Err^{t.A}) = \text{pr}(Err^{t.B}) = p$ , where  $\text{pr}(E)$  represents the probability of an event  $E$ . Before we proceed further, we need to recognize that there is the possibility that both  $t.A$  and  $t.B$  are incorrect yet contain the same erroneous value; in this case, these errors “cancel out” as far as the DQ technique  $T_{B \neq A}$  is concerned (since they cannot be detected by  $T_{B \neq A}$ ). We call this situation “error masking”, which is a particular type of confounding factors. Let us say that such masking will happen only with probability  $1 - c_1$ .

*Step 3: calculating probabilities for the events of interests.* To estimate the assessability scores, we are interested in events concerning a tuple  $t$  (i) whether  $t.A$  has an error, and (ii) whether a DQ problem is signaled by  $T_{B \neq A}$ . This estimation has to be adjusted for error masking. To compute the expected values for  $TP$ ,  $FP$  and  $FN$ , we will actually compute the probabilities of events concerning a particular tuple  $t$ , and then multiply this by the number of tuples in the relation.

First, true positives occur when  $t.A$  has an error (probability  $p$ ) that is correctly signaled by  $T_{B \neq A}$ . This happens when either  $t.B$  is correct (prob.  $(1 - p)$ ) or  $t.B$  is incorrect (prob.  $p$ ) but different from  $t.A$  (prob.  $c_1$ ); this yields probability:  $\text{pr}(Err^{t.A} \wedge Cor^{t.B}) + \text{pr}(Err^{t.A} \wedge Err^{t.B} \wedge (t.A \neq t.B)) = p \times (1 - p) + p \times p \times c_1$ .

False negatives occur when  $t.A$  has an error that is not signaled by  $T_{B \neq A}$ , because error masking occurs (which requires  $t.B$  to contain the exact same error); this has probability:  $\text{pr}(Err^{t.A} \wedge Err^{t.B} \wedge (t.A = t.B)) = p \times p \times (1 - c_1)$ .

False positives occur when  $t.A$  has no error yet  $T_{B \neq A}$  signals a problem, which arises according to our rule when  $t.B \neq t.A$  (i.e., when  $t.B$  has an error); this has probability:  $\text{pr}(t.A^{cor} \wedge t.B^{err}) = (1 - p) \times p$ .

<sup>3</sup> A variant of  $T_{B \neq f(X)}$  replaces the condition “ $t.B \neq f(t.X)$ ” with “ $d(t.B, f(t.X)) > \delta$ ”; so instead of requiring  $t.B$  and  $f(t.X)$  to be exactly the same, it only requires their distance (measured by  $d$ ) be less than a constant  $\delta$ .

*Step 4: formulating assessability scores.* Given the probabilities obtained in Step 3, the expected number of true positives, false positives and false negatives can be calculated as the number of tuples (say  $N$ ) times the respective probability as following:  $TP(T_{B \neq A}, A) = N \times (p(1-p) + p^2 c_1)$ ;  $FN(T_{B \neq A}, A) = N \times p^2(1-c_1)$ ;  $FP(T_{B \neq A}, A) = N \times (1-p)p$ ; The expected assessability scores for  $T_{B \neq A}$  can then be obtained by plugging these numbers into Equation 1, 2 and 3. Since  $N$  appears both in the numerator and denominator, it will cancel out, resulting in the effectiveness formulas in Table 2 (Section 4.2).

**Using Lookup Tables.** For an attribute with a standardized (and finite) domain, such as country name or postal code, a common DQ technique is to check its values against a lookup table for the attribute. Attributes with enumerated value domains (such as *gender*) also offer this possibility.

*Step 1: setting the stage.* Given the original schema  $S$  and an attribute  $A$  in  $S$ , we are interested in DQ design proposals of the form  $P_{lookup}(S, T_{L_A})$ , where  $T_{L_A}$  is the DQ technique that detects errors in  $A$  values using a lookup table  $L_A$ . In what follows, we illustrate the formal approach for this type of DQ techniques.

*Step 2: making probabilistic assumptions.* We make two passes through this analysis, in order to account for two different sources of problems. First, we assume as before there is a probability  $p$  that the recorded value of  $t.A$  is incorrect. In this case, error masking occurs when this erroneous value is still a valid value in the domain of  $A$  (e.g., “Australia” vs “Austria”) – an event to which we assign probability  $c_2$ . If we use  $Valid^{t.A}$  to name the event that the value  $t.A$  is valid and  $Invalid^{t.A}$  to mean the converse, we can represent these assumptions using following conditional probabilities:  $\text{pr}(Valid^{t.A} | Err^{t.A}) = c_2$  and  $\text{pr}(Invalid^{t.A} | Err^{t.A}) = 1 - c_2$ .

Second, we consider the possibility of the lookup table being imperfect, which is another type of confounding factors. In particular, we allow a probability  $s$  that some value (e.g., the name of a newly independent country) is missing from the lookup table  $L_A$ . If we use  $L_A^{t.A}$  to name the event that the value  $t.A$  is contained in  $L_A$ , and  $L_A^{-t.A}$  to mean the converse, we have  $\text{pr}(L_A^{t.A}) = 1 - s$  and  $\text{pr}(L_A^{-t.A}) = s$ . Notice here we are implicitly assuming that  $\text{pr}(L_A^{t.A})$  is independent from the characteristics of  $t.A$  values (e.g., *name* values of different length or in different languages).

*Step 3: calculating probabilities for the events of interests.* In the first case (i.e., assuming a perfect lookup table), true positives occur when  $t.A$  is incorrect and the value is not in the lookup table  $L_A$  (therefore  $t.A$  must be invalid, since all valid values are in  $L_A$ ); this has probability:  $\text{pr}(Err^{t.A} \wedge Invalid^{t.A}) = \text{pr}(Err^{t.A}) \times \text{pr}(Invalid^{t.A} | Err^{t.A}) = p \times (1 - c_2)$ .

False negatives occur when the error is masked (i.e., when  $t.A$  is incorrect but happens to be valid, and therefore is in  $L_A$ ); this has probability  $\text{pr}(Err^{t.A} \wedge Valid^{t.A}) = \text{pr}(Err^{t.A}) \times \text{pr}(Valid^{t.A} | Err^{t.A}) = p \times c_2$ .

Finally, in this case, there can be no false positives: every  $A$  value not in  $L_A$  is an incorrect  $A$  value.

<sup>4</sup> A more thorough, but complex, analysis would allow errors in the table values themselves or extra/out of date values.



In the second case (i.e., assuming a imperfect lookup table), false positives show up when  $t.A$  is correct, yet the value is missing from  $L_A$ ; this has probability:  $\text{pr}(Cor^{t.A} \wedge L_A^{-t.A}) = (1 - p) \times s$ .

For true positives, another source is possible, i.e., when an incorrect  $t.A$  value is valid (due to error masking), but is accidentally missing from  $L_A$ ; the total probability for true positives is therefore the one obtained in the first case plus following probability:  $\text{pr}(Err^{t.A} \wedge Valid^{t.A} \wedge L_A^{-t.A}) = \text{pr}(Err^{t.A} \wedge Valid^{t.A}) \times \text{pr}(L_A^{-t.A}) = (p \times c_2) \times s$ .

For false negatives, we need to multiply the probability obtained in the first case by  $(1 - s)$ , since they require the masking values also be in  $L_A$ .

*Step 4: formulating assessability scores.* Given the probabilities we obtained in Step 3, the expected assessability scores for  $T_{B \neq A}$  can be calculated in the same way as for the case of  $T_{B=A}$ . See Table 2 (Section 4.2) for the resulting effectiveness formulas for  $T_{L_A}$ .

### 4.2 What-If Analysis

The results of the formal approach are formulas representing the expected assessability scores for DQ techniques. These formulas are useful for several reasons. First, they identify conditions (e.g., parameters  $p$  and  $s$  in Table 2) that affect the effectiveness of a DQ technique. Second, as we show below, they allow us to perform trade-off analysis concerning different scenarios that involve one or more DQ techniques. (Each scenario produces a plot of effectiveness scores by fixing most parameters and allowing the others to vary.)

The formulas that represent expected precision, recall and F-measure (when  $\beta = 1$ ) for the DQ techniques  $T_{B \neq A}$  and  $T_{L_A}$ , together with a summary of the parameters used in these formulas, are shown in Table 2. In what follows, we first show how these two techniques are evaluated individually (in Scenarios 1 - 4) and then show how they are compared with each other (in Scenarios 5 - 10).

**Scenarios 1 - 4: Evaluating Individual DQ Techniques.** Scenarios 1 and 2 consider the impact of “error masking” (varying  $c_1$ ) on the effectiveness of  $T_{B \neq A}$ , while Scenarios 3

**Table 2.** Expected assessability scores for  $T_{B \neq A}$  and  $T_{L_A}$

<b>Technique:</b> $T_{B \neq A}$	<b>Technique:</b> $T_{L_A}$
<b>Assessability Scores:</b>	<b>Assessability Scores:</b>
$precision(T_{B \neq A}, A) = \frac{1+(c_1-1)p}{2+(c_1-2)p}$	$precision(T_{L_A}, A) = \frac{1+(s-1)c_2}{s/p+(s-1)(c_2-1)}$
$recall(T_{B \neq A}, A) = 1 + (c_1 - 1)p$	$recall(T_{L_A}, A) = 1 + (s - 1)c_2$
$F_1(T_{B \neq A}, A) = \frac{2+2(c_1-1)p}{3+(c_1-2)p}$	$F_1(T_{L_A}, A) = \frac{2+2(s-1)c_2}{1+s/p+(s-1)(c_2-1)}$
<b>Parameters:</b>	
$p$ : the probability that an $A$ value is erroneous	
$c_1$ : the probability that both $A$ and $B$ values in a tuple are erroneous, but contain different errors	
$c_2$ : the probability that an erroneous $A$ value is valid in the domain of $A$	
$s$ : the probability that a valid $A$ (with or without error) is not contained in the lookup table $L_A$	



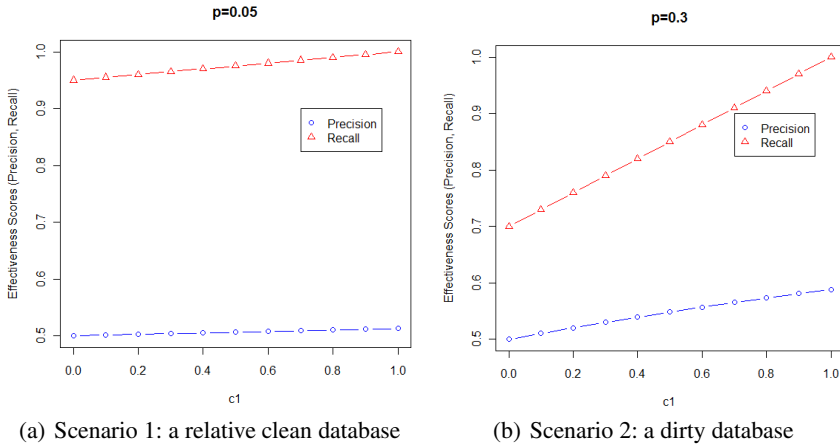


Fig. 1. Evaluation of  $T_{B \neq A}$

and 4 consider the impact of  $L_A$ 's "coverage" (varying  $s$ ) on the effectiveness of  $T_{L_A}$ . For each technique, the evaluation is carried out with respect to a relatively clean database ( $p = 0.05$ , in Scenarios 1 and 3) and a dirty database, ( $p = 0.3$ , in Scenarios 2 and 4).

The results for Scenarios 1 and 2, as given in Figure 1(a) and 1(b), show that the precision and recall of  $T_{B \neq A}$  decrease when the chance of "error masking" increases (i.e., as  $c_1$  decreases). This corresponds to our intuition. However a comparison of these two figures also reveals that, in a dirty database (i.e., with a larger  $p$ ), the effectiveness of  $T_{B \neq A}$  decreases *more precipitously* as the chance of "error masking" increases. For example, as  $c_1$  decreases from 1 to 0, the recall of  $T_{B \neq A}$  decreases by only 0.05 in the clean database, but by 0.3 in the dirty database.

The results for Scenarios 3 and 4 are shown in Figure 2(a) and 2(b) respectively. In both cases, as the "coverage" of the lookup table decreases (i.e., as  $s$  increases), we notice an intuitively expected decrease in precision; however, the dramatic nature of its drop is not so easily predicted by intuition, and is therefore a benefit of this analysis. We also note that recall is much less affected by the "coverage". Moreover, by comparing these two figures, we observe that the probability of errors in  $A$  has much greater impact on precision than on recall. More specifically, the recall of  $T_{L_A}$  remains the same when comparing the clean and dirty databases; however, in the dirty database, the precision decreases considerably slower as the "coverage" decreases. For example, when  $s$  increases from 0 to 1, the precision of  $T_{L_A}$  decreases by 0.95 in the clean database, and by only 0.7 in the dirty database.

**Scenarios 5 and 6: Comparing DQ Techniques - The Impact of Errors.** In this subsection, we compare DQ techniques  $T_{B \neq A}$  and  $T_{L_A}$  in two scenarios, by investigating the impact of the probability of errors in  $A$  (varying  $p$ ) on the effectiveness of these two techniques in an optimistic and a pessimistic setting. In the optimistic case, the chance of "error masking" is very small and the "coverage" of the lookup table is nearly perfect. More specifically, we assume that (i) in 99% of the cases, erroneous  $A$  and  $B$  values in a tuple contain different errors (i.e.,  $c_1 = 0.99$ ), (ii) only 1% of erroneous  $A$

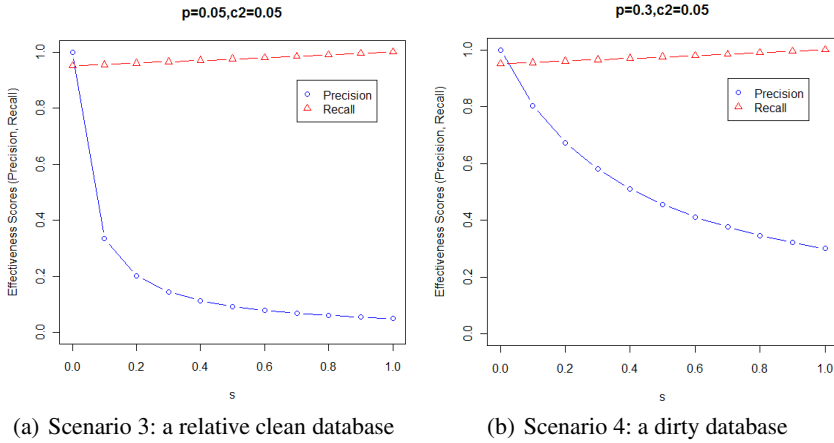


Fig. 2. Evaluation of  $T_{L_A}$

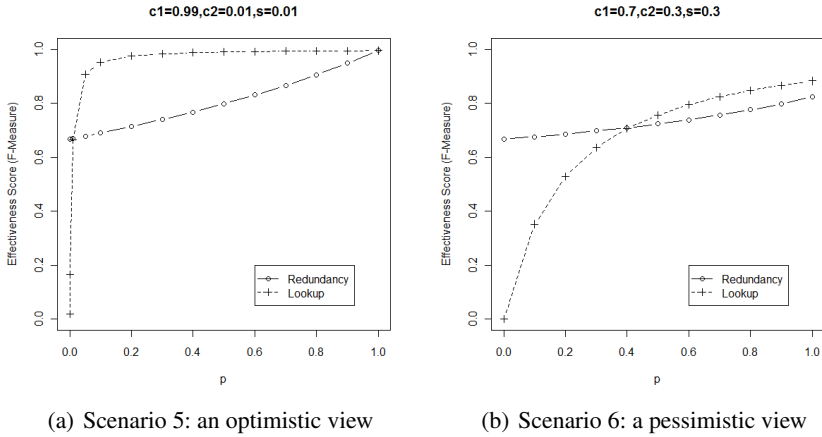


Fig. 3. Comparison of  $T_{B \neq A}$  and  $T_{L_A}$  - Impact of Errors

values happen to be other valid values (i.e.,  $c_2 = 0.01$ ), and (iii) only 1% of the valid  $A$  values are not contained in the lookup table  $L_A$  (i.e.,  $s = 0.01$ ). In the pessimistic case, we significantly increase the chance of “error masking” and decrease the “coverage” of the lookup table. More specifically, we set  $c_1 = 0.70$ ,  $c_2 = 0.30$  and  $s = 0.30$ . Figure 3(a) and 3(b) compare the F-Measures of  $T_{B \neq A}$  and  $T_{L_A}$  in these scenarios.

We observe that in both settings, the F-measure of  $T_{B \neq A}$  increases as the number of erroneous  $A$  values increases (i.e.,  $p$  increases). A similar pattern can be observed for  $T_{L_A}$  in the pessimistic setting; in the optimistic setting, the F-measure of  $T_{L_A}$  increases dramatically when  $p < 0.05$ , and remains almost constant when  $p \geq 0.05$ . These two figures suggest under what circumstances one DQ technique is preferable to the other one. More specifically, in an optimistic world,  $T_{B \neq A}$  outperforms  $T_{L_A}$  only when the probability of erroneous  $A$  values is quite small (i.e., when  $p < 0.01$ ), while in a

pessimistic world,  $T_{B \neq A}$  is a more effective choice than  $T_{L_A}$  as long as the error rate in  $A$  is less than 40% (i.e., when  $p < 0.4$ ).

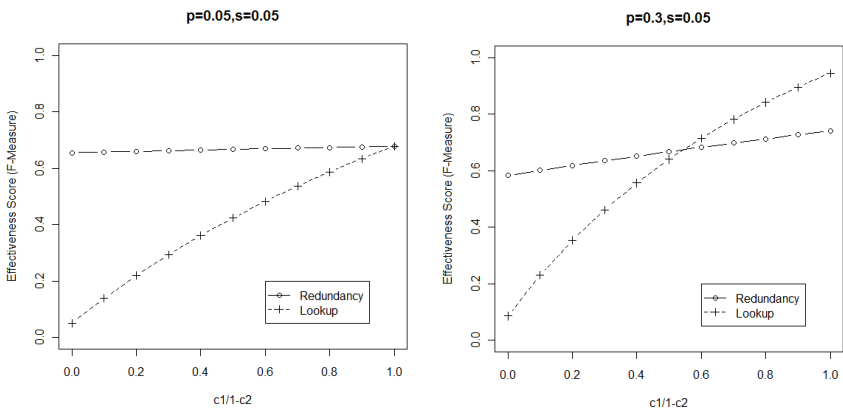
A briefer summary might be that in a typical situation, where the chance of “error masking” is reasonably small (say, less than 5%) and the “coverage” of lookup table is nearly perfect (say, more than 95%), a lookup-table based DQ technique is generally more effective in detecting errors than a redundancy-check based DQ technique, as long as the database is expected to have more than 5% erroneous values.

**Scenarios 7 - 10: Comparing DQ Techniques - The Impact of “Error Masking”.**

In this subsection, we compare the DQ techniques  $T_{B \neq A}$  and  $T_{L_A}$  in another four scenarios, by investigating the impact of the “error masking” (varying  $c_1$  and  $c_2$ ) on the effectiveness of these techniques. The comparison is carried out with respect to a relatively clean database, i.e.,  $p = 0.05$  (Scenarios 7 and 9) and a dirty database, i.e.,  $p = 0.30$  (Scenarios 8 and 10), and as well as with respect to a nearly perfect lookup table, i.e.,  $s = 0.01$  (Scenarios 7 and 8), and an imperfect lookup table, i.e.,  $s = 0.3$  (Scenarios 9 and 10).

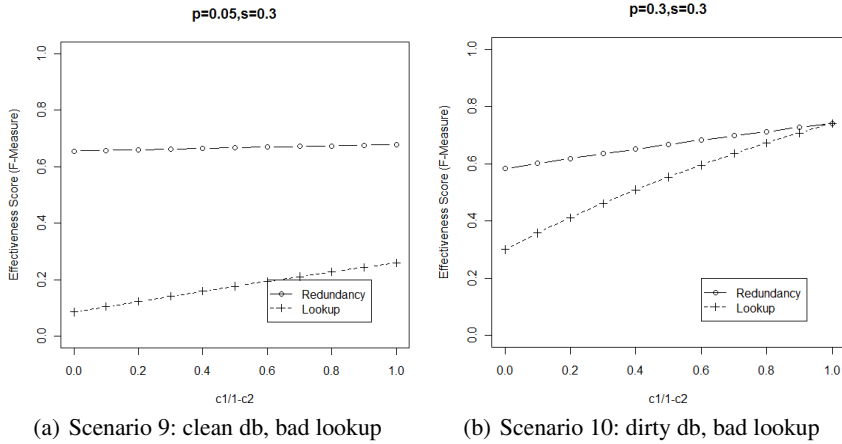
Figure 4(a) and 4(b) compare the F-Measures of  $T_{B \neq A}$  and  $T_{L_A}$  in Scenarios 7 and 8, while Figure 5(a) and 5(b) compare them in Scenarios 9 and 10. From these figures, a dominant pattern can be observed: the chance of “error masking” has more impact on  $T_{L_A}$  than on  $T_{B \neq A}$ , and this influence is independent of the probabilities of errors in  $A$  and the “coverage” of  $L_A$ . In other words, the F-measure of  $T_{L_A}$  increases more precipitously than that of  $T_{B \neq A}$  does (in all four cases) as the chance of “error masking” decreases (i.e., as  $c_1$  and  $1 - c_2$  increases).

In addition to this general pattern, the following conclusion can be reached according to these figures. For relative clean databases with less than 5% of erroneous values, a redundancy-check based DQ technique is always more effective than a lookup-table based technique. When there are more than 5% of erroneous values, the redundancy-check based technique still outperforms the lookup-table based one, unless a relatively low chance of “error masking” is guaranteed and the “coverage” of the lookup table is



(a) Scenario 7: clean db, good lookup (b) Scenario 8: dirty db, good lookup

**Fig. 4.** Comparison of  $T_{B \neq A}$  and  $T_{L_A}$  - Impact of “Error Masking” (I)



**Fig. 5.** Comparison of  $T_{B \neq A}$  and  $T_{L_A}$  - Impact of “Error Masking” (II)

nearly perfect. This conclusion, together with the one we made in Scenarios 5 and 6, gives us a complete comparison of  $T_{B \neq A}$  and  $T_{L_A}$ .

The general point is that the mathematical assessment of the effectiveness of DQ techniques based on probabilistic parameters allows us to make judgments about when to use one technique vs. another, or whether to use one at all – we need to remember that there is an overhead for putting into place a DQ technique.

## 5 Related Work

Software engineering researchers and practitioners have been developing and using numerous metrics for assessing and improving quality of software and its development processes [8]. In comparison, measures for DQ and DQ techniques have received less attention. Nevertheless, significant amount of effort has been dedicated to the classification and definition of DQ dimensions. Each such dimension aims at capturing and representing a specific aspect of quality in data, and can be associated with one or more measures according to different factors involved in the measurement process [1]. Measures for accuracy, completeness, timeliness dimensions have been proposed in [9,10,11].

Although to the best of our knowledge no general measurement framework exists for DQ techniques, performance measures have been proposed and used for certain types of techniques (such as Record linkage). Performance measures in this case are often defined as the functions of the number of true positives, false positives, etc [12,13]. For example, in addition to precision, recall and F-measures, the performance of a record linkage algorithm can also be measured using  $Accuracy = (TP + TN) / (TP + FP + TN + FN)$ , among others [13]. In these proposals, performance scores are obtained for particular applications of a record linkage algorithm on actual data sets, and are mainly used as a mechanism to tune the parameters (e.g., matching threshold) of the algorithm. The present paper focuses on the estimation of expected effectiveness scores and the comparison of DQ techniques under different scenarios. The formal techniques and what-if

analysis presented in this paper are therefore complementary to the existing performance measures used for record linkage algorithms.

As we have discussed, the schema of a database plays a significant role in ensuring quality of data in the database. Researchers in conceptual modeling have worked on the understanding and characterization of quality aspects of schemas, such as (schema) completeness, minimality, pertinence [14,1]. Moreover, quality measures have been proposed for ER schemas; for example, the integrity measure in [15] is defined using the number of incorrect integrity constraints and the number of correct ones that are not enforced. Quality measures for logical schemas have also been developed in [16,17,18,19]. The question that remains is — if quality of schema influences that of data, how is this influence reflected in their quality measures. The present paper can be seen as one step toward answering this question. Since DQ techniques rely on changes to the structure and elements of schemas (and the specifiable constraints according the changes), their effectiveness measures contribute to the measurement of schemas' *controllability* on DQ problems, another quality measure for schemas yet to be explored. Our effectiveness measures therefore help us to understand the relationship between the ability to control DQ problems at the schema level and the actual manifestation of these problems at instance level.

## 6 Conclusion

In this paper, we have proposed a quantitative approach for measuring the effectiveness of DQ techniques. Inspired by Information Retrieval, we started by proposing to calculate *numeric effectiveness scores* for a DQ technique by comparing its performance on a database instance with that of humans, who are assumed to have perfect knowledge of the world represented by that instance. As in Information Retrieval, this has the weakness of depending on the particular database instance used, and may require significant human effort in evaluating the actual data.

We therefore generalized the idea by introducing probabilistic assumptions concerning the occurrence of errors in data values and confounding factors that may render the DQ technique less effective. These assumptions are expressed in terms of probability distributions for various events, each characterized by certain parameters. We then showed with several examples how one can obtain *mathematical formulas* for the effectiveness of a DQ technique, which involve the parameters of the above-mentioned distributions. This is a significant advance, because it provides a way for the effectiveness of a DQ technique to be evaluated over a range of possible values for the parameters. This allows us for the first time to compare in a mathematically precise way different DQ techniques, and talk about the circumstances when one becomes better than another. Moreover, it lays the foundations for future research on optimal allocation of resources for DQ enforcement.

Ongoing and future work is needed to fulfill the promise of this approach. This includes to identify classes of DQ techniques (integrity constraints and workflows) for which the formal approach for deriving effectiveness formulas, as illustrated in this paper, can be mechanized or at least reduced to a systematic methodology. This will likely include a comprehensive classification of DQ techniques, and will result in a library of DQ techniques augmented by their effectiveness formulas. The next stage is to also make

the process of generating interesting scenarios more systematic. Finally, while this paper concentrated only on error detection, there are many other aspects, such as error correction and monitoring, that can be brought into this more precise, mathematical approach.

## References

1. Batini, C., Scannapieco, M.: *Data Quality: Concepts, Methodologies and Techniques (Data-Centric Systems and Applications)*, 1st edn. Springer, Heidelberg (2006)
2. Jiang, L., Topaloglou, T., Borgida, A., Mylopoulos, J.: Goal-oriented conceptual database design. In: *Proceedings of the 15th IEEE International Requirements Engineering Conference (RE 2007)* (2007)
3. Jiang, L., Borgida, A., Topaloglou, T., Mylopoulos, J.: Data quality by design: A goal-oriented approach. In: *Proceedings of the 12th International Conference on Info. Quality (ICIQ 2007)* (2007)
4. Bohannon, P., Fan, W., Geerts, F., Jia, X., Kementsietsidis, A.: Conditional functional dependencies for data cleaning. In: *IEEE 23rd International Conference on Data Engineering, 2007. ICDE 2007*, pp. 746–755 (2007)
5. Fan, W., Geerts, F., Jia, X., Kementsietsidis, A.: Conditional functional dependencies for capturing data inconsistencies. *ACM Trans. Database Syst.* 33(2), 1–48 (2008)
6. van Rijsbergen, C.: *Information Retrieval*, 2nd edn. Butterworth, London (1979)
7. Barbará, D., Goel, R., Jajodia, S.: Using checksums to detect data corruption. In: *Advances in Database Technology — EDBT 2000*, pp. 136–149 (2000)
8. Fenton, N.E., Pfleeger, S.L.: *Software Metrics: A Rigorous and Practical Approach*. PWS Publishing Co., Boston (1998)
9. Ballou, D., Wang, R., Pazer, H., Tayi, G.K.: Modeling information manufacturing systems to determine information product quality. *Manage. Sci.* 44(4), 462–484 (1998)
10. Pipino, L.L., Lee, Y.W., Wang, R.Y.: Data quality assessment. *Communications of the ACM* 45(4), 211–218 (2002)
11. Ballou, D.P., Pazer, H.L.: Modeling completeness versus consistency tradeoffs in information decision contexts. *IEEE Trans. on Knowl. and Data Engineering* 15(1), 240–243 (2003)
12. Gu, L., Baxter, R., Vickers, D., Rainsford, C.: Record linkage: Current practice and future directions. Technical report, CSIRO Mathematical and Information Sciences (2003)
13. Christen, P., Goiser, K.: Quality and complexity measures for data linkage and deduplication. In: Guillet, F., Hamilton, H.J. (eds.) *Quality Measures in Data Mining. Studies in Computational Intelligence*, vol. 43, pp. 127–151. Springer, Heidelberg (2007)
14. Batini, C., Ceri, S., Navathe, S.B.: *Conceptual Database Design: An Entity-Relationship Approach*. Benjamin/Cummings (1992)
15. Moody, D.L.: Metrics for evaluating the quality of entity relationship models. In: Ling, T.-W., Ram, S., Li Lee, M. (eds.) *ER 1998. LNCS*, vol. 1507, pp. 211–225. Springer, Heidelberg (1998)
16. Piattini, M., Calero, C., Genero, M.: Table oriented metrics for relational databases. *Software Quality Journal* 9(2), 79–97 (2001)
17. Calero, C., Piattini, M.: Metrics for databases: a way to assure the quality. In: Piattini, M.G., Calero, C., Genero, M. (eds.) *Information and database quality*, pp. 57–84. Kluwer Academic Publishers, Norwell (2002)
18. Baroni, A.L., Calero, C., Abreu, F.B., Piattini, M.: Object-relational database metrics formalization. In: *Sixth International Conference on Quality Software*, pp. 30–37. IEEE Computer Society, Los Alamitos (2006)
19. Serrano, M.A., Calero, C., Piattini, M.: Metrics for data warehouse quality. In: Khosrow-Pour, M. (ed.) *Encyclopedia of Info. Sci. and Techno. (IV)*, pp. 1938–1944. Idea Group (2005)

# Improving Model Quality Using Diagram Coverage Criteria

Rick Salay and John Mylopoulos

Department of Computer Science, University of Toronto  
Toronto, ON M5S 3G4, Canada  
{rsalay, jm}@cs.toronto.edu

**Abstract.** Every model has a purpose and the quality of a model ultimately measures its fitness relative to this purpose. In practice, models are created in a piecemeal fashion through the construction of many diagrams that structure a model into parts that together offer a coherent presentation of the content of the model. Each diagram also has a purpose – its role in the presentation of the model - and this determines what part of the model the diagram is intended to present. In this paper, we investigate what is involved in formally characterizing this intended content of diagrams as *coverage criteria* and show how doing this helps to improve model quality and support automation in the modeling process. We illustrate the approach and its benefits with a case study from the telecommunications industry.

**Keywords:** Modeling, Model quality, Diagrams.

## 1 Motivation

All models are created for a purpose [4]. For example, in a software development context, models may be used to communicate a software design, to help a designer work through alternative ideas, to support the work of various stakeholders, to enable a particular type of analysis, etc. The quality of a model corresponds to how well it can support its purpose by providing the information required by it – i.e. the purpose of a model determines its intended content. Thus, if the intended content can be characterized in terms of *content criteria* such as the required notation, coverage, accuracy, level of abstraction, etc. we can consider model quality to be measured by the degree to which the model meets these criteria.

In practice, a model is often manifested as a set of diagrams, possibly of different types, that decompose and structure the content of the model. The prototypical example of this is the UML which defines a single metamodel for UML models and identifies thirteen types of diagrams that can be used with it [13]. Like the models they present, each diagram has a purpose and plays a particular role in the presentation of model content, hence they too have content criteria. In particular, the content criteria relating to coverage, or *coverage criteria*, for a diagram identifies the part of the information carried by a model that is intended to be presented within the diagram. For example, in a UML model of a communication system, one class diagram may be intended to show the different types of communicating entities while another is intended to show the different types of messages that an entity of type *Terminal* can send.

Coverage criteria are not typically modeled and if they are made explicit at all, it is only through some informal means such as comments or as part of the name of the diagram. However, the explicit and precise expression of coverage criteria is a fruitful activity because it helps improve the quality of models in several ways. Firstly, it improves model comprehension because it provides information that allows model consumers to properly interpret the content of diagrams and assess their overall quality. For example, without explicit coverage criteria it may not be clear whether the associations in the class diagram of communicating entity types represent all or just some of the associations between these entities. Secondly, it can be used to identify types of defects that are not detectable through other means. For example, coverage criteria can be used to detect when the class diagram of communicating entity types contains classes it shouldn't or doesn't contain classes that it should. Finally, the coverage criteria can be used with change propagation mechanisms to properly maintain the intended content of diagrams as the model evolves.

In [11] we describe how the types of relationships that exist between models and between diagrams play a role in describing the intentions about content. In this paper, we explore the relationship between a diagram and a model in greater depth. In particular, we make the following contributions:

- The notion of diagram coverage criteria is introduced as a new kind of information that can be included in a model.
- Four kinds of modeling defects are identified that can only be detected using coverage criteria.
- A systematic approach for defining formal coverage criteria is presented and the validity conditions that coverage criteria must satisfy are specified.
- A strategy for parameterizing coverage criteria is defined that allows reuse of coverage criteria to reduce specification effort and to allow diagrams with standard types of intentions to be auto-generated.
- Empirical results are presented of the application of the approach to a medium size UML model with 42 diagrams.

The rest of the paper is structured as follows. Section 2 introduces the concepts related to diagram coverage criteria and illustrates them using examples. Section 3 formalizes these concepts and provides a systematic way of defining coverage criteria. Section 4 describes the results of applying the approach to a UML case study in the telecommunications domain. Finally in Section 5 we discuss related work and in Section 6 make some concluding remarks.

## 2 Diagram Coverage Criteria

In order to illustrate the idea of diagram coverage criteria we utilize examples from a UML case study taken from a standards document for the European Telecommunications Standards Institute (ETSI) [7]. The case study consists of three UML models: a context model (4 diagrams), a requirements model (6 diagrams) and a specification model (32 diagrams) and details the development of the Private User Mobility dynamic Registration service (PUMR) – a standard for integrating telecommunications networks in order to support mobile communications. Thus, for example, PUMR allows



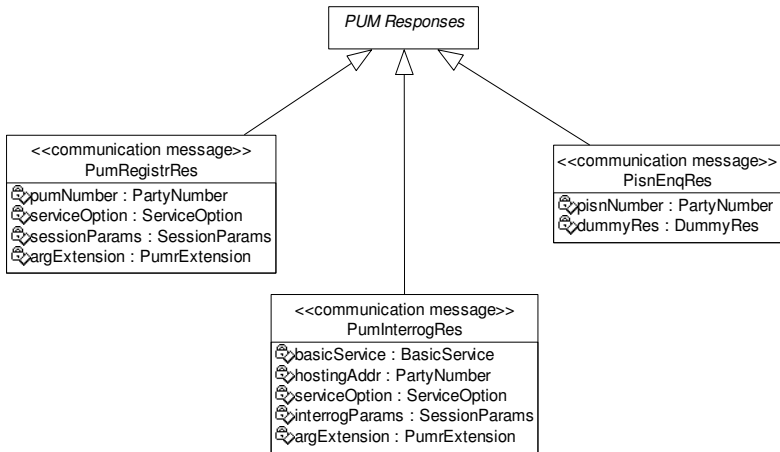
an employee using a mobile phone at her home company with a private exchange to roam to other private exchanges seamlessly. More specifically, it describes the interactions between Private Integrated Network eXchanges (PINX) within a Private Integrated Services Network (PISN). The following is a description from the document:

“Private User Mobility Registration (PUMR) is a supplementary service that enables a Private User Mobility (PUM) user to register at, or de-register from, any wired or wireless terminal within the PISN. The ability to register enables the PUM user to maintain the provided services (including the ability to make and receive calls) at different access points.” [7, pg. 43]

Consider diagram 65 from the specification model as shown in Figure 1. The intent of this class diagram is to show the detail for the types of response messages that can be exchanged between communication entities when they are trying to connect. The class *PUM Responses* is the abstract base class for these classes. This intention implies that the following constraints must hold between diagram 65 and the specification model:

1. Every class that is included in this diagram must either be *PUM Responses* or be a direct subclass of it.
2. Every direct subclass of *PUM Responses* in the specification model is included in this diagram.
3. For each class included in this diagram, every attribute in the specification model is included in this diagram.
4. No associations are included in the diagram.

These constraints constitute the coverage criteria for diagram 65. Assume that we identify three roles for stakeholders dealing with diagrams: definer, producer and consumer. The diagram definer asserts that such a diagram must exist and what it is intended to contain. The producer creates the content and the consumer uses it for their purposes. Expressing the coverage criteria explicitly and precisely is useful for all three roles. The definer can articulate the intent of the diagram and effectively communicate this to the producer. The producer can use this to assess whether they are conforming to this intent. Constraints (1) and (4) ensure that nothing is included that does not belong in the diagram while constraints (2) and (3) ensure that everything that does belong is included. The consumer can use the constraints to properly interpret the content of the diagram. For example, without (3) it may not be clear to the consumer whether or not the attributes for these classes shown in the diagram are all the attributes for these classes or there are some more that have been omitted from the diagram. Thus, while diagrams are typically assumed to be incomplete relative to the model, the coverage criteria provide the consumer with information about the ways in which the diagram *is* complete. If formalized, the coverage criteria are useful for automated support of the management of the diagram content in order to ensure that the intent of the diagram is maintained as the specification model evolves. For example, if the producer adds a class to the model via diagram 65 and does not make it a subclass of *PUM Responses*, this violates constraint (1) and can be flagged as such. On the other hand, if a subclass of *PUM Responses* is added to the specification model by some other means, such as manually through another diagram, change propagation, round-trip engineering, etc., the violation of constraint (2) can trigger the “repair” action of adding it to diagram 65.

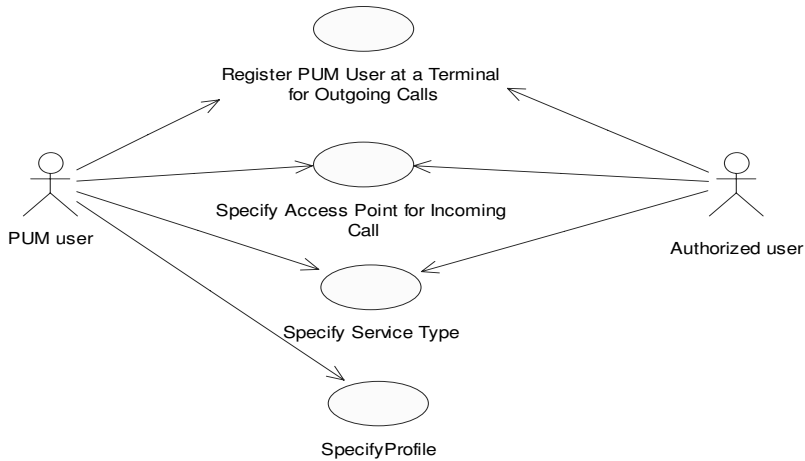


**Fig. 1.** Diagram 65 is a class diagram showing PUMR response message types carried in a connect signal

Note that the constraints in coverage criteria are different from constraints that are expressed within the metamodel. Metamodel constraints include invariants that must hold in the modeled domain (e.g. a class can't be a subclass of itself), completeness constraints from the modeling process (e.g. every use case requires an activity that describes it) and stylistic constraints (e.g. only single inheritance is permitted). Since diagrams do not exist within the model, these constraints do not address the content of diagrams. In contrast, coverage criteria are wholly concerned with the relationship between diagrams and the model. Furthermore, since the diagram intent is defined relative to the model, the coverage criteria is comprised of information that exists at the model level rather than at the metamodel level. For example, if model evolution causes some coverage criterion to be violated, a valid response may be to change the coverage criterion rather than the diagram. This represents a decision on the part of the model definer that the intent of a diagram has evolved.

The coverage criteria for diagram 65 are constraints that are mostly expressed in terms of the generic concepts in the metamodel (i.e. class, subclass, association, etc.) except for the mention of the specific class *PUM Responses*. The diagram has a special existential relationship to this element since it doesn't make sense to have a diagram that shows the subclasses of *PUM Responses* unless there exists a class in the model called *PUM Responses*. Thus, we consider it to be a *precondition* for the existence of diagram 65 that the model contain this class. When the precondition for a diagram is the required existence of a single model element, then the diagram is often a "detailing" of this element. For example, diagram 65 could be considered to be a detail view of the *PUM Responses* class. This special case has been leveraged by modeling tools [5, 6] to define a natural navigation path between diagrams containing the detailed element and the diagrams that are detail views of this element.

The coverage criteria for diagram 65 are strong enough that for any UML model that satisfies the precondition, the content of the diagram is uniquely determined – i.e. the coverage criteria constitutes a query on those specification models that satisfy the



**Fig. 2.** Diagram 44 is a use case diagram showing the registration use cases

precondition. The intuition here is that when the producer fills in a diagram they must always have some principle in mind by which they decide what should be included and what should not be included and following this principle results in a unique diagram<sup>1</sup>. This principle is exactly the coverage criteria. Thus, there is a general pattern for coverage criteria: there is a (possibly empty) precondition and the coverage criteria uniquely determine the content of the diagram on any model satisfying the precondition.

Now consider diagram 44 from the requirements model shown in Figure 2. The intent of this diagram is to show all use cases related to the registration of PUM users within a network. The coverage criteria can be expressed as follows:

1. Every use case that is included this diagram is a registration use case.
2. Every registration use case (in the requirements model) is included in this diagram.
3. Every actor (in the requirements model) associated with a use case included in the diagram is also included in the diagram.
4. Every association (in the requirements model) between any elements in the diagram is also included in the diagram.

Like the coverage criteria for diagram 65 this lists a set of diagram inclusion constraints that pick out a unique diagram for each model; however, unlike diagram 65, the truth of these conditions cannot be fully determined from the content of the requirements model alone. In particular, there is no information in the requirements model that can be used to determine whether or not a particular use case is a registration use case. This highlights another benefit of articulating the coverage criteria – it exposes contextual information that is assumed when interpreting the diagram and that may be missing from the model. One response to this is to extend the model

<sup>1</sup> Note that we are not suggesting that the information in a diagram can only be presented in one way but rather that what information is included in the diagram is determined completely by the principle the producer is following.

to include this information. In this case, this could be done in several ways ranging from formal to informal including: adding a use case called “Registration” which these use cases specialize, adding a profile at the metamodel level with an attribute to indicate the type of use case, using a naming convention on use cases to indicate the type of use case, annotating the use cases with comments, etc.

Another response to this situation is to treat the inclusion of a use case in the diagram as *meaning* that the use case is a registration use case. In this case, diagrams are not only used as views on the model but also to extend the model itself. Since diagrams are typically considered to only be relevant to the presentation of a model and not its content, this approach has the drawback that the information may not be preserved in further refinements of the model (e.g. into the code) and hence would be lost. This suggests that the first response may be preferred if this information is needed in downstream processes – i.e missing context information should be viewed as a case of model incompleteness.

## 2.1 Parameterized Coverage Criteria

In many cases, it is possible to generalize the diagram intention and coverage criteria by replacing certain constants by parameters. For example, let *DirectSubclass*[*c*:class] represent the coverage criteria for a type of class diagram having the generalized intention that the diagram shows a class *c* and the detail of all of its direct subclasses:

1. Every class that is included in this diagram must either be *c* or be a direct subclass of it.
2. Every direct subclass of *c* in the specification model is included in this diagram.
3. For each class included in this diagram, every attribute in the specification model is included in this diagram.
4. No associations are included in the diagram.

Now the coverage criteria for diagram 65 in Figure 1 could be expressed as *DirectSubclass*[*PUM Response*]. An obvious benefit of parameterized coverage criteria is reuse as it reduces the incremental effort to define the coverage criteria for different diagrams when the coverage criteria have the same form. However, there are other benefits as well. Formalized parameterized coverage criteria can be used to define a library of common types of diagrams that can then be used to automatically *generate* diagrams of these types and hence reduce modeling effort. For example, a model that is produced by a reverse engineering tool can be quickly structured by generating diagrams using different parameterized coverage criteria and various parameters. Furthermore, the generalized intent can be used to generate a meaningful diagram name (and other metadata) that reflects the intent of the diagram. For example, *DirectSubclass*[*c*:class] can be applied to various classes to produce the diagram of its subclasses and generate the name “The direct subclasses of *c*” for each diagram.

## 3 Formalization

The objective of this section is to encode coverage criteria formally in order to be precise about its structure, allow the definition of validity conditions that must hold

and for characterizing the types of defects that can be detected. In order to express metamodels in a formal way, we have chosen to use order-sorted first order logic with transitive closure (henceforth referred to as FO+) as the metamodeling formalism rather than using either MOF or Ecore with OCL. There are a number of reasons for this. Firstly, first order logic is widely known and has comparable expressive power to the other metamodeling approaches. Secondly, it has a textual representation that is more convenient when discussing formal issues. Finally, its semantics are formal and notions such as logical consequence and consistency are well defined.

Using FO+ we can define the metamodel of a model type as a pair  $\langle \Sigma, \Phi \rangle$  where  $\Sigma$  is called the *signature* and defines the types of model elements and how they can be related while  $\Phi$  is a set of axioms that defines the well-formedness constraints for models. Thus, a metamodel  $\langle \Sigma, \Phi \rangle$  is an FO+ theory and each finite model (in the model theoretic sense) of this theory will be considered to be a model (in the modeling sense) that is an instance of this metamodel.

For example, we define the metamodel (abstract syntax) of a simplified UML class diagram as follows.

$$\begin{aligned}
 \text{CD} = & \hspace{20em} (1) \\
 & \mathbf{sorts} \text{ class, assoc, attr, string} \\
 & \mathbf{pred} \text{ startClass: assoc} \times \text{class} \\
 & \quad \text{endClass: assoc} \times \text{class} \\
 & \quad \text{attrClass: attr} \times \text{class} \\
 & \quad \text{className: class} \times \text{string} \\
 & \quad \text{subClass: class} \times \text{class} \\
 \\ 
 & \mathbf{constraints} \\
 & \quad // \text{startClass, endClass, attrClass and className are functions}^2 \\
 & \quad \forall a:\text{assoc} \exists !c:\text{class} \cdot \text{startClass}(a, c) \\
 & \quad \forall a:\text{assoc} \exists !c:\text{class} \cdot \text{endClass}(a, c) \\
 & \quad \forall a:\text{attr} \exists !c:\text{class} \cdot \text{attrClass}(a, c) \\
 & \quad \forall c:\text{class} \exists !s:\text{string} \cdot \text{className}(c, s) \\
 & \quad // \text{a class cannot be a subclass of itself} \\
 & \quad \forall c:\text{class} \cdot \neg \text{TC}(\text{subClass}(c, c))
 \end{aligned}$$

The signature  $\Sigma_{\text{CD}}$  consists of the pair  $\langle \text{sorts}_{\text{CD}}, \text{pred}_{\text{CD}} \rangle$  where  $\text{sorts}_{\text{CD}}$  is the set of element types that can occur in a model while  $\text{pred}_{\text{CD}}$  is a sets of predicates used to connect the elements. We will say  $\Sigma_{T_1} \subseteq \Sigma_T$  to mean that  $\text{sorts}_{T_1} \subseteq \text{sorts}_T$  and  $\text{pred}_{T_1} \subseteq \text{pred}_T$ . The **constraints** section describes  $\Phi_{\text{CD}}$ . Note that the quantifier  $\exists!$  means “there exists one and only one” and the operator TC takes a predicate and produces its transitive closure.

---

<sup>2</sup> In FO+ we express functions as appropriately constrained predicates rather than including functions directly into the logic in order to treat this in a uniform way with other well-formedness constraints.

Here we are treating a type of diagram (i.e. class diagrams) in the same way as a type of model by giving it a metamodel defining its abstract syntax. We do this since in this paper we are not interested in the notational aspects of a diagram, only in what subsets of a model they can be used to present. Thus, we will take a diagram to be equivalent to the *submodel* it picks out from a model.

Assume that  $T = \langle \Sigma_T, \Phi_T \rangle$  is the metamodel of some model type and  $T1 = \langle \Sigma_{T1}, \Phi_{T1} \rangle$  is the metamodel for a type of submodel of  $T$ . For example, we could have  $UML = \langle \Sigma_{UML}, \Phi_{UML} \rangle$  as the model type and  $CD = \langle \Sigma_{CD}, \Phi_{CD} \rangle$  as the type of submodel we are interested in. Since  $T1$  is a type of submodel of  $T$  we will assume that  $\Sigma_{T1} \subseteq \Sigma_T$ . Note that in general, the constraints of a type of submodel may be either stronger or weaker than the constraints of the model. For example, in UML, communication diagrams can only represent interactions in which message overtaking does not take place [13] and so the constraints on communication diagrams are stronger than on interactions within a UML model.

Now assume that  $M:T$  is a model and  $Msub:T1$  is intended to be a submodel of it. We will interpret this to mean that the constraint  $Msub \subseteq M$  must hold. That is, each element and relation instance in  $Msub$  must also be found in  $M$ . If in addition, we state that  $Msub$  has coverage criteria  $CC(Msub, M)$ , then intuitively this will mean that  $CC$  contains the preconditions and the constraints that further limit which submodels  $Msub$  can be. Formally, we can express  $CC$  as a set of constraints on the combined signatures of  $T1$  and  $T$  using a special type of metamodel that includes the metamodels of  $T1$  and  $T$  and additional constraints showing how these models are related [11]. As an example, we will express the coverage criteria for diagram 65 as follows:

$$CC_{M65}(M65:CD, M:UML) = M65.CD + M.UML + \quad (2)$$

$$\mathbf{sort} M65.class \leq M.class, M65.assoc \leq M.assoc, M65.attr \leq M.attr \quad (3)$$

### constraints

// precondition

$$\exists mc:M.class \cdot M.className(c) = \text{“PUM Responses”} \wedge \quad (4)$$

// inclusion constraints

$$\forall c:M.class \cdot (\exists c1:M65.class \cdot c1 = c) \Leftrightarrow (c = mc \vee M.subClass(c, mc)) \wedge \quad (5)$$

$$\forall a:M.assoc \cdot (\exists a1:M65.assoc \cdot a1 = a) \Leftrightarrow FALSE \wedge \quad (6)$$

$$\forall a:M.attr \cdot (\exists a1:M65.attr \cdot a1 = a) \Leftrightarrow (\exists c:M65.class \cdot M.attrClass(a) = c) \wedge \quad (7)$$

$$\forall c1, c2:M65.class \cdot M65.subClass(c1, c2) \Leftrightarrow (M.subClass(c1, c2) \wedge c2 = mc) \wedge \quad (8)$$

$$\forall a:M65.attr, c:M65.class \cdot M65.attrClass(a, c) \Leftrightarrow M.attrClass(a, c) \wedge \quad (9)$$

$$\forall c:M65.class, s:M65.string \cdot M65.className(c, s) \Leftrightarrow M.className(c, s) \quad (10)$$

Line (2) indicates that  $CC_{M65}$  imports the signature and constraints for  $CD$  and  $UML$  and to avoid name clashes these are “namespaced” with “ $M65$ ” and “ $M$ ”, respectively. Thus, for example, the sort  $M65.class$  is distinct from the sort  $M.class$ . Line (3) asserts that the elements in  $M65$  are subsets of the elements in  $M$ . The precondition (4) asserts the constraints that must hold in  $M$  for the diagram to exist – in this case that  $M$  must contain a class named “PUM Responses”. The use of a

precondition distinguishes the case of a diagram not existing from the case that the diagram exists but is empty.

The inclusion constraints define the constraints that must hold between the content of M65 and M and are defined in the scope of the precondition so that the variable *mc* is bound. These encode the constraints for diagram 65 expressed in words in section 2. Thus, constraints 1 and 2 are expressed by (5), constraint 3 is expressed by (7) and (9) and constraint 4 is expressed by (6). Note that parameterized coverage criteria can be defined by allowing free variables in the definition. For example, if we allow *mc* to remain a free variable in the above then we define the parameterized coverage criteria *DirectSubclass*(M65:CD, M:UML)[*mc*: class].

The coverage criteria above are written in a standardized form. If we assume that we are expressing coverage criteria CC(Msub:T1, M:T) then the form is:

$$\begin{aligned}
 \text{CC}(\text{Msub:T1}, \text{M:T2}) &= \text{Msub.T1} + \text{M.T2} + & (11) \\
 \text{subsort} &\text{ for each } S \in \text{sorts}_{T1}, \text{Msub.S} \leq \text{M.S} \\
 \text{constraints} & \\
 // \text{precondition} & \\
 \text{precondition} &\wedge \\
 // \text{inclusion constraints} & \\
 \text{for each } S \in \text{sorts}_{T1}, & \\
 \forall x:\text{M.S} \cdot (\exists xI:\text{Msub.S} \cdot xI = x) &\Leftrightarrow Q_S(x) \wedge \\
 \text{for each predicate } P:S_1 \times \dots \times S_n \in \text{pred}_{T1} & \\
 \forall x_1:\text{Msub.S}_1, \dots, x_n:\text{Msub.S}_n \cdot & \\
 \text{Msub.P}(x_1, \dots, x_n) &\Leftrightarrow \text{M.P}(x_1, \dots, x_n) \wedge Q_P(x_1, \dots, x_n) \wedge
 \end{aligned}$$

In each inclusion constraint,  $Q_i$  represents a formula called the *inclusion condition* that may involve bound variables in the precondition. Intuitively, the inclusion conditions pick out the parts of M that belong in Msub and provide a systematic way of defining coverage criteria. Based on this form, the coverage criteria can be seen to consist more simply of the precondition and a set of inclusion conditions. For example, the coverage criteria for diagram 65 could be expressed more compactly as the set of definitions:

$$\begin{aligned}
 \text{precondition} &:= \exists mc:\text{M.class} \cdot \text{M.className}(c) = \text{"PUM Responses"} & (12) \\
 Q_{\text{class}}(c) &:= (c = mc \vee \text{M.subClass}(c, mc)) \\
 Q_{\text{assoc}}(a) &:= \text{FALSE} \\
 Q_{\text{attr}}(a) &:= \exists c:\text{M65.class} \cdot \text{M.attrClass}(a, c) \\
 Q_{\text{subClass}}(c1, c2) &:= (c2 = mc) \\
 Q_{\text{attrClass}}(a, c) &:= \text{TRUE} \\
 Q_{\text{className}}(c, s) &:= \text{FALSE} \\
 Q_{\text{startClass}}(a, c)^3 &:= \text{FALSE} \\
 Q_{\text{endClass}}(a, c) &:= \text{FALSE}
 \end{aligned}$$

<sup>3</sup> Inclusion constraints for startClass and endClass were omitted in (11) since these must always be empty relations because there are no associations.

When the coverage criteria is expressed in terms of inclusion conditions it is clear that for every  $M$  that satisfies the precondition, the inclusion constraints specify a unique submodel  $M_{sub}$  of  $M$ . This is because there is a constraint for each sort and predicate of  $M_{sub}$  that determines exactly what subset of these from  $M$  are included in  $M_{sub}$ . To ensure that the resulting submodel  $M_{sub}$  is also always well formed – i.e. that it satisfies the constraints  $\Phi_{T1}$  – we must add the following validity conditions:

$$M.\Phi_T \cup M_{sub}.\Phi_{T1} \cup \Phi_{CC} \neq FALSE \quad (13)$$

$$M.\Phi_T \cup \Phi_{CC} \models M_{sub}.\Phi_{T1} \quad (14)$$

Here,  $M.\Phi_T$  and  $M_{sub}.\Phi_{T1}$  are the imported versions of the constraints of  $T$  and  $T1$  found in  $CC(M_{sub}:T1, M:T2)$  and  $\Phi_{CC}$  are the set of subsort, precondition and inclusion constraints. Condition (13) says that the constraints in  $CC$  must be consistent and condition (14) guarantees that the submodel defined by the coverage criteria for each  $T$ -model is a well formed  $T1$ -submodel. As a simple example of a case where candidate coverage criteria does not satisfy condition (14), consider that in (12) we can change the definition of the inclusion condition  $Q_{attr}(a)$  to be  $Q_{attr}(a) := TRUE$ . This now says that  $M65$  contains all attribute elements of  $M$  but still only a subset of the class elements. This clearly can result in  $M65$  being an ill-formed class diagram since it can now contain attributes with no corresponding class – i.e.  $attrClass$  is no longer necessarily a function. In particular, this is due to the fact that for the modified constraints  $\Phi_{CC_{M65}}$  we have that:

$$\Phi_{UML} \cup \Phi_{CC_{M65}} \neq \forall a:M65.attr \exists !c:M65.class \cdot attrClass(a, c) \quad (15)$$

We can directly relate the structure of coverage criteria to the types of defects that coverage criteria can be used to detect as shown in Table 1. The first two types of defects can occur when an instance of the submodel violates the coverage criteria – either by excluding intended information or by including unintended information. The third type of potential defect indicates that the coverage criteria cannot be fully characterized using information in  $M$ . This was the case with diagram 44 discussed in Section 2 since the inclusion condition identifying a registration use case could not be expressed. As discussed there, the corrective action required depends on whether the information represented by the inclusion conditions is considered to be needed by downstream processes using the model. The fourth type of potential defect is the case where the inclusion condition *can* be specified using information in the model but this information is only “weakly modeled” using some informal scheme such as naming conventions. For example, if a convention is used to prefix all registration use cases with the string “Reg\_”, this would allow the inclusion condition to be defined by checking for this prefix. The potential problem with this is that the semantics of these conventions may be lost in downstream uses of the model unless they are recorded with the model in some way. Thus it may be preferable to promote this information to “first class” status by modeling it directly – e.g. assuming registration use cases all specialize a use case called “Registration”.



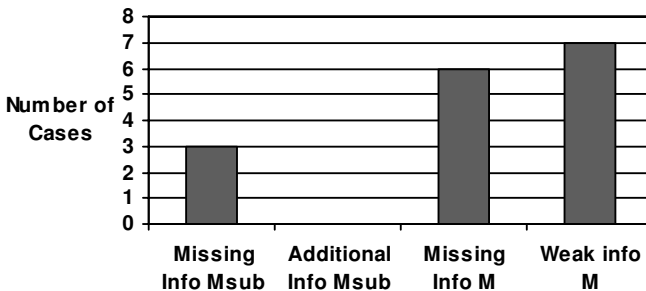
**Table 1.** Defect types

Defect Type	Description	Occurrence criteria
Missing information in Msub	Msub does not contain some information from M that it should.	An instance of an inclusion constraint in which right hand side is satisfied but the left hand side is not.
Too much information in Msub	Msub contains some information in M that it should not.	An instance of an inclusion constraint in which the left hand side is satisfied but the right hand side is not.
Missing information in M	The coverage criteria of Msub cannot be formally expressed in terms of the content of M.	One or more formulas $Q_i$ cannot be formally expressed using information in M.
Weakly modeled information in M	The coverage criteria of Msub is expressed in terms of content in M that is not modeled in the metamodel of M.	One or more formulas $Q_i$ are expressed using informal criteria such as naming conventions.

### 4 Application to the PUMR Example

In this section we discuss the results of our analysis to express the content criteria for the 42 diagrams over the three UML models in the PUMR example. Since we did not have access to the original definers of these diagrams, we relied on the documentation [7] of the diagrams to infer their intentions. Fortunately, the documentation is substantial and detailed so that we have a high level of confidence that our results are reasonable. To give a sense for the diversity of coverage criteria of the PUMR diagrams, we summarize a few in Table 2.

Figure 3 summarizes the defects found due to our analysis. The bars correspond to the types of defects described in Table 1. To some extent, the low number of missing/additional info defects found for Msub could be attributed to the fact that in



**Fig. 3.** Defects found in PUMR example

**Table 2.** Examples of coverage criteria from PUMR analysis

Diagram	Diagram Type	Summary of coverage criteria
62	Class Diagram	The communication classes in the QSIG package that are used by classes in the PUMR package.
73	Class Diagram	All error code classes in the PUMR package and their attributes.
52	Object Diagram	All the objects and links used in registration and de-registration interactions.
58	Statemachine Diagram	The content of statemachine “Registration Processing” except the content of composite state “Registration Request” (the content of “Registration Request” is shown in diagram 59).

**Table 3.** Parameterized coverage criteria used in the PUMR example

Parameterized coverage criterion	Diagram Type	Summary of coverage criteria	Inst.
<i>DirectSubSuper</i> [class]	Class Diagram	Immediate subclasses and superclasses of the class	2
<i>DirectSubclass</i> [class]	Class Diagram	Immediate subclasses of the class	4
<i>DirectAgg</i> [class]	Class Diagram	Classes directly aggregated by the class	3
<i>NameContains</i> [string]	Class Diagram	Classes whose name contains the string	6
<i>FullActivity</i> [activity]	Activity Diagram	The full contents of the activity	4
<i>FullSequence</i> [interaction]	Sequence Diagram	The full contents of the interaction	5
<i>FullState</i> [state]	State machine Diagram	The full contents of a composite state	1

expressing the coverage criteria we were trying to find the criteria that would best fit the existing diagrams. It is interesting that despite this, there were some clear errors that we were able to find. Cases that exhibited the third type of defect included diagram 44 shown in Figure 2 where there was no way to express the inclusion condition on registration use cases using information in the model. All of the examples of the fourth type of defect relied on naming conventions to identify a type of element. For example, diagram 70 shows the different enquiry message classes. These are all identifiable by having the stereotype “communication message” and by having a name with the prefix “PumEnq”.

Of the 42 diagrams, 25 could be seen to clearly be instances of more general parameterized coverage criteria types. This is elaborated in Table 3. Certain “large objects” (e.g. activities, interactions and statemachines) in a UML model have dedicated

diagram types. The most common coverage criteria is to show the full content of these objects in a diagram (e.g. *FullActivity*[activity], *FullSequence*[interaction]). The variety of coverage criteria that can be associated with these depend on their ability to show partial information. For example, statemachine diagrams can also be used to show the content of a single composite state and so we have *FullState*[state]. This capability can be used to decompose the presentation of a large statemachine across several diagrams. This is the case with diagrams 58 and 59 – together they depict the statemachine showing registration processing. A similar possibility exists with large interactions and activities but no instances of these occur in the PUMR example.

## 5 Related Work

Most of the work relating to diagrams deals with their role in providing a notation for a model. For example, in [1] Baar formalizes the relationship between the metamodels for the concrete syntax and the abstract syntax, in [2], Gurr defines conditions under which a notation is effective, etc. In contrast, our interest is in how diagrams delineate submodels and impose structure on a model and this bears a closer connection to the work on heterogeneous collections of related models.

The problem of inter-view consistency has been much studied, especially with UML (e.g. see [3]) and along with this, investigations into generic constraint management mechanisms such as change propagation and conformance detection have been pursued [10, 9]. Our focus is on the identification and elaboration of a new class of constraints that can characterize an aspect of modeling intention and can be of use in modeling. Thus, our concerns are somewhat orthogonal to but complementary with this work.

In another direction, generic configurable modeling environments have emerged such as MetaEdit+[6] and the Generic Modeling Environment (GME) [5]. The use of the diagram structure here is for the navigation from model elements in one diagram to other diagrams showing more detail about the element (e.g. its internal structure). Such a navigation approach is limited to expressing the intent of diagrams that detail model elements and cannot express more complex intentions such as the “dependency” class diagram containing all classes in package P1 used by classes in package P2. Furthermore, the types of detailing diagrams that can be expressed are restricted to those that can be defined by revealing/hiding particular element types defined in the metamodel. More complex coverage criteria such as only showing “direct” subclasses in diagram 65 are not possible.

Aspect oriented modeling (AOM) bears some similarity to our work and a wide variety of approaches for AOM have been proposed [12]. Here the idea is to provide a means for separately maintaining and developing *aspects* - subsets of the information in a model relating to particular concerns such as security or customization - and then allowing these to be woven together to produce the complete model when necessary. Since there is no consensus on what exactly an aspect is, it is difficult to clearly differentiate our work from this – is every diagram with an intent a valid aspect? The practice of AOM suggests otherwise. Aspects are typically associated with software concerns that crosscut the model whereas we have no such bias. The motivation of AOM is to provide techniques for separating concerns in a manageable way whereas

ours is to articulate the intent of diagrams in order to improve the quality of models. AOM emphasizes the independence of aspects whereas we focus on how submodels are related and are interdependent.

## 6 Conclusions and Future Work

All models and diagrams of a model have a purpose that circumscribes their contents through content criteria. Moreover, their quality can be assessed by how well their content meets these content criteria. In this paper, we focus on a particular type of content criteria for diagrams called coverage criteria. Coverage criteria for a diagram specify the part of the model that a given diagram is intended to contain. Although coverage criteria are not typically expressed explicitly, we propose that doing so can improve the quality of models by improving model comprehension by stakeholders, allowing the detection of defects that previously could not be detected and supporting automated change propagation and generation of diagrams. We have specified a systematic way of defining coverage criteria using preconditions and inclusion conditions and we have formally defined the conditions under which these conditions are valid. Finally, we have shown the results of applying these concepts to an actual UML case study consisting of 42 diagrams and the concrete benefits we obtained.

As part of future work, we are investigating how to extend this research to expressing coverage criteria about collections of diagrams. The motivating observation here is that the collection of diagrams presenting a model are typically structured further into related subgroups. For example, diagram 44 in Figure 2 can be grouped with another similar use case diagram (diagram 48) that shows the deregistration use cases and together, these two diagrams decompose the full set of use cases in the requirements model. If we take this subgroup of diagrams to be a kind of model (a *multi-model* [11]) then it can have its own coverage criteria that says that it consists of a set of use case diagrams that decompose the use cases in the requirements model. In this way, we hope to extend the ideas in this paper to characterize the intentions about the way collections of diagrams are structured.

**Acknowledgments.** We are grateful to Alex Borgida for his insightful comments on earlier drafts of this paper.

## References

1. Baar, T.: Correctly defined concrete syntax for visual models. In: Nierstrasz, O., Whittle, J., Harel, D., Reggio, G. (eds.) MoDELS 2006. LNCS, vol. 4199, pp. 111–125. Springer, Heidelberg (2006)
2. Gurr, C.: On the isomorphism, or lack of it, of representations. In: Marriott, K., Meyer, B. (eds.) Visual Language Theory, pp. 293–306. Springer, Heidelberg (1998)
3. Huzar, Z., Kuzniarz, L., Reggio, G., Sourrouille, J.L.: Consistency problems in uml-based software development. In: Jardim Nunes, N., Selic, B., Rodrigues da Silva, A., Toval Alvarez, A. (eds.) UML Satellite Activities 2004. LNCS, vol. 3297. pp. 1–12. Springer, Heidelberg (2005)

4. Ladkin, P.: Abstraction and modeling, research report RVS-Occ-97-04, University of Bielefeld (1997)
5. Ledeczi, A., Maroti, M., Bakay, A., Karsai, G., Garrett, J., Thomason IV, C., Nordstrom, G., Sprinkle, J., Volgyesi, P.: The Generic Modeling Environment. In: Workshop on Intelligent Signal Processing, May 17 (2001)
6. MetaEdit+ website, <http://www.metacase.com>
7. Methods for Testing and Specification (MTS); Methodological approach to the use of object-orientation in the standards making process. ETSI EG 201 872 V1.2.1 (2001-2008), [http://portal.etsi.org/mbs/Referenced%20Documents/eg\\_201\\_872.pdf](http://portal.etsi.org/mbs/Referenced%20Documents/eg_201_872.pdf)
8. MOF<sup>TM</sup> Query / Views / Transformations (QVT) – Final Spec., <http://www.omg.org/cgi-bin/doc?ptc/2005-11-01>
9. Nentwich, C., Capra, L., Emmerich, W., Finkelstein, A.: xlinkit: a consistency checking and smart link generation service. ACM TOIT 2(2), 151–185 (2002)
10. Nuseibeh, B., Kramer, J., Finkelstein, A.: A Framework for Expressing the Relationships Between Multiple Views in Requirements Specifications. IEEE TSE 20(10), 760–773 (1994)
11. Salay, R., Mylopoulos, J., Easterbrook, S.: Managing Models through Macromodeling. In: Proc. ASE 2008, pp. 447–450 (2008)
12. Schauerhuber, A., Schwinger, W., Retschitzegger, W., Wimmer, M.: A Survey on Aspect-Oriented Modeling Approaches (2006), <http://wit.tuwien.ac.at/people/schauerhuber>
13. UML 2.0 Metamodel, <http://www.omg.org/cgi-bin/apps/doc?ptc/04-10-05.zip>

# A Method for the Definition of Metrics over $i^*$ Models<sup>\*</sup>

Xavier Franch

Universitat Politècnica de Catalunya (UPC)  
c/Jordi Girona, 1-3, E-08034 Barcelona, Spain  
franch@lsi.upc.edu

**Abstract.** The  $i^*$  framework has been widely adopted by the information systems community for goal- and agent-oriented modeling and analysis. One of its potential benefits is the assessment of the properties of the modeled socio-technical system. In this respect, the definition and evaluation of metrics may play a fundamental role. We are interested in porting to the  $i^*$  framework metrics that have been already defined and validated in other domains. After some experimentation with  $i^*$  metrics in this context, the complexity inherent to their definition has driven us to build a method for defining them. In this paper, we present the resulting method,  $iMDF_M$ , which is structured into 4 steps: domain analysis, domain metrics analysis, metrics formulation and framework update. We apply our approach to an existing suite of metrics for measuring business processes performance and drive some observations from this experiment.

**Keywords:** goal-oriented models,  $i^*$ , metrics, business process performance.

## 1 Introduction

Goal-oriented modelling [1] is widely used in Information Systems (IS) development as a way to establish high-level goals and decompose them until obtaining measurable requirements. High-level goals capture the overall organizational objectives and key constraints; therefore they represent stable needs that are less sensitive to changes.

The  $i^*$  framework [2] 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, etc. As a modelling framework, one of its required applications is the ability to evaluate properties of the model that may help to detect some flaws in the modelled system, or to compare different alternatives with respect to some criteria.

As a result, some authors have explored techniques for driving the analysis of  $i^*$  models. Qualitative-predominant techniques were already formulated in [2] and later other techniques were proposed [3, 4]. Quantitative-predominant proposals aim at formulating metrics for measuring some criteria (see Section 2.1). Having good suites of metrics allows not only analysing the quality of an individual model, but also comparing different alternative models with respect to some properties in order to select the most appropriate alternative.

---

\* This work has been partially supported by the Spanish project TIN2007-64753.

Having this in mind, we proposed *i*MDF, an *i*\* Metrics Definition Framework [5] (see Section 2.2). The framework maintains a language of patterns in which metric templates are defined by means of OCL expressions expressed over an *i*\* metamodel. We are especially interested in the case in which the metrics are not defined from scratch, but they already exist in the domain that is being modelled with *i*\* (e.g., organizations, business processes, software architectures, etc.) and they have been properly validated. Therefore, the problem we face is not metric definition and validation, but mapping metrics from the starting domain to *i*\*.

As a result of experimentation with *i*MDF in the context described above, we have observed that the process for defining metrics may be quite complex, because it requires a full understanding of the domain that is being modelled using *i*\*, as well as the suite of metrics itself. Therefore, we have formulated a method, *i*MDF<sub>M</sub>, for driving the process of definition of metrics in *i*\*. The presentation of this method is the main goal of the paper.

To illustrate the method, we define a suite of *i*\* metrics for business process design and evaluation based on a proposal from Balasubramanian and Gupta [6] that in its turn consolidates others' proposals. The definition of this suite becomes a second goal of the paper, both for the interest of the result itself (i.e., a representative iteration in the incremental construction of a comprehensive catalogue of metrics in *i*\*), and for the feedback over the framework (for refining the language of patterns, acquiring some more lessons learned, etc.).

Basic knowledge of *i*\* is assumed in the paper, see [2] and the *i*\* wiki (<http://istar.rwth-aachen.de>) for a thorough presentation.

## 2 Background and Previous Work

### 2.1 Quantitative Analysis of *i*\* Models

In spite of the high dissemination of the *i*\* framework, only a few approaches have been proposed presenting some kind of metrics for measuring *i*\* models. We are mostly interested in quantitative-dominant proposals because they allow more objective and repeatable analysis of *i*\* models. Apart from *i*MDF itself, we mention Kaiya et al.'s AGORA method [7] that provides techniques for estimating the quality of requirements specifications with emphasis in the AND/OR decomposition of goals. Sutcliffe and Minocha [8] propose the analysis of dependency coupling for detecting excessive interaction among users and systems; they combine quantitative formulae based in the form of the model with some expert judgment for classifying dependencies into a qualitative scale. Bryl et al. propose structural metrics for measuring the Overall Plan Cost of agent-based systems [9].

### 2.2 *i*MDF: A Framework for *i*\* Metrics

The *i*MDF framework is the result of our work on *i*\* metrics over time:

- Phase 1. Preliminary work. Several metrics were defined ad-hoc for comparing alternative *i*\* models with respect to some properties [10]. This phase revealed the convenience of having some foundations for defining these metrics and provided the necessary expertise for formulating a framework with this goal.

- Phase 2. Formulation of the  $i$ MDF framework. From this experience and the study of other work done in the field, the  $i$ MDF framework was formulated embracing:
  - o A metamodel [11] including the most relevant concepts in  $i^*$ . It is conceived as extensible, since this is a crucial characteristic of the  $i^*$  framework.
  - o General forms of  $i^*$  metrics [12] and patterns for producing them [5].
  - o A preliminary (formative) validation based on experimentation over individual metrics coming from several sources (e.g., [13]). The result formed a first catalogue of  $i^*$  metrics in  $i$ MDF.
- Phase 3. Validation of the framework. As mentioned in the introduction, we are currently porting some existing measurement proposals over  $i$ MDF. Also, some work is planned for formulating and validating metrics about the structural quality of  $i^*$  models (complexity, cohesion, ...). As a result, we are enlarging the  $i$ MDF catalogue whilst learning some insights about limitations of this approach.

Precisely, one of the identified limitations, the lack of clear guidance to define the metrics, has motivated the present work.

### 3 $i$ MDF<sub>M</sub>: A Method for Defining Metrics over $i^*$ Models

In this section we describe the  $i$ MDF<sub>M</sub> method for developing metrics over  $i^*$  models in the  $i$ MDF framework. It consists of 4 steps (see Fig. 1) described below.

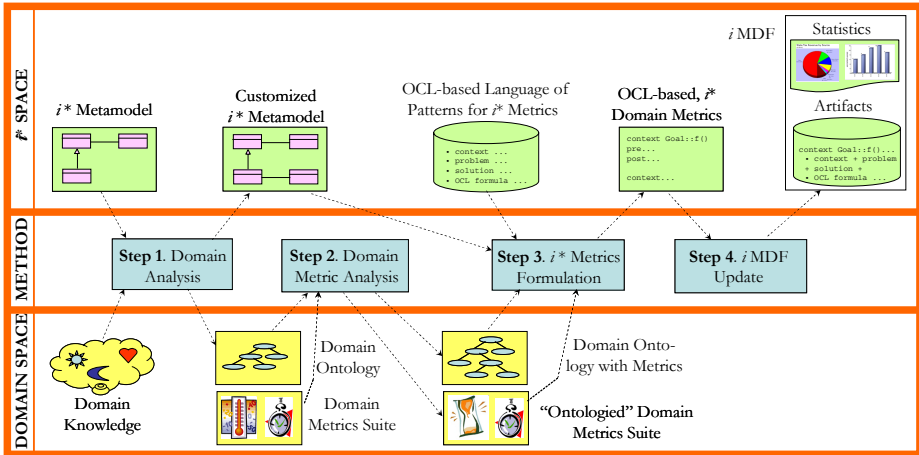


Fig. 1. The  $i$ MDF<sub>M</sub> method: steps and artifacts

#### 3.1 Step 1: Domain Analysis

The goal of this step is to gain understanding about the domain whilst establishing the mapping from concepts in that domain onto the  $i^*$  framework. Therefore, Domain Analysis comprises two different activities:



- *Activity 1.1: Create a Domain Ontology.* From the knowledge about the domain (in the form of domain model semantics, related ontologies, tacit knowledge, etc.), an ontology is created or eventually, reused.
- *Activity 1.2: Map the Domain Ontology onto the  $i^*$  Metamodel.* The correspondence between domain concepts and  $i^*$  constructs is established here. Concepts like e.g. business process, stakeholder, software component, etc., are therefore mapped onto  $i^*$  constructs like goal, task, agent, etc.
  - *Activity 1.2.1: Customize the  $i^*$  metamodel.* The  $i^*$  metamodel as defined in [11] is refined into a specialization for the domain. This refinement step may involve adding some new attributes or even classes, or more usually integrity constraints that impose restrictions on the way  $i^*$  constructs are used to support the domain ontology.

### 3.2 Step 2: Domain Metrics Analysis

This step aims at analysing the departing suite of domain metrics before its formalization is tackled. The analysis moves along two directions, exogenous (seeking an accurate correspondence with the domain ontology), and endogenous (making the metrics uniform and complete). The activities performed are thus:

- *Activity 2.1: Extend the Domain Ontology.* The domain ontology is extended to incorporate those concepts that did not appear in the former analysis of the domain and that become necessary for using the metric suite.
- *Activity 2.2: Consolidate the Domain Metrics Suite.* The suite of metrics is analysed in the search of inconsistencies, lack of uniformity, ambiguities, etc. The domain ontology is extensively used during this activity. As a result, the suite is reformulated: definitions are clarified and eventually some metric may experiment some change (e.g., the subject over which the metric is applied may change).

### 3.3 Step 3: $i^*$ Metrics Formulation

This step makes operative the metrics in terms of  $i^*$  constructs following the mapping from the domain ontology to the  $i^*$  metamodel established in Step 2:

- *Activity 3.1: Map the Metrics onto the  $i^*$  Metamodel.* The formulation of the metric is analysed and the definition rephrased in terms of the  $i^*$  metamodel.
- *Activity 3.2: Express the Metrics in OCL.* The rephrased definition is made operative as OCL formulae expressed over the  $i^*$  metamodel class diagram taking the integrity constraints into account.
  - *Activity 3.2.1: Apply Language of Patterns.* Patterns from the  $i$ MDF catalogue are identified and applied wherever possible. In fact, it is expected that patterns cover most of the situations to be faced during the process of metrics definition, making thus this process easier.

### 3.4 Step 4: $i$ MDF Update

Last, the result of the process is analysed to learn more about the method and the whole framework. This step combines three different activities:

- *Activity 4.1: Update Statistics.* Statistics refer specially to applicability of the patterns that form our language and are used in the Activity 4.2 below.
- *Activity 4.2: Update Language of Patterns.* At some moment, as a result of the accumulated statistics, patterns seldom used may be removed, or may be reformulated. Also, most used patterns may be further analysed for possible specializations. Last, some new patterns may be added after the current process.
- *Activity 4.3: Update Metric Catalogue.* Finally the decision to include or not the result of the current process has to be taken. Although the usual case should be to add the obtained metrics to the catalogue, we could eventually discard the suite for some reason (e.g., concerns on the mapping from the domain ontology to the  $i^*$  metamodel). Also, it could be the case that some particular metric is removed from the catalogue.

## 4 Applying $iMDF_M$ on a Business Process Modeling Metrics Suite

Balasubramanian and Gupta consolidated a metrics framework composed of eight metrics for business process design and evaluation [6]. These metrics come from others’ proposals (remarkably Nissen [14, 15], and Kueng and Kawalek [16]) and address performance aspects such as process cost, cycle time, process throughput and process reliability. In this section we apply  $iMDF_M$  over this suite of metrics.

### 4.1 Step 1: Domain Analysis

[6] proposes a 3-view model for business processes. The *workflow view* reveals the sequence of constituent activities and the business participants that execute them. In business processes, an activity may be defined as “work that a company performs” [17]. This is quite similar to the notion of *task* in the  $i^*$  framework, so we establish this fundamental equivalence (see Table 1). A sequence of activities may be thought as a particular kind of *routine* in  $i^*$ , i.e., a sequence of intentional elements that are inside some actor’s boundary in the Strategic Rational (SR) view of the  $i^*$  model. To represent a sequence of activities, the routine must fulfill: 1) its components are just tasks; 2) constraints expressing precedence relationships are included; 3) there exists one and only one initial task. In particular, if task T2 goes after task T1, we assume a constraint Follows(T2, T1); if task T branches into T1, ..., Tk,  $k > 1$ , we assume  $k$  constraints Branches(T, T1), ..., Branches(T, Tk). We consider also a Join constraint which acts the opposite than Branches. Participants are represented as actors.

**Table 1.** Mapping among the concepts on [6] and the  $i^*$  metamodel constructs

Business Process Ontology according to [6]		$i^*$ Metamodel
View	Concept	$i^*$ Element
Workflow view	Activity	Task
	Sequence of activities	Routine
	Business participant	Actor
Interaction view	Interaction	Resource dependency
	Business segment	Actor
	Operation	Is-part-of
Stakeholder-state view	Process stakeholder	Actor
	Milestone	Goal
	Visibility need	Goal placement + dependency

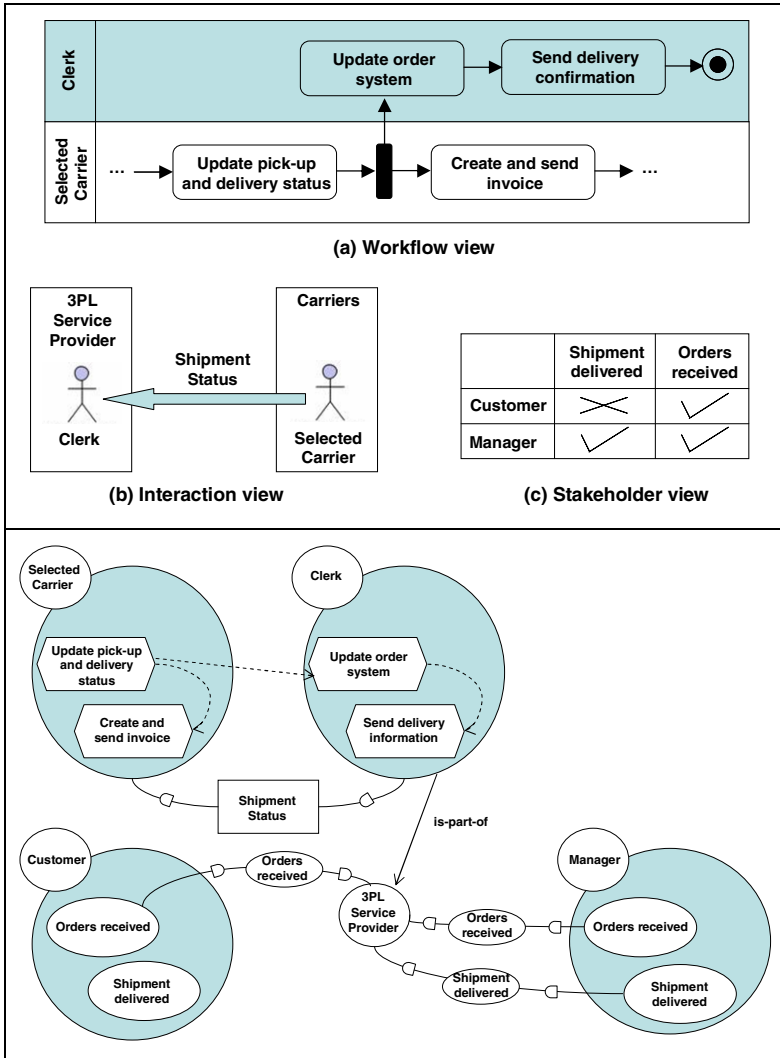


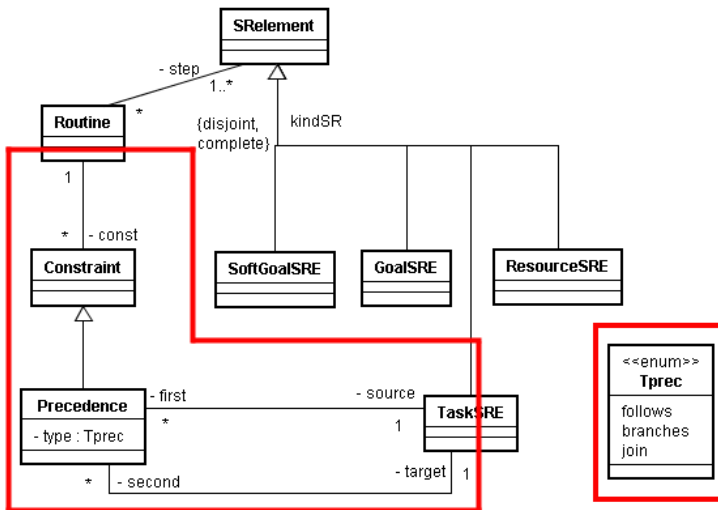
Fig. 2. An extract of process model according to the 3 views proposed by Balasubramanian and Gupta (up) and its mapping onto the *i\** framework using the mapping given in Table 1 (down)

The *interaction view* reveals the interaction among the business participants and the business segments in which they operate. Interactions take the form of information transmitted between participants (e.g., transportation order, invoice, shipment status); therefore they may be modeled as resource dependencies: when the participant A interacts with B for transmitting the information C, we say that the actor B depends on A by means of a resource dependum C. For stating that a participant operates in a business segment, we again may represent segments by actors and represent this “operating” notion using a “is-part-of” relationship from the participant actor to the segment actor. We assume that a service provider will always be present in the model.

The *stakeholder-state* view reveals the important process stakeholders and the fulfillment of their process visibility needs with respect to identified process states or milestones. Again stakeholders may be modeled as actors. Milestones (e.g., orders received) can be represented as goals in *i\**, placed inside the boundaries of those process actors that must satisfy the goal. If a stakeholder A has a visibility need with respect to milestone M, we include M also in A's boundary; if the need is satisfied, then we establish a dependency from that goal to the business segment corresponding to the service provider actor.

Each business process is constituted by these three views. In the usual case, the model will include several business processes whose views will coexist in the *i\** model. Therefore, routines, stakeholders and milestones will appear altogether. Tasks may be part of different routines, with different constraints in the general case.

Fig. 2 shows an extract of a process model appearing in [6] and its correspondence in *i\** according to the explanation above. The workflow view generates 2 actors and 4 tasks, as well as several precedence constraints (represented in the model as dotted arrows instead of textually) reflecting the activity relationships. The interaction view shows one interaction involving the same two actors, generating a resource dependency between them. Also, the operation of business participants in business segments appears in the view (just one of them has been represented in the *i\** model by means of a is-part-of relationship). Last, the stakeholder view shows some milestones of the different stakeholders and three of them are satisfied in the given process model (so the fourth one will be unsatisfied or alternative means for satisfaction should be explored).



**Fig. 3.** Extract of the *i\** metamodel as defined in [11] including the extension for Balasubramanian and Gupta's suite of metrics (framed). Integrity constraints not included. The complete class diagram can be downloaded from [www.lsi.upc.edu/~franch/supplementaryMaterial/iStarMetaModelWithBPMconstructs.pdf](http://www.lsi.upc.edu/~franch/supplementaryMaterial/iStarMetaModelWithBPMconstructs.pdf).

The  $i^*$  metamodel must be customized to this mapping (this customization will be completed in Step 2):

1) We add an OCL invariant for restricting routine steps to tasks.

2) A subclass of *Constraint* named *Precedence*, with an attribute type that takes values from {Follows, Branches, Join}, is added (see Fig. 3, where the changes are framed; the whole metamodel is not included for space reasons, see [11] for details). It has two roles bound, *source* and *target*, to *tasks* that belong to the same routine than the constraint. New integrity constraints for avoiding error conditions (e.g., loops, joining disjoint paths) must also be added, as well as an integrity constraint for ensuring that there is just one initial state (a task that is not target of any other).

3) Resource dependencies inferred from the interaction view must be aligned with the precedence relationships stated in the workflow view. If a resource dependency stems from actor A to actor B, then some of B's activities must be executed before some A's activity according to those relationships. An integrity constraint ensures it.

It must be mentioned that the  $i^*$  model generated with this mapping does not present a lot of fundamental concepts: softgoals, resources inside actors' boundaries, links inside actors' boundaries, etc. We could have chosen to add some integrity constraints (or directly to prune the metamodel) to reflect this fact, but we think that better not: the modeler may decide to add this information to exploit fully the capabilities of the  $i^*$  framework. This has an important consequence when formulating the metrics: to be general enough, we have to consider this fact and in particular, we mention that allowing decomposition of tasks inside SR diagrams will have an impact on the final form that metrics will take in  $i^*$ .

## 4.2 Step 2: Domain Metric Analysis

We summarize next the business process metrics framework proposed in [6]. Since the departing proposal was quite uniform, consolidation was straightforward, i.e. definitions are basically quotations from that paper except in one case (APF, see below); for sake of brevity we do not provide the rationale for the metrics, see [6] for discussion. Underlined terms stand for concepts added to the domain ontology.

- *Branching Automation Factor* (BAF). Proportion of decision activities in a process that do not require human intervention. A decision activity is an activity that branches into several others in the workflow view.
- *Communication Automation Factor* (CAF). Proportion of inter-participant information interchanges in a process where the information source is a system.
- *Activity Automation Factor* (AAF). Proportion of total activities in a process that are either interactive or automated. An interactive activity is an activity performed by a human actor and assisted through a system. An automated activity is one that is performed entirely by a system.
- *Role Integration Factor* (RIF). Ratio of number of activities performed by a process actor where the process control is not passed to another participant within the same organization to the total number of activities performed by that actor.
- *Process Visibility Factor* (PVF). Proportion of number of process states required to be visible to process stakeholders that are actually reported to or recorded for the relevant stakeholders. A process state is a point where a milestone is achieved.

- *Person Dependency Factor* (PDF). Proportion of activities performed by human participants that are executed using human discretion within the entire process. In [6], human discretion is an attribute of activities in the workflow view.
- *Activity Parallelism Factor* (APF). Proportion of activities that are executed in parallel in a process. We think that this definition is not much accurate, for instance given a process with 5 activities T1, ..., T5 the result would be the same for one process with branches (T1, T2), ..., (T1, T5) and another with (T1, T2), (T1, T3), (T2, T4), (T2, T5). Therefore, we prefer to adopt the definition by Nissen [14] as the length of longest path of activities that must be executed sequentially divided by the total number of activities.
- *Transition Delay Risk Factor* (TDRF). Ratio of the number of activity control transitions to any human participant to the total number of transitions between participants in a process.

It is worth to mention that one of the metrics, RIF, is different than the others, since it is local to roles. For finishing this unifying step, we define an augmented version of RIF, RIF+, as the average of the local measures of RIF for all human roles.

### 4.3 Step 3: $i^*$ Metrics Formulation

In this section we tackle  $i^*$  metrics formulation. We base this step on the  $i$ MDF language of patterns [5], see Table 2 below for a sample. For space reasons, we cannot present the metrics in detail. For illustration purposes, we show the two most difficult cases that may give an upper bound of the complexity of the process.

**AAF.** The main pattern applied is Normalization (shown in Table 2). The *Elem* is the  $i^*$  Routine that represents the business process under analysis in the  $i^*$  model, and the resulting *Type* is a float number. The *Size* is the number of tasks in that Routine:

```
Size ::= self.step.oclAsType(TaskSRE)-> size()
```

**Table 2.** Examples of patterns (given in abridged form, see [5] for details)

Category	Name	Description
Declaration	Individual	The metric applies just to one type of element, <i>Elem</i>
<b>context</b> <i>Elem::metric(): Type</i>		
Definition	Sum	The element metric's value is the sum of its components' values
<b>context</b> <i>Aggregated::metric(): Type</i>		
<b>post:</b> result = self.aggregees().metric()->sum()		
Numerical	Normalization	The metric needs to be restricted to some interval
<b>context</b> <i>Elem::metric(): Type</i>		
<b>post:</b> <i>Size</i> = 0 implies result = 1.0		
<b>post:</b> <i>Size</i> > 0 implies result = <i>Value</i> / <i>Size</i>		
Navigational	Property evaluation	The value of some property is needed
<b>context</b> <i>Node::propertyEval(name: String): Type</i>		
<b>pre:</b> self.value->select(v   v.property.name = name)->size() = 1		
<b>post:</b> self.value->select(v   v.property.name = name).val		

For *Value*, it must be noted that an automated activity is represented in an *i\** model as a task that: 1) belongs to a Software actor and, 2) the task and all of its subtasks, if any, have dependencies just to other Software actors. An interactive activity is a task that: 1) belongs to a Human actor and, 2) the task itself, or some of its subtasks, has some dependency going to a Software actor:

```
Value ::= self.const.oclAsType(Precedence)->select(t | t.interactive() or t.automated())->size()
```

The refinement of the auxiliary operations is given below; they are obtained by applying the following patterns: *isHuman* and *isSoftware*, applying *propertyEval* (see Table 2); *allSubtasks*, applying *TopDownDecomposition* over task-decomposition links; *requiresSoftware*, applying *TransitiveCheck* (if task T1 depends on task T2, the condition holds either if T1 is inside a human actor, or if T2 or any of its subtasks depends on a human actor). Other auxiliary patterns were also applied.

```
Task::interactive() ::= isHuman(self.owner) and
                    exists(t | self.allSubtasks()->includes(t) and t.requiresSoftware())
```

```
Task::automated() ::= isSoftware(self.owner) and
                    forAll(t | self.allSubtasks()->includes(t) implies not t.requiresSoftware())
```

**APF.** Again we apply the Normalization pattern, computing the length of the longest path of the business process (i.e., Routine) and dividing by the total number of activities. Computing the length of the longest path is not straightforward due to possible branches and joins:

```
Elem ::= Routine; Type ::= Float
Size ::= self.step.oclAsType(Task)->size()
Value ::= self.allPaths()->select(p | self.allPaths()->forAll(p2 | size(p) >= size(p2))) ->size()
Routine::allPaths() ::= self.step->select(t | t.initialActivity()).allPaths()
Task::allPaths() ::= if self.finalActivity() then self
                    else self.allDirectSuccessors().allPaths()->prepend(self) end-if
Task::allDirectSuccessors() ::= self.firstComp->select(source=self).target
Task::finalActivity() ::= self.firstComp->size() = 0
```

#### 4.4 Step 4: *i*MDF Update

Special importance takes the update of the language or patterns. It is still too early in our research to decide the removal of some pattern from the language, although some were not used in this particular case. Concerning the discovery of new patterns, we remark that 7 out of the 8 metrics were defined by a similar application of the *Normalization* pattern as done with AAF:

- The *Size* parameter is the size of a collection of *i\** elements, e.g. the collection of all tasks in AAF, the collection of all stakeholder goals in PVF, etc.
- *Value* is defined by applying a filter (i.e., an aggregation operation such as Sum, Count, etc.) over the same collection than before.
- As a consequence, the *Type* is a float (in the interval [0, 1]).

Let's call *Col* that collection. Thus, the form that the *Normalization* pattern takes in these seven metrics is:

```

context Routine::metric(): Float
    post: Col->size() = 0 implies result = 1.0
    post: Col->size() > 0 implies result = Col.filter()->size() / Col->size()
    
```

Given its rationale, it is reasonable to expect that this variation of the Normalization pattern will be useful in future experiments and case studies. This is why we have decided to enlarge our pattern language with this expression upgraded to pattern (specialization of the *Normalization* pattern –we have specializations in our pattern language), just abstracting *Routine* to *Elem*.

Concerning the metrics catalogue, we incorporated this new suite.

## 5 Observations

In this section we summarize the key observations on this application of the  $iMDF_M$  method and we try to extract some general risks and facts summarized in Table 3.

### 5.1 Step 1: Domain Analysis

This first step was really crucial for the success of the experiment, even more than expected beforehand. We observed two different sources of difficulties, corresponding to the two identified activities.

**Creation of the domain ontology.** In the 3-view model of business processes proposed in [6], we faced the problem of model integration (risk R1 in Table 3). We found two concrete difficulties:

**Table 3.** Risks ( $R_i$ ) and facts ( $F_j$ ) that may appear during the application of  $iMDF_M$

Step	Act.	Risk / Fact
Domain Analysis	1.1	R1 Need of aligning different types of domain models that do not match exactly
	1.2	R2 Lack of some of the $i^*$ expressive power in the domain ontology
		R3 Some concepts of the domain ontology cannot be directly mapped to $i^*$
Metrics Analysis	2.1	R4 The metrics suite is not completely aligned with the domain ontology
	2.2	R5 The suite of metrics is not uniformly defined
		R6 Some metric is not accurately defined and demands further investigation
Metrics Formulation	3.1	R7 Definition of many properties in the metamodel needed for mapping metrics
		R8 Mapping of metrics defined over an $i^*$ metamodel richer than strictly needed
	3.2	R9 Some inherent characteristics may make the process harder (e.g., transitivity)
		F1 As the process progresses, reuse of concepts and OCL facilitates the process
		F2 The language of patterns is a good starting point for the metrics definition
$iMDF_M$ update	4.1	F3 Keeping track of pattern use statistics is essential for maintaining the framework
	4.2	F4 Upgrading an OCL expression into patterns mostly depends on frequency of use
	4.3	F5 Incorporating the result of the process into the catalogue will be the usual case



- The relationship among interactions in the interaction view and transitions in the workflow view is not explicit. If just one transition exists among two actors in the workflow view, it may be inferred, but otherwise the link must be established by observation or even it may require further investigation.
- The relationship among the milestones in the stakeholder view and the activities in the workflow view is not explicit. In this case, it is much harder to try to observe the link.

Both are instances of the same generic difficulty: aligning different types of models that are part of the departing domain.

**Mapping onto the  $i^*$  metamodel.** Basically we found two types of difficulties:

- Those coming from the departing ontology. If we assume that a business process model cannot be modified, the consequence is that the resulting  $i^*$  model is not as rich as it could be (risk R2). For instance, as a consequence of the observations above, resource dependencies cannot be established at the level of intentional elements but just at the level of actors. We may argue that this is not a problem since our goal is to define metrics on the  $i^*$  models that are equivalent to the original ones. Thus if the departing metrics were formulated without needing this information, we can do the same over the  $i^*$  models. This being true, we also think that wasting some capabilities of  $i^*$  models may make the approach less useful and less attractive. We envisage two solutions: 1) to refine the departing ontology; 2) to refine the resulting  $i^*$  model adding the missing information. Trade-offs between all the options should be considered in detail before taking any decision.
- Those coming from the fundamental differences between the domain ontology and the  $i^*$  metamodel (risk R3). Here, we have succeeded in translating all the constructs from the business process case, even those that had not a direct counterpart in  $i^*$ , by enriching the  $i^*$  metamodel. Enriching the metamodel means losing some kind of standardization, but as shown e.g. in [11], there are a lot of variants in the  $i^*$  framework and this is one of the features that makes  $i^*$  attractive.

## 5.2 Step 2: Domain Metrics Analysis

**Extension of the Domain Ontology.** In the definition of the departing metrics appeared some concepts that were not present in the ontology after Step 1 (risk R4). Notions like “decision activity”, “parallel execution of activities” and “process state” were not described precisely enough in the framework and we were forced to set our own interpretation of these terms. Other concepts like “activity control transitions” and “passing process control” had to be carefully examined. As a result, our [6]-based business process ontology grew.

**Consolidation of the Metrics Suite.** The definition of the metrics needed to be examined in detail. In our case, we found risks that may arise in future situations:

- Non-uniform definition of the metrics suite (risk R5). Uniformity is a fundamental property for conceptual frameworks. In our case, the metric RIF was clearly different from the others since it focused not just on a business process but also on a role. Thus, it did not fit with the overall goal of the metric framework, namely evaluating business processes. As a result, we proposed a slightly modification RIF+, although we also kept RIF to be respectful with the original proposal.

- Not accurate definition of a metric (risk R6). We look for metrics giving as much relevant information as possible, thus we were not happy with the definition of APF given in [6] and we preferred the original definition in [14], even paying the price of having a metrics quite different in structure than the others, therefore hampering somehow the uniformity criteria stated above.

### 5.3 Step 3: $i^*$ Metrics Formulation

**Mapping the Metrics onto the  $i^*$  Metamodel.** Firstly, an issue is to what extent we need to add information (represented by properties in the  $i^*$  metamodel) to  $i^*$  models (risk R7). Too many properties would eventually require a lot of effort in the definition. In this case, we just needed 3 properties for the 8-metric framework. One of them, *Nature*, for knowing the type of an actor (human, software, etc.) was already introduced in *iMDF* before this experiment. Another one, *Process-Stakeholder*, to know the actor that owns a routine, has a high probability of reuse. Both of them are quite straightforward to evaluate. Thus their need is not really a strong drawback in terms of effort. The third one, *DecisionActivity*, to check if an activity is a decision activity, could be more difficult to handle in the general case but, in the departing proposal, decision activities are explicitly labeled as such. So, this third property does not raise any relevant problem neither (although it has a lower chance of reuse).

On the other hand, we have defined our metrics without considering some simplifications of the model to make them more robust (risk R8). Just to mention an illustrative example, in some metrics we have considered that tasks could be decomposed into subtasks although the departing framework as defined in [6] did not mention this case.

**Expression of the metrics in OCL.** Two of the most error-prone and cumbersome characteristics to face are transitive closure and transitive definition of some operations (risk R9). Illustrative examples are: for the first case, the generation of all the paths or subtasks; for the second case the analysis of chains of dependencies.

In the positive side, as the definition of metrics progresses, it becomes easier to write them (fact F1). Two related reasons behind: the flavor of the metrics is similar after Step 2, and also some OCL expressions may be reused.

**Use of the pattern language.** For the definition of metrics itself, we used intensively the pattern language. The detailed results are given in the next subsection, but as a kind of summary we are quite happy with the results, the language demonstrated to be powerful and versatile enough (fact F2).

### 5.4 Step 4: *iMDF* update

**Updating statistics.** We have applied 59 patterns to define the 8 metrics (without considering RIF+). Each metric needs two declaration patterns (one for the context, other for the type) and a third pattern applied is *Proportion (Normalization)* in the case of APF). The most complex in terms of number of applications has 11 whilst APF has just 4 (because we couldn't solve `allPaths()` by patterns) and next, PDF has 6. If we consider RIF+, we add 6 new applications of patterns. Keeping track of this statistics provide useful insights to the *iMDF* framework (fact F3).

**Updating the pattern language.** We have been able to formulate most of the metrics using intensively our pattern language. During the experiment, we faced two different expressions that could be upgraded into patterns. As seen in section 4.4, we defined a new pattern *Proportion* due to its intensive use in this framework and the conjecture that the situation dealt is likely to happen in the future. On the contrary, the *allPaths()* operation needed in APF, which was difficult and done ad-hoc, seemed so particular that we decided not upgrading it into a pattern (fact F4).

**Updating the catalogue of metrics.** For the metrics catalogue, since all the metrics were successfully solved, the whole suite of metrics could be incorporated into the catalogue. After the several experiences we have had, this is expected to be the usual case, provided that the whole experiment makes sense (fact F5).

## 6 Conclusions and Future Work

In this paper we have presented a method for defining metrics in  $i^*$  using the *iMDF* framework. Since we are interested in porting already existing, validated metrics to  $i^*$ , the method is largely concerned with the analysis of the domain and the metrics themselves, and the mapping onto the  $i^*$  metamodel, more than on design of completely new metrics, which would a different matter of research. The method has been articulated by defining the relevant activities (organized into steps) and the artifacts involved. We have identified some risks and facts that may be used in future cases.

In addition, this paper has fulfilled a second goal, to offer a new suite of performance metrics for business process models represented in  $i^*$ . This new suite enforces our current catalogue in a domain we hadn't addressed before. Our language of patterns has been enlarged and we have obtained more statistical data about pattern use.

As future work, we are planning new experiments on different fields to the method and the whole *iMDF* framework whilst offering an increasingly large catalogue of metrics to the community. The experiments shall also assess the effort required to use this approach; this is a crucial validation to perform, since *iMDF* requires knowledge in: domain analysis, ontology construction, metamodeling and metrics. On the other hand, about implementation, after a first prototype available over an existing tool, we are starting to build a new tool taking advantage of the recent proposal of an XML-like standard for encoding  $i^*$  models called *iStarML* [18]. Our plans are to build the tool able to import models expressed in an *iStarML*-based grammar (the codification of the customization of the  $i^*$  metamodel). Translators from other models to *iStarML* (following the rules coming from Steps 1 and 2 of the process) would allow evaluating metrics over models built in the departing ontology.

## References

1. van Lamsweerde, A.: Goal-oriented Requirements Engineering: A Guided Tour. In: Proc. 5th ISRE Intl' Symposium. IEEE, Los Alamitos (2001)
2. Yu, E.: Modelling Strategic Relationships for Process Reengineering. PhD Dissertation, Univ. of Toronto (1995)

3. Giorgini, P., Mylopoulos, J., Nicciarelli, E., Sebastiani, R.: Formal Reasoning Techniques for Goal Models. In: Spaccapietra, S., March, S.T., Kambayashi, Y. (eds.) ER 2002. LNCS, vol. 2503. Springer, Heidelberg (2002)
4. Sebastiani, R., Giorgini, P., Mylopoulos, J.: Simple and Minimum-Cost Satisfiability for Goal Models. In: Persson, A., Stirna, J. (eds.) CAiSE 2004. LNCS, vol. 3084, pp. 20–35. Springer, Heidelberg (2004)
5. Franch, X., Grau, G.: Towards a Catalogue of Patterns for Defining Metrics over  $i^*$  Models. In: Bellahsène, Z., Léonard, M. (eds.) CAiSE 2008. LNCS, vol. 5074, pp. 197–212. Springer, Heidelberg (2008)
6. Balasubramanian, S., Gupta, M.: Structural Metrics for Goal Based Business Process Design and Evaluation. *Business Process Management Journal* 11(6) (2005)
7. Kaiya, H., Horai, H., Saeki, M.: AGORA: Attributed Goal-Oriented Requirements Analysis Method. In: Procs. 10th RE Intl' Conference. IEEE, Los Alamitos (2002)
8. Sutcliffe, A., Minocha, S.: Linking Business Modelling to Socio-technical System Design. In: Jarke, M., Oberweis, A. (eds.) CAiSE 1999. LNCS, vol. 1626, p. 73. Springer, Heidelberg (1999)
9. Bryl, V., Giorgini, P., Mylopoulos, J.: Designing Cooperative IS: Exploring and Evaluating Alternatives. In: Meersman, R., Tari, Z. (eds.) OTM 2006. LNCS, vol. 4275, pp. 533–550. Springer, Heidelberg (2006)
10. Franch, X., Maiden, N.A.M.: Modelling Component Dependencies to Inform Their Selection. In: Erdogmus, H., Weng, T. (eds.) ICCBSS 2003. LNCS, vol. 2580. Springer, Heidelberg (2003)
11. Ayala, C.P., et al.: A Comparative Analysis of  $i^*$ -Based Goal-Oriented Modeling Languages. In: Procs. 17th SEKE Intl' Conference, KSI (2005)
12. Franch, X.: On the Quantitative Analysis of Agent-Oriented Models. In: Dubois, E., Pohl, K. (eds.) CAiSE 2006. LNCS, vol. 4001, pp. 495–509. Springer, Heidelberg (2006)
13. Grau, G., Franch, X.: A Goal-Oriented Approach for the Generation and Evaluation of Alternative Architectures. In: Oquendo, F. (ed.) ECSA 2007. LNCS, vol. 4758, pp. 139–155. Springer, Heidelberg (2007)
14. Nissen, M.E.: Valuing IT through Virtual Process Measurement. In: Procs. 15th ICIS Intl' Conference. ACM, New York (1994)
15. Nissen, M.E.: Towards Enterprise Process Engineering: Configuration Management and Analysis. NPS Technical Report, NPS-GSBPP-02-003 (2002)
16. Kueng, P., Kawalek, P.: Goal Based Business Process Models: Creation and Evaluation. *Business Process Management Journal* 3(1) (1997)
17. White, S.A.: Introduction to BPMN. Report at BPMN website (2004), <http://www.bpmn.org/Documents/Introduction%20to%20BPMN.pdf>
18. Cares, C., Franch, X., Perini, A., Susi, A.: iStarML: An XML-based Model Interchange Format for  $i^*$ . In: Procs. 3rd  $i^*$  Intl' Workshop, CEUR Workshop Proceedings, vol. 322 (2008)

# Preference Model Driven Services Selection

Wenting Ma, Lin Liu, Haihua Xie, Hongyu Zhang, and Jinglei Yin\*

Key Lab for Information System Security, MOE  
Tsinghua National Lab for Information Science and Technology  
School of Software, Tsinghua University

\* School of Computer and Information Technology,  
Beijing JiaoTong University Beijing, China, 100084  
Tel.: +86(10)62773284

{mwt07@mails., linliu@, xiehh06@mails., hongyu@}tsinghua.edu.cn

**Abstract.** Service, as a computing and business paradigm, is gaining daily growing attention, which is being recognized and adopted by more and more people. For all involved players, it is inevitable to face service selection situations where multiple qualities of services criteria needs to be taken into account, and complex interrelationships between different impact factors and actors need to be understood and traded off. In this paper, we propose using goal and agent-based preference models, represented with annotated NFR/i\* framework to drive these decision making activities. Particularly, we present how we enhance the modeling language with quantitative preference information based on input from domain experts and end users, how softgoals interrelationships graph can be used to group impact factors with common focus, and how actor dependency models can be used to represent and evaluate alternative services decisions. We illustrate the proposed approach with an example scenario of provider selection for logistics.

**Keywords:** Services Selection, Goal Modeling, Decision Making, Actor Dependency Network.

## 1 Introduction

Service-oriented computing has gained its momentum in recent years both as a new computing and business paradigm. In the services environment, the social nature of software services gets fully embodied when related players make decisions during service discovery, publication, selection and revocation. Current techniques, framework and ontology for services selection mainly focuses on qualities of services at the system level rather than services qualities related to the high-level business objectives and contexts.

In the case of the supply-chain management domain, there are many well established theories and computational models for making integrated planning and optimal decisions for a given business service planning problem. Together with the new and fast development of the business world and information technology, supply-chain management concepts emerge as an important underlying model for all business services that involves manufacture and distribution of products. In this spirit, this line of thinking can easily migrate into the services world, that is, we view service requestors

and providers as agents with intent, who evaluate alternative ways to form a suitable service network based on runtime demand.

Multi-criteria decision making problems exist everywhere, and many researchers have been working on multi-motive decision models and methods. Some focus on the decision process itself and some work on domain-specific applications [10, 15]. Few research efforts have been made to model and analyze multi-criteria decisions in a systematic way, i.e., from the very beginning step of identifying the criteria, to the last step of making the final decision. In this paper, we adopt graphical notations of the NFR modeling methods to model the interrelations among different criteria, and then we add numerical annotations to the nodes in the model to represent preferences of decision makers. Actor dependency models in  $i^*$  can be used to represent and evaluate alternative services networking decisions. Algorithms for reasoning about these models to identify optimal solutions for the given decision problem is also given. We illustrate the proposed approach with an example scenario of provider selection for logistic services.

The structure of this paper is organized as follows: Section 2 presents a method for setting up and analyzing services selection criteria with annotated NFR. Section 3 introduces a decision making method based on agent-based preference model and criteria. Section 4 discusses related work and concludes the paper.

## 2 Setting Up Services Selection Criteria with Annotated NFR

Generally speaking, given a conventional multi-criteria decision problem, the very first step is to identify the essential impact factors to be included in the selection criteria in an extensive way. Besides, rough evaluation about the importance of these criteria could be in place. In order to collect decision criteria, focus group meetings, brain-storming sessions, literature reviews, and subject matter expert (SME) interviews can all be sources of the initial list criteria. No matter how trivial a factor looks like, as long as it has substantial influence, it should be taken into consideration.

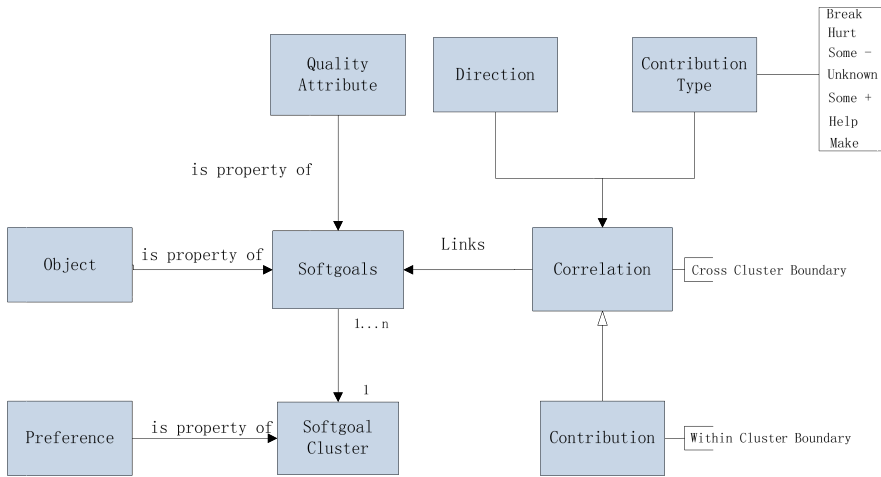
When facing the service-oriented context, we first need an online community of experts contribute a QoS ontology for the particular business service domain, which allows service agents to compare the available services and making selections based on their preferences. Providers can express their capabilities and consumers can express their preferences using this ontology. In this paper, we base our upper ontology of business service QoS on the Non-Functional Requirements (NFR) Framework [1] in general, and we use domain-specific NFRs as a lower ontology for particular business services.

### 2.1 An Upper Ontology for Services Selection Based on NFR Framework

Fig. 1 shows the upper ontology for services selection we build based on concepts from the NFR framework. The notion of softgoal in NFR is used to denote a factor of services selection criteria. Correlation links, which associate two softgoals, illustrate how the satisfaction of one softgoal contributes to that of another. The correlation element has two attributes: direction and type. If there is a correlation link from softgoal A to softgoal B, it means A has some influence on B. There are seven types of

contributions existing in conventional NFR framework, namely, break, hurt, some -, unknown, some+, help and make, representing to what extent, one softgoal contributes to another. For example, break means a full negative impact, while make means there is a full positive impact, hurt means a partial negative, help means a partial positive impact, some+ means a negative impact with unknown degree, some- means a positive impact with unknown degree, and unknown means it is uncertain what relationship exists between the two softgoals, but there is a relation.

Based on the basic NFR concepts we have made an extension. The concept of Softgoal Cluster is introduced, which consists of one or more softgoals and denotes softgoals with a common theme or subject. Softgoals belonging to one softgoal cluster has characters in common or stronger correlation, which is substituted with contribution links in NFR. To each Softgoal Cluster, a Preference value is associated to it as a quantitative attribute, which denotes expert preference to it.



**Fig. 1.** An upper ontology for services selection based on NFR framework

## 2.2 Source of Lower Ontology –A Service Example from the Logistics Domain

Taking the example of logistic services, there are many well-known catalogues in the literature we may ground the lower ontology on. There is a 23-parameter criterion for vendor selection identified by Dickson in 1966[2]. Based on his investigation of 273 procurement managers, which 170 feedbacks were received, the 23 parameters are ranked based on importance as in table 1. Weber collected and analyzed literature in this domain between 1967 and 1990, and re-ranked the 23 criteria based on their popularity and frequency of reference [8]. The detailed parameters and their comparative importance and popularity can be found in table 1. Since Dickson’s time, the criteria of vender selection have been hardly changed, and research is mainly focused on how to evaluate and select venders [14].

**Table 1.** 23 criteria and order of vender selection

Dickson's order	Importance	Criteria	Literature amount	Percentage	Weber's order
6	Important +	Price	61	80	1
2	Important +	Delivery	44	58	2
1	Important ++	Quality	40	53	3
5	Important +	Production facilities / capability	23	30	4
20	Important	Geographical location	16	21	5
7	Important +	Technical capability	15	20	6
13	Important	Management and organization	10	13	7
11	Important	Industry reputation and position	8	11	8
8	Important +	Financial position	7	9	9
3	Important +	Performance history	7	9	10
15	Important	Repair service	7	9	11
16	Important	Attitude	6	8	12
18	Important	Packaging ability	3	4	13
14	Important	Operating controls	3	4	14
22	Important	Training aids	2	3	15
9	Important +	Bidding procedural compliance	2	3	16
19	Important	Labor relation record	2	3	17
10	Important +	Communication System	2	3	18
23	Important -	Reciprocal arrangement	2	3	19
17	Important	Impression	2	3	20
12	Important	Desire for business	1	1	21
21	Important	Amount of past business	1	1	22
4	Important +	Warranties and claim policies	0	0	23

As a consequence, a list of candidate criteria is identified. In the meantime, we also collect input about the relative importance of each criterion. Such kind of list can help rationalize the services selection decision process by conducting qualitative and quantitative analysis with evidence data. However, these criteria are still lacking an internal structure that represents the interdependencies among individual criterion. It may prevent us from understanding the impact criteria in more depth. So in next step, we should make a correlation analysis to all the criteria and build a lower services selection ontology based on this knowledge.

### 2.3 Multiple Criteria Correlation Analysis Using Conventional NFR Model

To analyze the correlation of criteria, goal-oriented modeling approach, such as NFR is used. As we mentioned in 2.1, two model elements in NFR are used in our correlation analysis, NFR softgoal and correlation link. Each criterion is modeled as an NFR softgoal, while correlation link denotes the relation of two softgoals. After finding all the direct relationship among criteria, we analyze the coupling and cohesiveness of the nodes in the diagram and divide them into several criteria group. Then, each criteria group is represented as a Softgoal Cluster.

During this step, we should note that only direct correlations should be marked, and derived/indirect ones should be omitted from the diagram since redundant relationships will prevent us from finding the right clusters of criteria. Relationships are



usually recommended by experts with profound understanding about problem domain. A negative relation should always have an importance level attached, because it can lead to a contradiction with existing criteria, or even groups of criteria identified in our analysis. If some factor is loosely coupled with others, different treatment should be taken based on the importance of this factor. For very important factors that cannot be omitted, it should stay in a group. Otherwise it can be omitted depends on how many factors are identified, and the distribution of relative importance represented as a percentage.

To illustrate that, we use the twenty three factors for vender selection listed in table 1. All factors are treated as non-functional requirements. After adding their relations and adjusting their position in the graph, we can get the following result (Fig. 2).

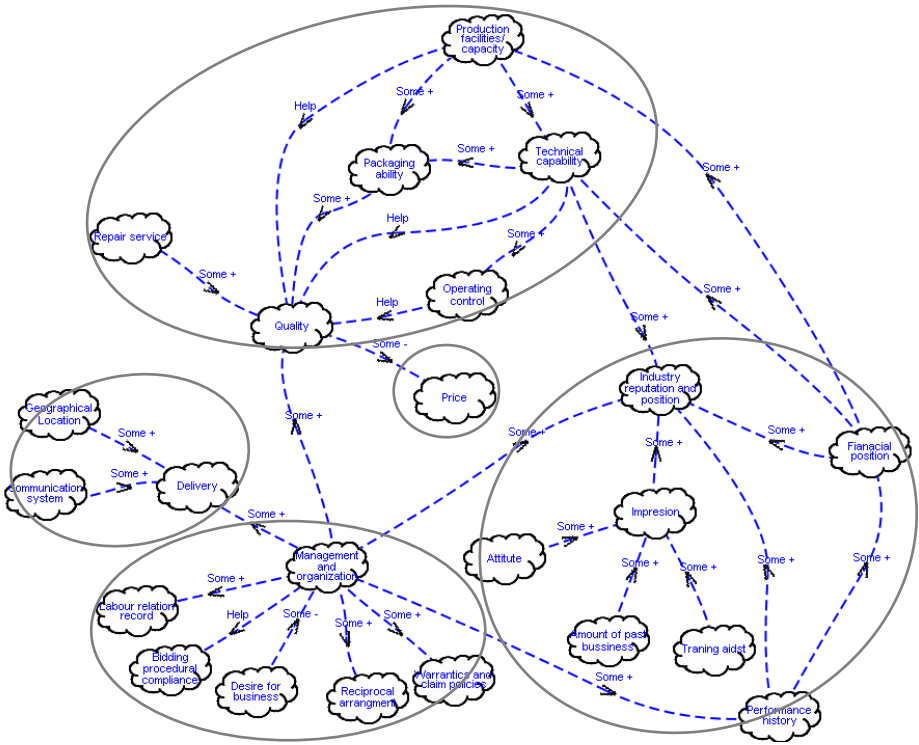


Fig. 2. Correlation analysis of services selection criteria

In our example, the categories are easily identified due to simplicity of the model. For complicated ones, assign weights to relations and automated clustering algorithms can be applied to identify clusters based on a customer-defined policy (people could adjust the result if needed). Each cluster should form a hierarchy with a root node representing a distinctive quality attribute and a few leaf nodes contributing/ influenced by the root node, and we use ellipse to indicate the boundary of each softgoal cluster. In our example, five categories are identified. We can use the root elements of them as the group name, and they are: delivery, quality, price, management and

reputation. But what is the relative importance of these groups? How can we denote the importance? These are the questions we want to answer next.

### 2.4 Preference Annotated Hierarchical NFR Model

It is very difficult to quantify the impact of services selection factors, which is soft in nature. Standardizing these quantified data and making them comparable is even harder. Continuing the thread of analysis in section 2.3, we use the softgoal clusters identified above as our decision criteria. Then we adopt AHP (analytic hierarchy process) method to calculate the weight of each factor, which is a critical decision making tool. Although there are possible drawbacks of AHP, such as poor scalability, incapable of reflecting intrinsic characteristics of complex evaluation issues, many researchers have proved that AHP is an easy-to-use and effective method to solve complex decision making problems. It is understandable not only to the experts, but also to general stakeholders. After the stakeholders and experts conduct a pair-wise comparison to the elements, their relative weights can be calculated. Then we can make an extension to the NFR framework by annotate each softgoal cluster with their weight. As shown in Fig. 3, we use the ellipse to denote the boundary of each cluster, and a cloud is associated to each cluster to show the name and the weight of cluster (format: cluster name: weight).

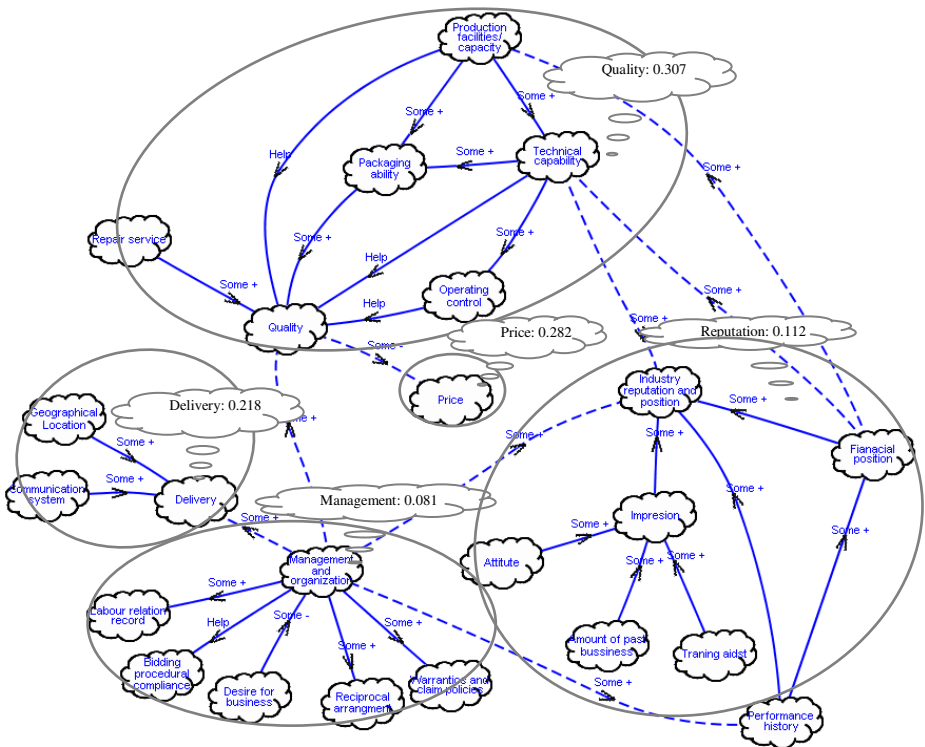


Fig. 3. Correlation analysis using annotated hierarchical NFR model

We use NFR method to analyze correlations of the selection criteria, and find out criteria groups standing for all the detailed factors. AHP method is used to find the weights of criteria groups. All these are the basis of the following decision making phase.

### 3 Services Selection Based on Preferences Modeling

Once the criteria for services selection decision are identified, the next step is to evaluate the performance of the candidates. First, we evaluate each single criterion of each candidate objects. Second, we use the results of single criteria evaluation to make decision considering factors from all categories. In this stage, the generic preference information from experts will be used.

*i*\* framework is a widely used strategic intentional modeling method. Actors' goals and tasks could be modeled intuitively. Services selection problem always involves at least two types of actors. And there are mutual dependencies between them. So *i*\* model [11, 12] is a natural fit for modeling services selection. In this section, we first introduce the quantification method, an approach called Weighted Set-Valued Statistics [10], and then we introduce a single criteria decision making model. Then in section 3.3, we introduce how to make multi-criteria decision based on the analysis result of 3.2. Sensitivity analysis is discussed as a further step of our method in 3.4.

#### 3.1 Weighted Set-Valued Statistics Approach to Calculate the Contribution of Candidate

In the services selection process, different candidate usually have different contributions when carrying out a same required task. So in order to calculate the satisfaction degree of softgoal, it is essential to know for a given task, what impact each candidate will result in. It can only be estimated by people with sufficient domain knowledge. In general, the principle they use to decide the value of *e* (*t*, *s*) is estimating the task *t*'s execution result which contributes to softgoal *s*. And the estimation method we introduce is weighted set-valued statistics.

Suppose several domain experts estimate the value of a certain *e* (*t*, *s*). The set of experts is represented with  $P = \{p_1, p_2, \dots, p_n\}$ . Each expert *p<sub>i</sub>* ( $1 \leq i \leq n$ ) gives an evaluation range for *e* (*t*, *s*), for simplicity, we use *q* to denote the value of *e* (*t*, *s*). The evaluation range of *e* (*t*, *s*) is as below:

$$[q_1^{(i)}, q_2^{(i)}], -1 \leq q_1^{(i)} < q_2^{(i)} \leq 1$$

The lower bound of the range means the worst execution impact, and the upper bound of the range means the best execution effect.

According to experts' evaluation, we define Projective Function as the equation below, which expresses *p<sub>i</sub>*'s evaluation range projecting onto the axis *x*:

$$Y_{[q_1^{(i)}, q_2^{(i)}]}(q) = \begin{cases} 1, & q \in [q_1^{(i)}, q_2^{(i)}] \\ 0, & \text{others} \end{cases}$$

Integrating all experts' Projective Functions, we can get the Set-Valued Function:

$$Y(q) = \frac{1}{n} \sum_{i=1}^n Y_{[q_1^{(i)}, q_2^{(i)}]}(q)$$

The narrower the range given by expert  $pi$  is the more confident is he about his evaluation. Under normal circumstances, that means his evaluation is more accurate, so his weight on this evaluation should be higher than others'. In order to distinguish from the weight of softgoal, we call it  $e$ -weight of expert. For designation, we use  $wi$  to denote the  $e$ -weight of expert  $pi$ . The value of  $wi$  is computed according to the evaluation range given by  $pi$ :

$$w_i = \frac{d_i}{\sum_{i=1}^n d_i}, \text{ in which } d_i = \frac{1}{(q_2^{(i)} - q_1^{(i)})}$$

Then we get the Weighted Set-Valued Function:

$$\bar{Y}(q) = \sum_{i=1}^n w_i Y_{[q_1^{(i)}, q_2^{(i)}]}(q)$$

The expanded form of the weighted set-valued function is:

$$\bar{Y}(q) = \begin{cases} a_1, q \in [b_1, b_2] \\ a_2, q \in [b_2, b_3] \\ \dots \\ a_L, q \in [b_L, b_{L+1}] \end{cases}$$

In this formula,  $b_1, b_2, \dots, b_{L+1}$  is an ascending sequence of all end points of evaluation ranges given by experts.  $L$  is the number of intervals divided by these end points.  $a_i$  ( $i=1, 2, \dots, L$ ) is the sum of  $e$ -weights of experts whose evaluation range contain the interval  $[b_i, b_{i+1}]$ .

According to Set-Valued Statistics, the expectant value of  $e$  ( $t, s$ ) is:

$$E(q) = \frac{\int_{b_1}^{b_{L+1}} \bar{Y}(q) q dq}{\int_{b_1}^{b_{L+1}} \bar{Y}(q) dq} = \frac{\frac{1}{2} \sum_{i=1}^L a_i (b_{i+1}^2 - b_i^2)}{\sum_{i=1}^L a_i (b_{i+1} - b_i)}$$

We expect experts' evaluations approach the actual value as closely as possible. In other words, the variance of all evaluations should be the lower the better. The variance is:

$$D(q) = \frac{\int_{b_1}^{b_{L+1}} \bar{Y}(q) (q - E_i(q))^2 dq}{\int_{b_1}^{b_{L+1}} \bar{Y}(q) dq} = \frac{\frac{1}{3} \sum_{i=1}^L a_i \{ [b_{i+1} - E(q)]^3 - [b_i - E(q)]^3 \}}{\sum_{i=1}^L a_i (b_{i+1} - b_i)}$$

We compute the standard deviation based on the variance:

$$S(q) = \sqrt{D(q)}$$

The smaller the standard deviation is the more accurate is the evaluations. We define credibility of set-valued statistics to represent the accuracy of evaluation:

$$B = 1 - \frac{S(q)}{E(q)}$$

If  $B \geq 0.9$ , we think experts' evaluations can be accepted and  $E(q)$  is considered to be the quantified value of  $e(t, s)$ . Otherwise, experts do not quit the evaluation process until the credibility's value reaches 0.9.

### 3.2 Services Selection Based on Single Criterion

There are always two classes of actors involved in the service decision making process. Both have their own goals and intentions. The dependencies representing the selection criteria become a bridge between them. To establish a dependency, actors need to perform certain tasks. To what extent the dependency or goal can be achieved depends on how each actor takes actions to operationalize these softgoals.

To evaluate the task execution result, related original data should be collected, which includes objective historical data related to the criteria. It can also be collected from domain experts or by a survey. Although these data is more subjective, it is helpful for our decision making process. Besides, standardized evaluation is also needed. Because we will compare different candidates, evaluation must give every facet a standardized quantitative mark. It is usually set up by a domain expert who understands the specific domain. Here, we choose weighted set-valued statistics approach introduced in 3.1 to calculate the contribution of candidates.

In order to illustrate how to use annotated  $i^*$  model to evaluate the achievement of each single criteria, procurement is used as an example. Due to space limitation, we will choose quality as the sole criterion to be considered. The analysis of other factors follows a similar process. For example, a company wants to procure some material. There is historical product data on quality- three suppliers' rates of return (Table 2). Because of cooperation and domain reputation, etc., other related data can also be saved as our primal evaluation data.

Depending on the different trust level or knowledge level, different situations could be modeled. Here we give two example cases based on different cooperation degree or trust level.

**Table 2.** Three suppliers' rates of return

	Supplier A	Supplier B	Supplier C
Rate of return (%)	2.6	3.8	3.2

**Scenario 1. Single point decision model.** In general, when the procurer only checks the overall rank of the concerned parameter, it is the supplier's responsibility to operationalize the standard. In this case, there are either little cooperation or strong trust between the procurer and supplier. The procurer could also check the products if it satisfies their need using certain preset standard. As shown in Fig. 4, annotated  $i^*$  models with related data can help us position the decision point in the right context. For single point decision model, the verdict is made based on whether the single soft-goal dependency is satisfied or not. The procurer defines a range, while suppliers provide their performance rate.

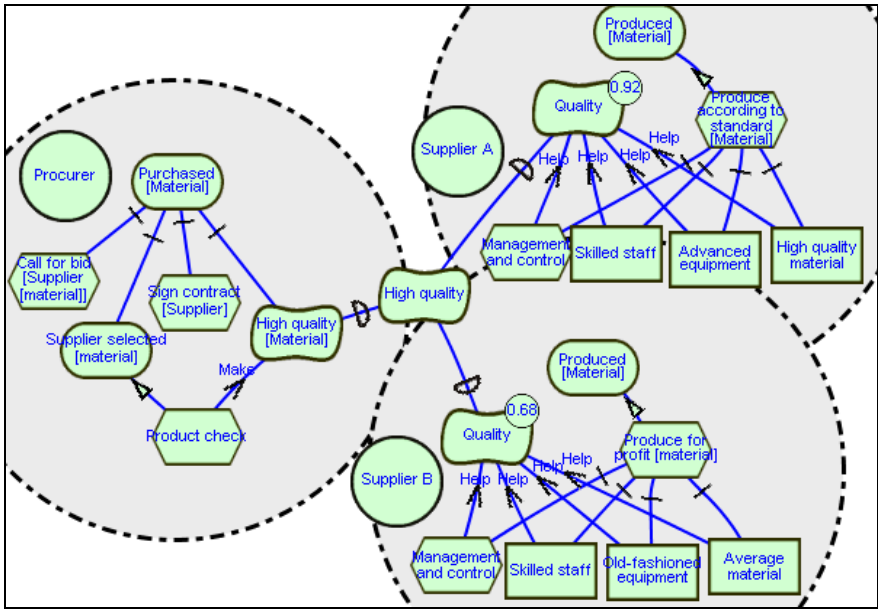


Fig. 4. Single point decision model in annotated  $i^*$

As we have mentioned before, a decision can't be made with only model and primal data. We need evaluation criteria. Here, expert knowledge is used to make such a standard. In our example, because there is no other detailed information about suppliers, what the experts could use is three suppliers rate of return. What experts should do is to quantify the rate to a standard score (0 to 1, for example), which could denote the contribution of each supplier. Assuming the usage of weighted set-valued statistics approach, we get the contribution of each supplier to quality and the result is listed below in Table 3. Based on the estimation of experts, supplier A is the best choice when the quality criterion is considered only.

Table 3. Three suppliers' contribution to quality

	Supplier A	Supplier B	Supplier C
Contribution	0.92	0.68	0.81

**Scenario 2: Multi-points decision model.** A more complex scenario is that the procurer not just checks the overall rank of a concerned parameter, but also controls multiple other operational level parameters. In this case, it is the procurer's responsibility to operationalize the standard. In this case, there are either close cooperation or weak trust between the procurer and supplier. The procurer could also check every factor that it believe have impact on product quality. As shown in Fig. 5, annotated  $i^*$  models with related data can help us position the decision point in the right context. For multi-point decision model, the verdict is made based on the sum of each related factor that supplier feel relevant to the quality of products.

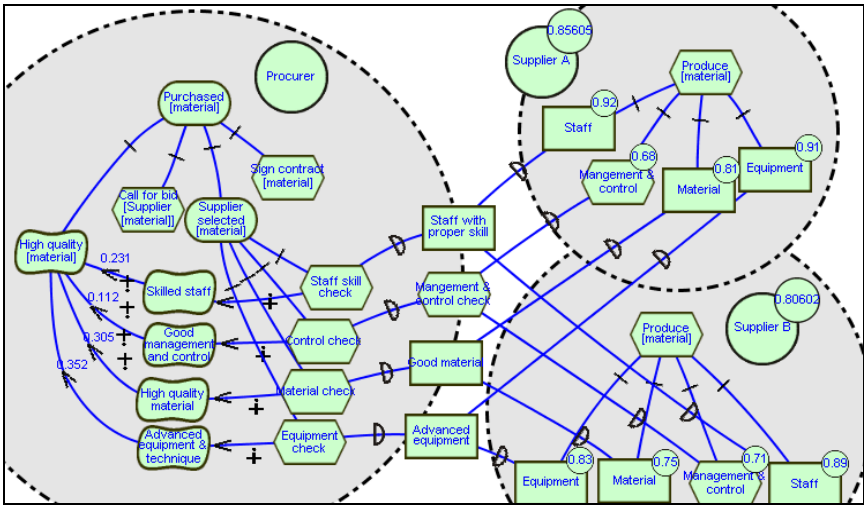


Fig. 5. Multi-points decision model in annotated  $i^*$

To evaluate the performance of each supplier, two techniques mentioned are used. AHP can help decide the weight of each sub-softgoal, and weighted set-valued statistics approach is used to decide the contribution of each task result through experts' evaluation. And then we can get the overall score about achievement of single softgoal. In our example, assuming related results of three suppliers are shown as in Table 4.

The final score  $q$  of supplier is calculated by the following formula:

$$q_i = \sum_{i=1}^4 w_i r_i$$

Table 4. Weights and three suppliers' contributions to sub-softgoals

Sub-softgoal	1-staff	2-Management and control	3-material	4-equipment	$q$
Weight(w)	0.231	0.112	0.305	0.352	
Result(r)-A	0.92	0.68	0.81	0.91	0.85605
Result(r)-B	0.89	0.71	0.75	0.83	0.80602
Result(r)-C	0.75	0.77	0.71	0.88	0.78580

Then we can choose supplier A as the best based on the calculation results.

The two cases about quality criterion, not only explain these two particular situations about procurement services selection, but also give general suggestions on how to evaluate one criterion. It depends on how much information regarding the supplier one has, and how much knowhow knowledge is available to the procurer. Different annotated  $i^*$  models can be developed to help analyze the problem.

### 3.3 Multi-criteria Decision Making

After every criterion of each service instance is evaluated, the multi-objective decision making becomes straightforward. AHP method has given the weight of contribution of

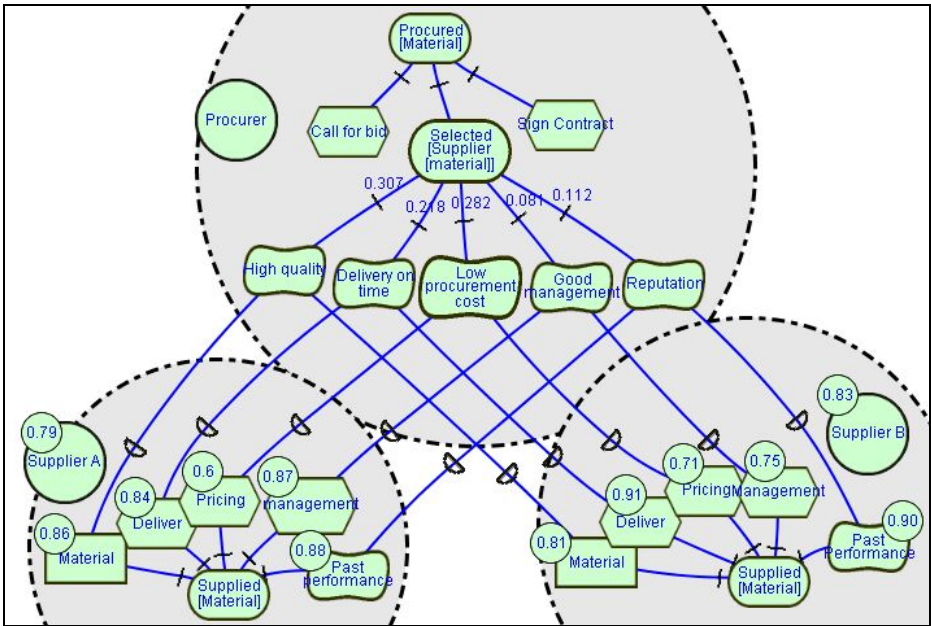


Fig. 6. High leveled multi-criteria decision making model

each criterion (Softgoal Clusters). High level multi-criteria decision making can be modeled as in Fig. 6. The notation of the link from softgoal to goal is its weight, which is an extension of  $i^*$  framework.

Each instance can receive a final total score given single criteria the weight of each criterion. And we can choose the one having highest score as the result of decision. In our example, assuming related results of three suppliers are shown as in Table 5.

The final score  $M$  of supplier is calculated by the following formula:

$$M_i = \sum_{i=1}^5 w_i r_i$$

Table 5. Weights and three suppliers' contributions to the softgoals

softgoal	1- quality	2-delivery	3- cost	4- management	5- reputation	$M$
Weight(w)	0.307	0.218	0.282	0.081	0.112	
Result(r)-A	0.85605	0.84085	0.60012	0.87414	0.87741	0.78842
Result(r)-B	0.80602	0.91121	0.71487	0.74885	0.90445	0.82500
Result(r)-C	0.78580	0.87414	0.67445	0.91004	0.76640	0.78154

Then instance B is the best choice, recommended by our evaluation framework.

For readability of models, we can show models at different abstract levels during different stage of the analysis process. For example, if there are many criteria, detailed models for each criterion should be given independently. The multi-objective decision should show the objects at highest abstract level. On the other hand, the multi-objective model can be made with single criteria models in detail within one model file.



### 3.4 Sensitivity Analysis of the Decision Result

Human factors are deeply involved in our model driven decision method, from stakeholders’ evaluation of the importance of criteria to experts’ evaluation of the candidates’ contribution. Sometimes, a small change of people’s attitude will significantly affect the final decision. So sensitivity analysis is necessary to help people understand the potential threats to validity.

Sensitivity analysis can be conducted in two situations. One is criteria weights sensitivity analysis which analyzes how the changed weights of two criteria affects the final result, and the other is satisfaction sensitivity analysis which analyzes how the changed satisfaction evaluation affects the decision result. In our example, if we make sensitivity analysis of supplier A and B, we can get the results in Table 6 and 7.

Data in Table 6 means how much the weight change could impact the final score of the two suppliers. For example, the data “+0.2095” in the column quality and row delivery means supplier A and B will have same score if the weight of quality adds 0.2095 (with the corresponding loss of delivery). “(Data)” means we cannot reach the balance of score by changing weights of the two factors.

Data in Table 7 means how much the satisfaction change of A will result in the two suppliers having the same score. for example, “+0.119” means if the quality satisfaction of A adds 0.119, the score of A will be the same with B. “(Data)” has the same meaning with that in Table 6.

**Table 6.** Criteria weights sensitivity analysis of supplier A and B

softgoal	quality	delivery	cost	management	reputation
quality		-0.2948	-0.1530	(+0.3351)	(-0.3272)
delivery	+0.2095		(-0.5681)	+0.1289	(+0.5822)
cost	+0.1530	(+0.5681)		+0.1051	(+0.2875)
management	(-0.3351)	-0.1289	-0.1051		(-0.1656)
reputation	(+0.3272)	(-0.5822)	(-0.2875)	(+0.1656)	

**Table 7.** Satisfaction sensitivity analysis of supplier A to B

Softgoal	quality	delivery	cost	management	reputation
Satisfaction Chang	+0.119	(+0.168)	+0.130	(+0.452)	(+0.327)

Sensibility analysis is needed when some candidates have similar scores. It will help examine whether adjustment is needed before getting the final decision.

## 4 Related Work and Discussion

The services paradigm presents a promising direction for enterprises to compose complex services applications from existing individual services on demand. It is important for enterprise to select their service partners dynamically to form strong and optimal alliances. Thus, services selection is an interesting research area that has attracted many researchers’ attentions.

Earlier work such as [7] inherited ideas from autonomous agents match making in AI, in which the selection of services is based on semantic matching of the service functionalities rather than service qualities. Later, [6] proposes a services selection framework and ontology to support the systematic treatment of services selection based on quality of services, in which an upper ontology of QoS and a middle ontology specifying domain independent quality concepts are given. There is also research on services selection algorithms based on multiple QoS constraints [5, 13]. [15] proposes a Bayesian approach helps make decisions about adoption or rejection of a alternative from uncertain information. As a complementary, our work in this paper gears towards defining a methodology for developing domain-specific ontology for services selection.

As a natural extension to our earlier work on goal and scenario-based modeling approach for information systems [4], this paper proposes using goal and agent-based preference models, represented with annotated NFR/*i*\* modeling framework to facilitate services selection decision making activities. In particular, we present how we enhance the modeling language with quantitative preference information based on input from domain experts and end users, how softgoals interrelationships graph can be used to group impact factors with common focus, and how actor dependency models can be used to represent and evaluate alternative services decisions. The proposed approach is illustrated with running example scenarios of provider selection for logistic services. We adopt the characteristic of NFR to find the correlations of factors and find out main factor categories which affects the decision making, and use annotated *i*\* model to fulfill the decision making process.

In the future, the weighted set-valued statistics approach in this paper can be further enhanced with other quantification measures. Their efficacy in supporting services selection can be studied and evaluated. Another possible future line of research is to develop domain specific services selection knowledge base and integrate with widely used service execution platform.

## Acknowledgement

Financial support from the National Natural Science Foundation of China (Grant No.60873064), the National Basic Research and Development 973 Program (Grant No.2009CB320706), the National 863 High-tech Project of China (Grant No.2007AA01Z122) and the Key Project of National Natural Science Foundation of China (Grant no. 90818026) are gratefully acknowledged.

## References

1. Chung, L., Nixon, B.A., Yu, E., Mylopoulos, J.: Non-Functional Requirements in Software Engineering. Kluwer Academic Publishers, Dordrecht (2000)
2. Dickson, G.: An Analysis of Vendor Selection Systems and Decisions. *Journal of Purchasing* 1966(2)
3. Liu, L., Liu, Q., Chi, C.-h., Jin, Z., Yu, E.: Towards service requirements modelling ontology based on agent knowledge and intentions. *International Journal of Agent-Oriented Software Engineering* 2(3), 324–349, DOI:10.1504/IJAOSE.2008.019422

4. Liu, L., Yu, E.: Designing Information Systems in Social Context: A Goal and Scenario Modelling Approach. *Information Systems* 29(2), 187–203 (2004)
5. Liu, Y., Ngu, A., Zeng, L.: QoS computation and policing in dynamic web service selection. In: Proceedings of the 13th International World Wide Web Conference, New York, USA, pp. 66–73 (2004)
6. Maximilien, E.M., Singh, M.P.: A framework and ontology for dynamic Web services selection. *Internet Computing* 8(5), 84–93 (2004)
7. Sycara, K.P., Widoff, S., Klusch, M., Lu, J.: Larks: Dynamic Matchmaking among Heterogeneous Software Agents in Cyberspace. *Autonomous Agents and Multi-Agent Systems* 5(2), 173–203 (2002)
8. Weber, C.A., Current, J.R., Benton, W.C.: Vendor selection criteria and methods. *European Journal of Operational Research* 50 (1991)
9. Xiang, J., Qiao, W., Xiong, Z., Jiang, T., Liu, L.: SAFARY: A semantic web service implementation platform. In: Proceedings of APSEC-SOPOSE 2006, Bangalore, India, December 9 (2006)
10. Xie, H., Liu, L., Yang, J.: i\*-Prefer: Optimizing Requirements Elicitation Based on Actor Preferences. Accepted by proceedings of ACM Software Applications Conference (to appear) (March 2009)
11. Yu, E.: Agent Orientation as a Modeling Paradigm. *Wirtschaftsinformatik* 43(2), 123–132 (2001)
12. Yu, E.: Towards Modeling and Reasoning Support for Early-Phase Requirements Engineering. In: Proceedings of the 3rd IEEE International Symposium on Requirements Engineering (RE 1997), Washington D.C., USA, January 6–8, 1997, pp. 226–235 (1997)
13. Yu, T., Lin, K.J.: Service selection algorithms for Web services with end-to-end QoS constraints. *Information Systems and E-Business Management* (2005)
14. Yahya, S., Kingsman, B.: Vendor rating for an entrepreneur development programme: a case study using the analytic hierarchy process method. *Journal of Operational Research Society* 50(9), 916–930 (1999)
15. Zhang, H., Jarzabek, S.: A Bayesian Network Approach to Rational Architectural Design. *International Journal of Software Engineering and Knowledge Engineering* 15(4), 695–717 (2005)

# Secure Information Systems Engineering: Experiences and Lessons Learned from Two Health Care Projects

Haralambos Mouratidis<sup>1</sup>, Ali Sunyaev<sup>2</sup>, and Jan Jurjens<sup>3</sup>

<sup>1</sup> School of Computing and Technology, University of East London, England  
haris@uel.ac.uk

<sup>2</sup> Institut für Informatik, Technische Universität München, Germany  
sunyaev@in.tum.de

<sup>3</sup> Computing Department, The Open University, Great Britain  
j.jurjens@open.ac.uk

**Abstract.** In CAiSE 2006, we had presented a framework to support development of secure information systems. The framework was based on the integration of two security-aware approaches, the Secure Tropos methodology, which provides an approach for security requirements elicitation, and the UMLsec approach, which allows one to include the security requirements into design models and offers tools for security analysis. In this paper we reflect on the usage of this framework and we report our experiences of applying it to two different industrial case studies from the health care domain. However, due to lack of space we only describe in this paper one of the case studies. Our findings demonstrate that the support of the framework for the consideration of security issues from the early stages and throughout the development process can result in a substantial improvement in the security of the analysed systems.

## 1 Introduction

Current information systems contain a large number of important and sensitive information that needs to be protected. Therefore, the need to secure these systems is recognised by academics and practitioners alike. This is reflected in the current literature where it is now widely accepted [13] [5] that security should be embedded into the overall information systems development and not added as an afterthought. As a result, a number of researchers are working towards the development of modelling languages and methodologies that can support the consideration of security as part of the information systems development process, and various approaches coming from different schools of thought have been reported in the literature (see for example [13]). Along these lines, a number of model-based security engineering approaches have been proposed [8][1][2]. In such approaches, a model of the system is initially constructed and a corresponding implementation is derived from that model either automatically or manually. An important limitation of these approaches is the lack of consideration of the earlier stages of the development process, such as early requirements analysis. To overcome this issue, in previous work, which was presented in CAiSE 2006 [14], we integrated the UMLsec approach [8] with the secure Tropos methodology [11]. The resulting framework allows the construction of an initial security requirements model that is constantly refined until a well defined model of the

system has been constructed. In particular, the framework defines a set of guidelines and transformation steps to enable developers to “translate”, in a structured manner, the initial high level security requirements models, defined in Secure Tropos, to a well defined design model defined in UMLsec. Our framework is different from other works [16][1][2][6][10] trying to integrate security considerations into the development lifecycle. Existing work is mainly focused either on the technical or the social aspect of considering security. Moreover, approaches are usually applicable only to certain development stages. In contrast our approach considers security as a two dimensional problem, where the technical dimension depends on the social dimension. Moreover, our approach is applicable to stages from the early requirements to implementation. The next two sections describe the application of the framework to the two industrial case studies.

In this paper we report on the application of our framework to an industrial case study for the development of a Telematics system at a German hospital. We then reflect on the applicability of this framework and our experiences from its applications to two industrial case studies from the health care domain, the described German Hospital Telematics case study and the Single Assessment Process of the English National Health Service (NHS) case study. The paper is structured as follows. Section 2 provides a summary of the main elements of the framework to assist readers not familiar with it. Section 3 discusses the case study and it demonstrates how our framework was applied and how the security of the Telematics system was improved. Section 4 reflects on the application of the framework. Our reflection is mainly sub-divided into three main areas: *Framework Development*, *Lessons Learned*, and *Improvements*. Section 5 concludes the paper.

## 2 Secure Tropos meets UMLsec: A Model-Based Security Aware Framework

As mentioned above, the framework, under discussion in this paper, has been presented in CAiSE 2006 [14]. Therefore, the aim of this section is not to repeat the details of the framework but rather to summarise it, in order to enable the readers of the paper to understand the following sections. The security-aware process of the framework includes four main stages: *Security Analysis of System Environment*, *Security Analysis of System*, *Secure System Design*, and *Secure Components Definition*. In each of these stages a number of models are defined that are then refined in the later stages. In particular, the main aim of the first stage is to understand the social dimension of security by considering the social issues of the system’s environment, which might affect its security. In doing so, the environment in which the system will be operational is analysed with respect to security. In particular, in line with the Secure Tropos methodology, the stakeholders of the system along with their strategic goals are analysed in terms of actors who have strategic goals and dependencies for achieving some of those goals. Then the security needs of those actors are analysed in terms of security-related constraints that are imposed to those actors. Such analysis results into the Secure Tropos security-enhanced model. Then, for each of the actors depicted on the Secure Tropos security-enhanced model, security goals and entities are identified, in order to satisfy the imposed security constraints. This information is modelled

with the aid of the Secure Tropos security-enhanced goal model. During the second stage, the technical dimension of security is analysed by employing modelling and reasoning activities similar to the ones used in the previous stage, but now the focus is on the system rather than its environment. The output of this stage is refined Secure Tropos security-enhanced actor and goal models. During the third stage, the aim is to define the architecture of the system with respect to its security requirements. To achieve this, a combination of Secure Tropos and UMLsec models are employed. The Secure Tropos security-enhanced actor and goal models are furthered refined and provide input to the Secure Tropos architectural style model, which defines the general architecture and the components of the system. The Secure Tropos models are then transformed to UMLsec Class and Deployment diagrams, which are used to model the security protocols and properties of the architecture. To support the transformation of the Secure Tropos to UMLsec models, the framework defines a set of guidelines and steps [14]. In particular, two main transformation guidelines have been defined along with eight steps that describe each of the guidelines in detail. During the fourth stage, the components of the system are identified in detail. To achieve this, UMLsec activity diagrams are used to define explicitly the security of the components and UMLsec sequence diagrams or state-chart diagrams are used to model the secure interactions of the system's components. For example, to determine if cryptographic session keys, exchanged in a key-exchange protocol, remain confidential in view of possible adversaries, UMLsec state-chart diagrams can be used to specify the security issues on the resulting sequences of states and the interaction with the component's environment. Moreover, the constraints associated with UMLsec stereotypes are checked mechanically, based on XMI output of the models and using sophisticated analysis engines such as model-checkers and automated theorem provers. The results of the analysis are given back to the developer, together with a modified model, where the weaknesses that were found are highlighted [9].

### 3 Case Study

The case study is based on experience during a project with healthcare professionals of the University Hospital in Munich, Germany. Some of the authors have long standing project relationships with this establishment and that relationship was the starting point of the project. The project involved around 13 people, including the hospital's head of the computer centre, a number of physicians, a data protection officer, and a number of computer scientists including some of the authors. Gradually every insured patient in Germany is to receive a new smart-card based patient card, which will replace the past insurance cards. This new electronic patient card will be able to carry administrative functions as well as control access to the health data of the patient. As such, the electronic patient card is the central part of a Telematics infrastructure which can provide access to multiple forms of information and can store data locally. Besides the storage of data on the electronic patient card, other applications are possible. These applications include: drug order documentation, electronic physician letters, treatment cost receipts, emergency case data, general patient data, and an electronic health record. In accordance with the new German health reform, the next phase after the introduction of the electronic patient card will be the electronic patient document. For this reason, the goal is to realize a uniform Telematics platform as a

communication turntable for all parties, involved in the health care industry. Many different aspects must be considered during the development and implementation of such a health care Telematics infrastructure. Due to ethical, judicial, and social implications, medical information requires extremely sensitive handling. Guaranteeing the protection of the patient-related information and the health care information-systems is becoming increasingly important. On the other hand, there is an acceptance problem on the part of the end users (patients, care providers, cost units). Data collection and requirements elicitation took place through analysis of existing specifications (that are confidential and cannot further discuss) and a number of interviews with the stakeholders. Our interviews with a number of health care professionals [17] revealed that the main problems were deficient communication between medical practices and hospitals and bad scheduling in hospitals. All interviewees identified an existent demand for IT support in health care networks. But at the same time they expressed some worries about the security level and dependability of using information systems.

Following the steps of our framework, it is important to understand the environment of the system and reason about the security constraints imposed by that environment to the various system stakeholders. To keep the analysis of the case study in a manageable length, for this paper, we focus our analysis on three main stakeholders: the *Patient*, the *Hospital* and the *Physician*. Security constraints related to the distribution of medical information are imposed by the environment (such as German health data protection laws) and also by the *Patient*. As mentioned above, a secure Tropos security-enhanced actor diagram is used to initially model this information, which is later refined by adding the system-to-be, as another actor who has dependencies with the existing actors. This model is shown in Figure 1. As shown in that

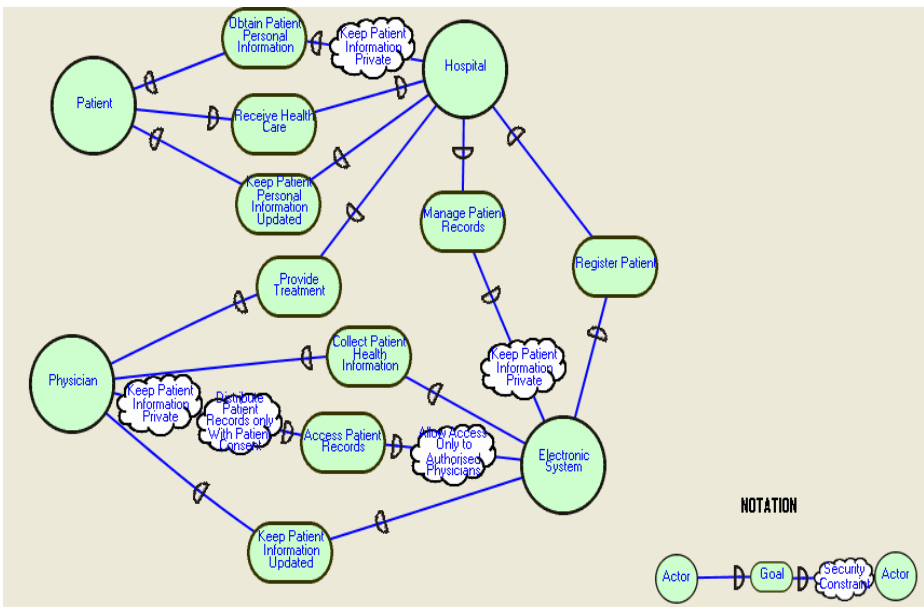


Fig. 1. secure Tropos security enhanced actor diagram

figure, the *Physician* depends on the *Electronic System* to access patient records. However, there are a number of security constraints imposed both to the *Physician* and to the *Electronic System* for that dependency to be valid.

The Secure Tropos security-enhanced actor diagram is further refined by analysing the internal goals of the *Electronic System*. This analysis results in the Secure Tropos security-enhanced goal diagram that models the various internal goals, tasks and security constraints of the system. In particular, our analysis indicates that for the system to satisfy its security constraints, various secure goals are introduced such as *Ensure System Privacy*, *Ensure Data Integrity*, *Ensure Data Availability*, *Ensure Secure Transfer of Records*. These abstract goals have been analysed further and appropriate secure tasks have been identified such as *Encrypt Data*, *Check Digital Signatures*, *Perform Auditing*, *Transfer Data through Virtual Private Network*, *Enforce Access Control* and so on.

When all the secure goals and secure tasks of the system have been identified, the main aim is the identification of a suitable architecture. The core idea of the physician-hospital architecture that was considered in this application is based on a separation of the central database into two independent, and stand-alone partial databases, whose linking returns the inquired answer, just as is the case with a central database. For the user of the system, the procedure remains transparent. Every kind of electronic communication between the medical practices and the hospitals is fundamentally based on one central storage and processing place: the core database. This core database contains and processes all organizational, administrative, and medical information about the patient. The idea of this architecture is to split this core database into two separate databases: first, the so-called "Metadatabase", and secondly, the "Hospital Information System database" ("HIS-database"). The "Metadatabase" contains all administrative data of the patient (name, first name, date of birth, address, insurance data etc.).

The "HIS-database" contains all medical data (like diagnostic images, data, pictures, treatment, medicines etc.) of the respective patient. This sensitive health information does not have a reference to the individual person; it is stored pseudonymously. Additionally, these two "records" possess an attribute named "ID". With its assistance, the combination of the two suitable entities (the administrative data of a patient and his/her health information) can be realized. The two databases are kept physically separate from each other. They are completely autonomous, i.e. there is no direct connection between the two databases. The access is gained through an encrypted connection and is possible to only one of the two databases at any given point in time.

Following the steps and transformation rules of the framework (see [14]) UMLsec deployment diagrams are constructed from the Secure Tropos models to represent the architecture defined in our analysis. When the components of the system have been defined, the next step involves the verification of the security of the modelled architecture. For this purpose, UMLsec sequence diagrams are employed and security properties, identified as important during the analysis of the system (modelled in Secure Tropos models), such as integrity, secrecy, authenticity are used to evaluate the architecture and to indicate possible vulnerabilities.

For example, consider figure 2 that illustrates the sequence diagram of the transmission of data between the user (e.g. Doctor) and the databases. It allows its secret



information to be read using the operation *getMetaData()*, whose return value is also secret. That specification violates the security information flow requirement, since partial information about the time input from the higher sensitivity level operation *getMetaData()* is leaked out via the return value of the lower sensitivity level operation *getHISData()*.

However, our analysis indicated that in order to avoid such violation, the system’s architecture should include a wrapper with a function of placing artificial inquiries to the databases. Artificial inquiries are constantly placed against the system in a way that does not simply place them sequentially after each other; instead, they overlap, at best several times over the entire time. Through this variation, it is impossible for the attacker to filter and/or further pursue individual inquiries. Thus, the problem of the possible time inquiries on the part of the attacker would be solved. The attacker is not able to plumb individual-queries and has thus no possibility thereby to extract the numerical data (time stamps).

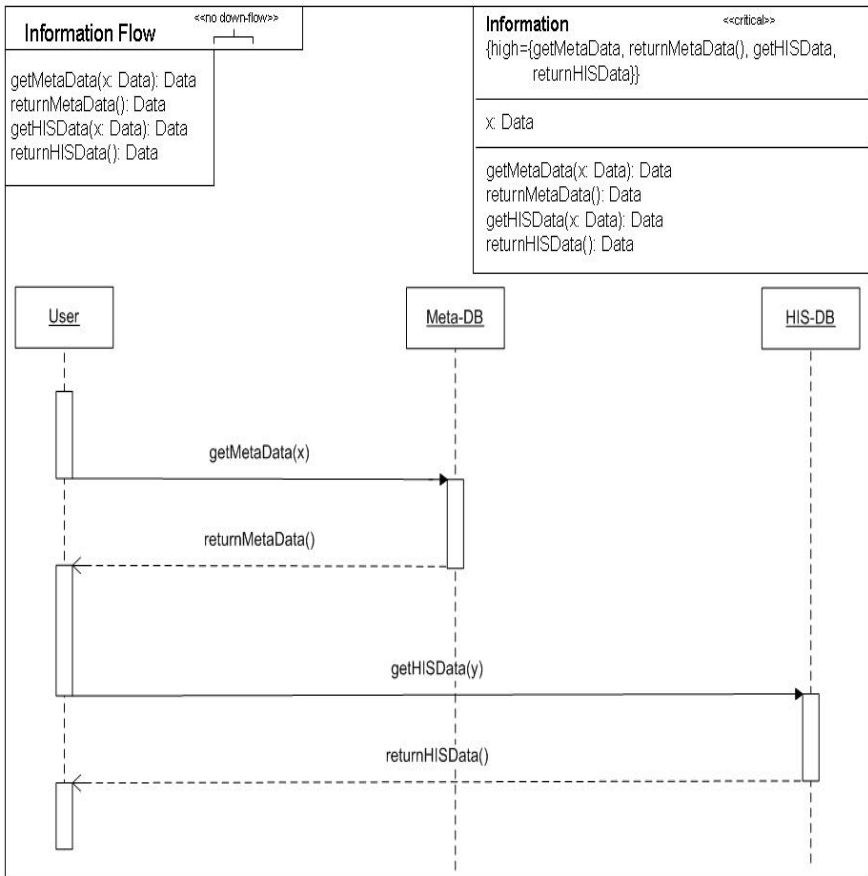


Fig. 2. Sequence Diagram illustrating transmission of data

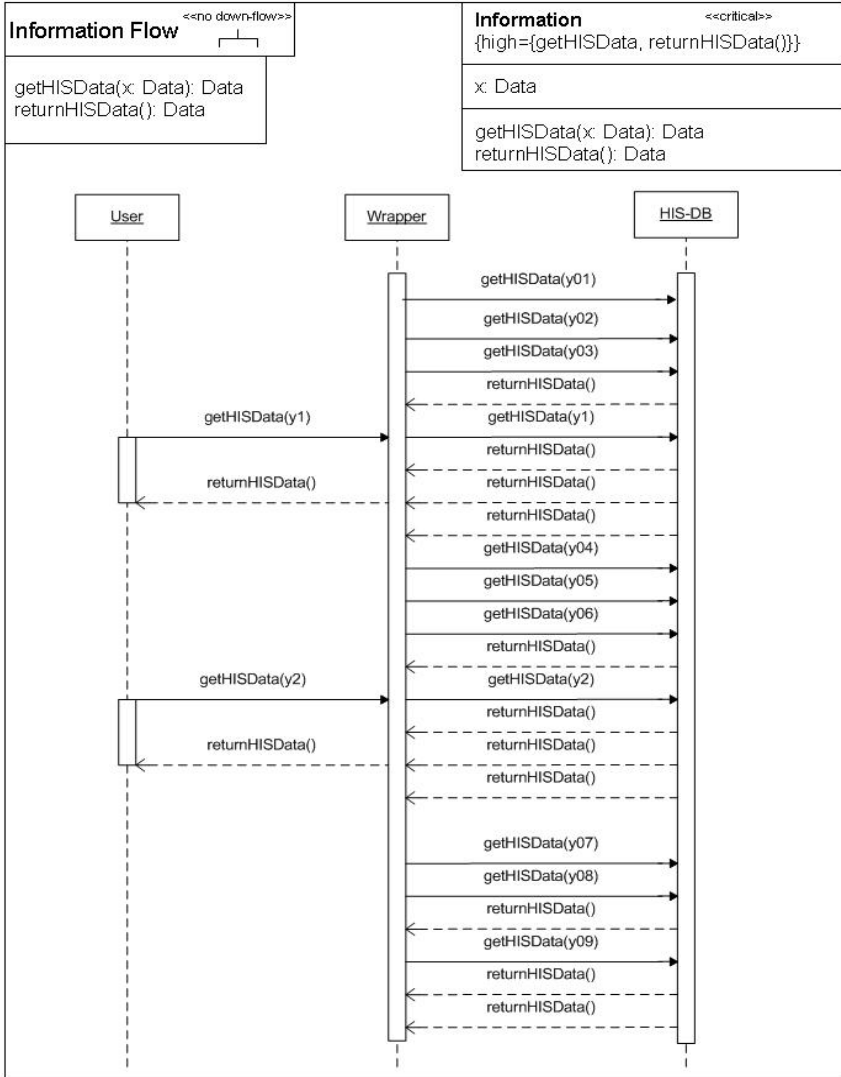


Fig. 3. Refined Sequence Diagram

The refined sequence diagram is shown in Figure 3. If a correct query is posed to the system, the wrapper simply continues to lead the query. If this is not the case, the wrapper generates more own queries. Each time, the wrapper examines whether a query is present, in order to be able to then act accordingly.

The wrapper constantly sends artificial inquiries to the HIS-Database, with each clock pulse the query is passed on. If a "correct" query against the system has been placed, the wrapper simply passes on the query. One does not know on which clock pulse the query falls. The allocation of a date to the appropriate operation sequence is

thus not realizable. The attacker cannot find out which response belongs to which request, and accordingly, which queries from both databases belong together. With this discreteness, a more complex distribution for clock pulses is possible as well. With regards to security verification, there is only one way for the attacker to gain knowledge of the right assignment between the two databases (i.e. Meta-Database and HIS-Database).

The attacker could succeed if he can extract which tuple of information has been queried at which time. Then, the attacker could use this information to find out about the linking of the information between the two databases.

Here, the concern is about an indirect loss of information since following our analysis the system encrypts and securely keep the information. The date of the query procedure indirectly reveals partial knowledge of the confidential information. The attacker can possibly assume the information which was queried after each other and/or time near briefly by the two databases could belong together. The assumption is based on the fact that in the normal case, after the client requests something from the server, the server responds relatively timely. It is the same with the two databases, the HIS-Database and the Meta-Database. First, the user requests something from the Meta-Database and then simultaneously (so the physician does not have to wait) the HIS-Database is queried. If the attacker wants to extract the combination of these two databases, he just has to wait from the Meta-Database-Request until the HIS-Database-Request and the subsequent answers and would then probably be able to extract the correct combination. In the above sequence diagram, the secret information is allowed to be read using the operation *getMetaData()*, whose return value is also secret. The data object is supposed to be prevented from indirectly leaking out any partial information about *high* via *non-high* data, as specified by the stereotype *<<no down-flow>>*. It is important that the observable information on the time of query allows no conclusions about the information that are being requested. Therefore, by applying the UMLsec tool suite [9], one can now make sure that the proposed design in fact fulfils its security requirements. More importantly by following the transformation rules and guidance of the framework, we are able to track specific security solutions (mechanisms) to specific security requirements.

The solution above developed improves on a number of security problems that current telematics platforms, such as HealthBase, TempoBy, MeDaCom, IHE, Intermediation platform, RITHME, PICNIC, and NHSnet, suffer. A large number of telematics platform exist. For our research, we have empirically compared a number of them, against the system developed by employing our framework. Most of the telematics concepts use the same security standards and techniques. The assurance of data security and data integrity is based on the electronic communication with the following six points: (1) There is only one central storage and processing place - the database in the clinic/hospital; (2) A special software is installed and implemented at all attached and entitled points of the network entrance; (3) There is a smart-card reader at each entitled point of network entrance; (4) The connection is based on a virtual private network. There is a VPN router and the required clients for it; (5) Hospitals as well as the medical practices communicate through special firewalls which have been devised for this kind of electronic communication; (6) The medical and administrative personal data is transmitted in encrypted form. Having just one, generally used, central database could be a possible security weakness open to exploitation,

since all stored patient information is concentrated in a single location. Even though there are several security mechanisms in place for the protection of the records database as well as for the connection to and from the database; there are no further protection mechanisms for the data in the case of capture and decryption by an attacker. The eavesdropping or interception of the transmitted data could happen via interception of the transmitted packages as well as a direct hardware infiltration of the database connection within the hospital. An administrator, or a person who is responsible for the setup and maintenance of the database system, could be an attacker. The attacker could intercept the transmitted packages, save them locally in his hard drive and decrypt them without time- or place-restrictions. In the above mentioned Telematics platforms, the central database contains complete data for each patient. This means it contains purely administrative data as well as medical information about the patient. This is the reason, why the protection of the sensitive information and its access restriction/non-readability to attackers cannot be presupposed anymore. However, these issues are all dealt with by the architecture described in the previous section.

## 4 Reflection

In this section we reflect on the framework based on its application to the case study described above, as well as a second case study again from the health care domain. However, due to lack of space we cannot explicitly describe the application of our framework to the case study. The project involved health and social care professionals, such as General Practitioners and Nurses, specialist health professionals, such as Social Workers, and health care IT professionals. Its main aim was to analyse and specify the requirements of a software system to deliver the Single Assessment Process (SAP) [12], a health and social care needs assessment process for older people in England, with particular emphasis on its security. Two important conclusions were drawn. First of all, it became obvious from the discussions with various social care professionals and patients, that privacy was the number one security attribute required by this system. Secondly, the project identified the lack of security-aware methodologies that could assist developers in analysing the electronic Single Assessment Process (eSAP) system with security in mind.

So the following sections discuss our reflections from both these case studies. Our discussion is mainly sub-divided into three main areas: *Framework Development*, which discusses challenges faced during the development of the framework, along with the solutions provided at that point, and it reflects on whether the given solutions are satisfactory based on the application of the methodology to the case study; *Lessons Learned*, which discusses the lessons learned from applying the methodology to the case study; and *Improvements*, which provides an insight about how the methodology can be improved for the application in large industrial context.

### 4.1 Framework Development

In this sub-section we reflect on the issues and challenges we faced during the development of our framework and we discuss, by reflecting on the application of the framework to the case studies, whether the decisions taken at the framework's development stages were successful.

**Challenge 1: Integration.** A major challenge on the development of the framework was the seamless transition from the secure Tropos models to the UMLsec models.

**Solution:** To achieve the above challenge, we decided to employ a functional integration [15], where individual approaches' models stay intact and guidelines to translate the models from one approach to another and indicate the inputs and the outputs of these models are defined. A number of guidelines and steps were also defined to support the integration [14].

**Reflection:** The initial guidelines and the steps defined assisted in the translation of the models from secure Tropos to UMLsec. However, during the development of the Telematics system we identified some inconsistencies between the secure Tropos and the UMLsec models. By investigating this issue, we found out that the problem existed due to some errors in some of the guidelines. In particular, initially our guidelines suggested that actor related resources of the secure Tropos security-enhanced actor models should be translated in a 1-to-1 analogy to attributes of the UMLsec class models. However, by applying the framework to the Telematics case study it became obvious that this was not the case, and in most of the cases, a resource will result in more than one attributes. This was mainly because the secure Tropos security-enhanced actor models contain analysis information whereas the class UMLsec models contain design information. Another issue that was raised during the application of such guidelines was the possible need to formalise them using a transformation language. This clearly constitutes area of future work.

**Challenge 2: Process.** An important issue of our work was the development of a process to support the development of framework.

**Solution:** We decided to base the development process on the Secure Tropos development process. In particular, the development process enables the construction of an early requirements model that is furthered refined, following a top-down approach, to a security model that is amenable to formal verification with the aid of automatic tools [9]. The refinement process is governed by a set of rules and activities [14]. It is worth mentioning that the process is highly iterative.

**Reflection:** The application of our framework to the case study indicated some problems. In particular, initially all the information from the early and late requirement models was effectively refined to the design models. However, during the application of the framework to the case study, it became apparent that this should not be the case. This was mainly because the analysis models contain reasoning information which should not be transformed to the design models. For instance, analysis models can contain information on different alternatives for satisfying a particular security constraint. At the initial development of the framework, we would transform all these alternatives to the design models. However, by applying the framework to the case study and when trying to transform all the reasoning information to design, we were faced with a large number of design goal conflicts. Currently, such conflicts need to be overcome manually but we envisage that in the future some automatic support can be provided.

## 4.2 Lessons Learned

In this sub-section we discuss with the aid of a number of questions all the lessons learned from the application of the methodology.

### **How easy is the framework to learn?**

The described approach results from the integration of two existing security-aware approaches. As such, a number of software engineers are familiar with their concepts and notation. Especially, the UMLsec approach effectively uses UML concepts and notation and therefore it is easy to understand by developers familiar with UML. The Secure Tropos approach on the other hand, is based on the  $i^*$  [19]/Tropos [3] notation and concepts that although not as popular as UML, it is well known in the requirements engineering area. Therefore, although an initial effort is required to understand the framework, we expect that developers familiar with UML and/or Tropos will be able to grasp the concepts and notations of the methodology easily.

### **Did you come across any unexpected obstacles during the application of the framework to any of the case studies?**

The application of the framework to the case studies did not yield any unexpected obstacles. As discussed above, there were some inconsistencies between the models due to some errors on the guidelines, but we were expecting something like this, since it was the first time the framework was applied to real-life health care case studies. On the other hand, once these inconsistencies were solved, the framework worked as expected and we were able to analyse the environment of the system in terms of its security and transform this analysis to a design, which we could verify.

### **Was the framework modified to enable its application to the case studies?**

The framework was not modified to fit the case studies, but the application of the framework to the case studies resulted in a number of modifications as discussed in the previous section.

### **How the framework helps the analysis and design of the system with respect to security?**

The application of the framework to the case studies revealed that it helps the analysis and design of the system with respect to security in various ways:

- (i) Developers are able to consider security both as a social aspect as well as a technical aspect. It is widely known that security is most often compromised not by breaking dedicated security mechanisms but by exploiting vulnerabilities in their usage. Therefore, as argued widely in the literature, it is not enough just to consider security mechanisms and protocols, but an understanding of the human factor and the environment of the software system is also required. By considering both the social aspect and the technical aspect of security, our framework allows developers to obtain a clear understanding of any potential security vulnerabilities that might arise from the interplay of the two security aspects and therefore minimize, leading to the development of secure software systems.

- (ii) The framework allows the definition of security requirements at different levels and as a result it provides better integration with the modelling of the system's functionality.
- (iii) Security is not considered in isolation but simultaneously with the rest of the system requirements. Such treatment of security helps to minimize the number of conflicts between security and other requirements. Such conflicts are usually the reason for security vulnerabilities, therefore by minimizing these conflicts, the security vulnerabilities of the system are also minimized.
- (iv) The framework allows the consideration of the organisational environment for the modelling of security issues, by facilitating the understanding of the security needs in terms of the real security needs of the stakeholders, and then it allows the transformation of the security requirements to a design that is amenable to formal verification with the aid of automatic tools. This introduces a well structured approach to model-based security engineering.

### **Was the framework appropriate for the health care domain case studies?**

In general, the framework was appropriate for the two studies from the health care domain. The health care sector is quite complex and security issues are affected not only by related technologies but also from the human factor. The framework allowed us, in collaboration with the health care professionals involved, to analyse both these security dimensions by (i) analysing the security issues imposed to the system by its environment (various stakeholders); (ii) reasoning about different possible solutions that satisfy the system's security requirements.

### **What useful conclusions did you derive for model-based security engineering?**

It is worth mentioning that the case studies were not set up to assess the usefulness of model-based security engineering and/or compare it with other security and non-security engineering approaches. However, some useful conclusions were drawn. First of all, it became obvious that, as with all the security-aware approaches, some extra effort and knowledge is required due to the security aspect. In particular, basic knowledge is needed about security terminology and theory and extra effort is required by the developers to analyse and model the security concerns. Therefore, we expect that model-based security engineering will be an attractive option to developers looking to develop security-critical systems rather than a general option for any software system development. Secondly, the production of models that integrate security concerns, as opposed to the production of models without security concerns, allows developers to (i) reason in a conclusive way and by taking into account simultaneously the general requirements of the system together with the security requirements and therefore identify any conflicts; (ii) develop a extensive and precise security-aware documentation, something that it is required by common security standards, such as the Common Criteria [4].

## **4.3 Improvements**

In this section we discuss how the framework can be improved to enable its application in large industrial context.

### **Tool Support**

Currently the framework is supported by different types of tools corresponding to the two approaches integrated (i.e. Secure Tropos and UMLsec). The transformation from the secure Tropos models to the UMLsec models takes place manually (the Secure Tropos tool produces details of the models developed in XML, which can be used to feed information on the UMLsec tool). Although, this does not prevent the application of the framework to large industrial projects, it is a concern in terms of efficiency and time. The development of a tool to support the transformation of the models will substantially reduce the time that is required to transform the models and therefore increase the applicability of the framework to that type of projects. Such tool will also support the analysis and resolution of design goal conflicts.

### **Documentation**

Currently, the only documentation about the framework is a set of research papers and internal reports describing some of its aspects, as well as documentation that describes the two original approaches, i.e. secure Tropos and UMLsec. It is important, however, to produce a complete documentation that will explain the original approaches, the advantages of the integration, the transformation steps, the models and the new development process. Such documentation will help to understand the framework and to make it accessible to a larger number of developers. In other words, it will help to improve the usability of the framework.

### **Model related improvements**

There are two main issues related to the improvement of the models. The first is related to the version of the UML in which the UMLsec definition is based. Currently, UMLsec is based on UML1.5. The subsequent release of UML 2.0 raises the question to what extent the UMLsec approach is dependent on a particular version of UML, or whether it can be used flexibly with different UML versions (including UML 2.0). This would be a very interesting question to explore. The second issue is related to the linkage between the models and the implementation (code). In particular, automatic generation of text-sequences from the models to provide assurance that the code correctly implements the models and thus satisfies the security requirements of the system would be desirable.

## **5 Conclusions**

This paper presented an experience report from the application of a model based security engineering framework to two case studies from the health and social care domain (German Hospital Telematics and electronic Single Assessment Process). Apart from the reflections described above, our experience of employing such framework to the health care case studies has indicated two important issues related to each of the individual methodological components of the framework. Secure Tropos concepts are more intuitive and comprehensible than for instance Object Oriented concepts for



people without information systems engineering background, such as the majority of health care professionals. As such, this provides an advantage during the requirements elicitation stage for health care case studies. Using UMLsec, the extension of the Unified Modelling Language (UML) for secure systems development and the concept of model-based security requirements, security requirements are handled as an integrated part of the development and derived from enterprise information such as security policies, business goals, law and regulation as well as project specific security demands. These are then updated and refined iteratively and finally refined to security requirements at a technical level, which can be expressed using UMLsec, and analyzed mechanically using the tool-support for UMLsec by referring to a precise semantics of the used fragment of UML. This allows one to validate design against security requirements early in the development cycle.

## References

1. Alam, M., Hafner, M., Breu, R.: Constraint based role based access control in the SEC-TET-framework A model-driven approach. *Journal of Computer Security* 16(2), 223–260 (2008)
2. Basin, D., Doser, J., Lodderstedt, T.: Model Driven Security for Process Oriented Systems. In: *Proceedings of the 8th ACM symposium on Access Control Models and Technologies*, Como, Italy (2003)
3. Bresciani, P., Giorgini, P., Giunchiglia, F., Mylopoulos, J., Perini, A.: TROPOS: An Agent Oriented Software Development Methodology. *Journal of Autonomous Agents and Multi-Agent Systems* 8(3), 203–236 (2004)
4. Common Criteria, <http://www.commoncriteriaportal.org/>
5. Devanbu, P., Stubblebine, S.: Software Engineering for Security: a Roadmap. In: *Proceedings of ICSE 2000 (track on The future of Software engineering)* (2000)
6. Hermann, G., Pernul, G.: Viewing business-process security from different perspectives. *International Journal of electronic Commerce* 3, 89–103 (1999)
7. Jennings, N.R.: An agent-based approach for building complex software systems. *Communications of the ACM* 44(4) (April 2001)
8. Jürjens, J.: *Secure Systems Development with UML*. Springer, Heidelberg (2004)
9. Jürjens, J., Shabalin, P.: Tools for Secure Systems Development with UML. In: *FASE 2004/05 special issue of the International Journal on Software Tools for Technology Transfer*. Springer, Heidelberg (2007)
10. McDermott, J., Fox, C.: Using Abuse Case Models for Security Requirements Analysis. In: *Proceedings of the 15th Annual Computer Security Applications Conference* (December 1999)
11. Mouratidis, H., Giorgini, P., Manson, G.: Modelling Secure Multiagent Systems. In: *The Proceedings of the 2nd International Joint Conference on Autonomous Agents and Multi-agent Systems*, Melbourne, Australia, pp. 859–866. ACM, New York (2003)
12. Mouratidis, H., Philp, I., Manson, G.: A Novel Agent-Based System to Support the Single Assessment Process of Older People. *Journal of Health Informatics* 9(3), 149–162 (2003)
13. Mouratidis, H., Giorgini, P.: *Integrating Security and Software Engineering: Advances and Future Visions*. Idea Group Publishing (2006)

14. Mouratidis, H., Jürjens, J., Fox, J.: Towards a Comprehensive Framework for Secure Systems Development. In: Dubois, E., Pohl, K. (eds.) CAiSE 2006. LNCS, vol. 4001, pp. 48–62. Springer, Heidelberg (2006)
15. Muhanna, W.: An Object-Oriented Framework for Model Management and DSS Development. *Decision Support Systems* 9(2), 217–229 (1993)
16. Sindre, G., Opdahl, A.L.: Eliciting security requirements with misuse cases. *Requir. Eng.* 10(1), 34–44 (2005)
17. Sunyaev, A.: *Telematik im Gesundheitswesen - Sicherheitsaspekte*, tech. rep., TU Munich (2006)
18. Wooldridge, M., Ciancarini, P.: Agent-Oriented Software Engineering: The State of the Art. In: Ciancarini, P., Wooldridge, M. (eds.) AOSE 2000. LNCS, vol. 1957, pp. 1–28. Springer, Heidelberg (2001)
19. Yu, E.: *Modelling Strategic Relationships for Process Reengineering*, Ph.D. Thesis. Dept. of Computer Science, University of Toronto (1995)

# An Architecture for Requirements-Driven Self-reconfiguration

Fabiano Dalpiaz, Paolo Giorgini, and John Mylopoulos

University of Trento - DISI, 38100, Povo, Trento, Italy  
{fabiano.dalpiaz, paolo.giorgini, jm}@disi.unitn.it

**Abstract.** Self-reconfiguration is the capability of a system to autonomously switch from one configuration to a better one in response to failure or context change. There is growing demand for software systems able to self-reconfigure, and specifically systems that can fulfill their requirements in dynamic environments. We propose a conceptual architecture that provides systems with self-reconfiguration capabilities, enacting a model-based adaptation process based on requirements models. We describe the logical view on our architecture for self-reconfiguration, then we detail the main mechanisms to monitor for and diagnose failures. We present a case study where a self-reconfiguring system assists a patient perform daily tasks, such as getting breakfast, within her home. The challenge for the system is to fulfill its mission regardless of the context, also to compensate for failures caused by patient inaction or other omissions in the environment of the system.

## 1 Introduction

There is growing demand for software systems that can fulfill their requirements in very different operational environments and are able to cope with change and evolution. This calls for a novel paradigm for software design where monitoring, diagnosis and compensation functions are integral components of system architecture. These functions can be exploited at runtime to monitor for and diagnose failure or under-performance, also to compensate through re-configuration to an alternative behavior that can better cope with the situation on-hand. Self-reconfiguration then is an essential functionality for software systems of the future in that it enables them to evolve and adapt to open, dynamic environments so that they can continue to fulfill their intended purpose.

Traditionally, self-reconfiguration mechanisms are embedded in applications and their analysis and reuse are hard. An alternative approach is externalized adaptation [1], where system models are used at runtime by an external component to detect and address problems in the system. This approach – also known as model-based adaptation – consists of monitoring the running software, analyzing the gathered data against the system models, selecting and applying repair strategies in response to violations.

We analyze here the usage of a special type of system models: requirements models. Deviations of system behavior from requirements specifications have been discussed in [2], where the authors suggest an architecture (and a development process) to reconcile requirements with system behavior. Reconciliation is enacted by anticipating deviations

at specification time and solving unpredicted circumstances at runtime. The underlying model is based on the goal-driven requirements engineering approach KAOS [3].

In this paper, we propose a conceptual architecture that, on the basis of requirements models, adds self-reconfiguration capabilities to a system. The architecture is structured as a set of interacting components connected through a Monitor-Diagnose-Compensate (MDC) cycle. Its main application area is systems composed of several interacting systems, such as Socio-Technical Systems [4] (STSs) and Ambient Intelligence (AmI) scenarios. We have chosen to use Tropos [5] goal models as a basis for expressing requirements, for they suit well for modeling social dependencies between stakeholders. We enrich Tropos models adding activation events to trigger goals, context-dependent goal decompositions, fine-grained modeling of tasks by means of timed activity diagrams, time limits within which the system should commit to carry out goals, and domain assumptions that need to be monitored regardless of current goals.

We adopt the BDI paradigm [6] to define how the system is expected to reason and act. The system is running correctly if its behavior is compliant with the BDI model: when a goal is activated, the system commits to it by selecting a plan to achieve it. The architecture we propose monitors system execution and looks for alternatives when detects no progress or inconsistent behaviour.

The closest approach to our work is Wang et al. [7], which proposes a goal-oriented approach for self-reconfiguration. Our architecture differs from hers in the details of the model we use to monitor for failures and violations. These details allow us to support a wider class of failures and changes, also to compensate for them.

This paper is structured as follows: Section 2 presents the baseline of our approach, Section 3 describes our proposed architecture for self-reconfiguration, whereas Section 4 explains how to use it. Section 5 details the main monitoring and diagnosis mechanisms the architecture components use. Section 6 shows how the architecture can be applied to a case study concerning smart-homes. Section 7 presents related work and compares our approach to it. Finally, Section 8 discusses the approach and draws conclusions.

## 2 Baseline: Requirements Models

A requirements-driven architecture for model-based self-reconfiguration needs a set of models to support full modeling of requirements. A well established framework in Requirements Engineering (RE) is goal-oriented modeling [3], where software requirements are modelled as goals the system should achieve (with assistance from external agents). Among existing frameworks for requirements models, we have chosen Tropos [5], for it allows to describe systems made up of several socially interacting actors depending on each other for the fulfillment of their own goals. Recently, Jurata et al. [8] have revisited the so-called “requirements problem” – what it means to successfully complete RE – showing the need for requirements modeling frameworks richer than existing ones. The core ontology they propose is based on the concepts of goal, softgoal, quality constraint, plan, and domain assumption. Direct consequence of this result is that goal models alone are insufficient to completely express system

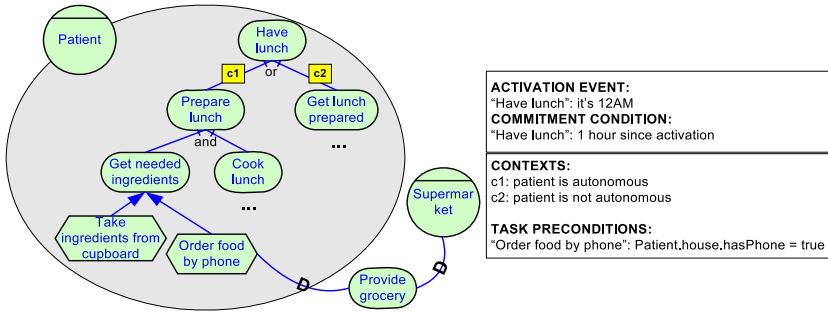


Fig. 1. Enriched Tropos goal model used by our architecture

requirements, and in our framework we support some of the suggested ingredients to express requirements.

We adopt an enriched version of Tropos, which contains additional information to make it suitable for runtime usage: (i) activation events define when goals are triggered; (ii) commitment conditions express a time limit within which an agent should commit to its goal; (iii) contexts express when certain alternatives are applicable (like in Ali et al. [9]); (iv) preconditions define tasks applicability. In Fig. 1, two agents (*Patient* and *Supermarket*) interact by means of a dependency for goal *Provide Grocery*. The top-level goal of patient – *Have lunch* – is activated when it's 12AM, and the patient should commit to its achievement within one hour since activation. Two alternatives are available to achieve the goal, that is *Prepare lunch* and *Get lunch prepared*. In this scenario, the former option is applicable only in context *c1*, that is when patient is autonomous, whereas the latter option is applicable when the patient is not autonomous (*c2*). Goal *Prepare lunch* is and-decomposed to sub-goals *Get needed ingredients* and *Cook lunch*. The former goal is a leaf-level one, and there are two tasks that are alternative means to achieve it (means-end): *Take ingredients from cupboard* and *Order food by phone*. The latter task requires a dependency for goal *Provide grocery* on agent *supermarket*.

A shared language to express information about domain is clearly needed. This language is used to formally express contexts, preconditions, domain assumptions, and any relation between domain and requirements. We exploit an object diagram (as in [9]), where context entities are objects, their properties are attributes, and relations between entities are association links. For instance, the precondition for task *Order food by phone* ( $Patient.house.hasPhone = true$ ) can be expressed in an object model with classes *Patient* and *House*, where *Patient* is linked to *House* by an aggregation called *house*, and *House* has a boolean attribute *hasPhone*. Domain assumptions are rules that should hold regardless of current goals. For example, a domain assumption for our small example is that each patient has exactly one house. Finally, we use a fine grained definition of tasks, in which each task is a workflow of monitorable activities to be carried out within time constraints. The completion of each activity is associated to the detection of an associated event, which is expressed over the context model. We provide further details about this formalism (timed activity diagram) in Section 5.

### 3 System Architecture

In this section we propose our conceptual architecture for structuring systems able to self-reconfigure. We present the architecture logical view in Fig. 2, exploiting an UML 2.0 component diagram to show the components and the connections among them. Component diagrams depict not only the structure of a system, but also the data flow between components (through provided and required interfaces).

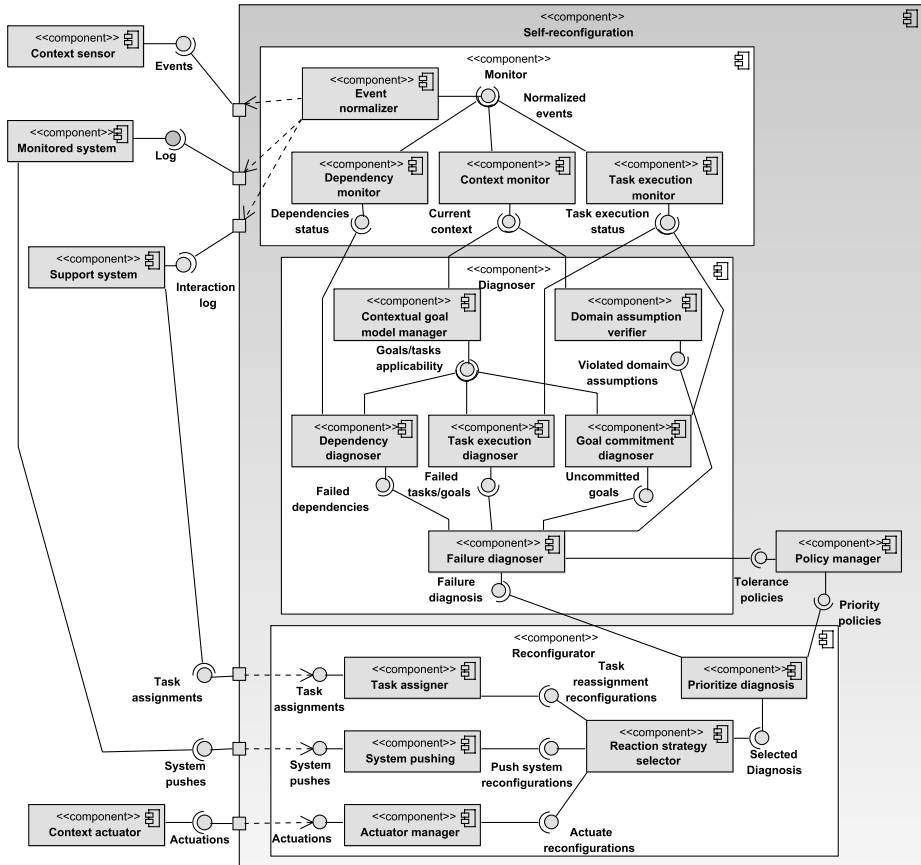


Fig. 2. Logical view on the proposed architecture for self-reconfiguration

#### 3.1 External Components

Our architecture supports systems characterized by decentralized and non-monolithic structure, such as Socio-Technical Systems (STSs) and Ambient Intelligence (AmI) scenarios, which require quick and effective reconfiguration in response to context change or failure. A set of external components interacts with the self-reconfiguration component, providing inputs and enacting reconfigurations.

The component *Context sensor* represents any system providing up-to-date information about the context where the system is running. In AmI settings, sensors are spread throughout the environment and collect data such as temperature, light level, noise, presence. Also desktop applications have several context sensors that provide useful values such as free memory, CPU utilization, mainboard temperature, and list of active processes. The component context sensor provides changes in the context through the interface *Events*. Possible events are changes in the light level, detection of humans in front of the door, identification of loud noise in the bathroom.

*Monitored system* is the system the self-reconfiguration component assists, that is the stakeholder whose requirements are monitored to diagnose and compensate failures. This system need not necessarily be software or hardware, but can be – and often is – a human or an organization. Examples of monitored systems are anti-virus software, patients living in smart-homes, firemen in crisis management settings. This component provides all available information concerning the current status of the system through the interface *Log*, and requires from the interface *System pushes* advice on what should be done (which *goals*) and how it should act (which *tasks*). A patient can be reminded to take her medicine by sending an SMS to her mobile phone (system pushes interface).

*Support system* represents any system connected to the monitored system by requirements level links by goal, task, or resource dependencies from the monitored system. For example, anti-virus software may depend on update sites for the resource “updated virus definition file”, while patients may depend on social workers for the goal “prepare breakfast”. The provided interface *Interaction log* contains information about the status of dependencies with the monitored system; the required interface *Task assignments* provides the tasks or goals for which the monitored system depends on support systems. If the patient should prepare breakfast but did not commit to it, the self-reconfiguration component can order breakfast from a catering service (the support system), enacting a dependency from the patient to the catering service for goal “prepare breakfast”.

*Context actuator* identifies any actuator in the environment which can receive commands to act on the context. Examples of actuators in AmI scenarios are sirens, door openers, automated windows, and remote light switches. The component gets from the required interface *Actuations* the commands to enact.

### 3.2 Self-reconfiguration Component

The self-reconfiguration capabilities of our architecture are provided by the component *self-reconfiguration*. We identified three major sub-components in the reconfiguration process, each enacting a phase in a Monitor-Diagnose-Compensate cycle. *Monitor* is in charge of collecting, filtering, and normalizing events and logs; *Diagnoser* identifies failures and discovers root causes; *Reconfigurator* selects, plans and deploys compensation actions in response to failures.

The monitoring phase starts with the *Event normalizer* gathering the current status of the monitored system, of the context, and of the interaction with support systems. These events are normalized according to a shared context model, e.g. defining transformation schemes using XSLT [10]. The event normalizer provides the translated data through the interface *Normalized events*. This interface is required by three different components, each handling a specific type of events. *Dependency monitor* computes

the status of existing dependencies and exposes it through the provided *Dependencies status* interface. *Context sensor* is in charge of updating the interface *Current context*, processing the normalized events related to changes in the context. For instance, if the house door is closed (*door.status = closed*) and we just received an event such as *open(door, time<sub>i</sub>)*, the status of the door will change to open (*door.status = open*). The component *Task execution monitor* handles events concerning the execution of tasks and provides the interface *Task execution status*. For example, if the patient is executing the task “Open door” and event *pressed(patient, button, time<sub>j</sub>)* is received, the status of task “Open door” will turn to success.

The diagnosis phase – responsibility of the component *Diagnoser* – is essentially a verification of the current status against requirements models. Models specify what should happen and hold: which goals should / can / cannot be achieved, which tasks can / cannot be executed, the domain assumptions that should not be violated. The richer the requirements models are, the more accurate the diagnosis will be. In contrast, the granularity of detected events is bounded by technological and feasibility concerns, and also increases the overhead introduced by the architecture. Detecting if a patient is sitting on a sofa is reasonably realizable (e.g., using pressure sensors), while detecting if she is handling a knife the wrong way is far more complex.

*Contextual goal model manager* analyzes the goal model to identify goals and tasks that should / can / cannot be achieved, and provides this output through the interface *Goals / Tasks applicability*. The component *Domain assumption verifier* checks a list of domain assumptions against the current context, and exposes identified violations through the provided interface *Violated domain assumptions*.

*Dependency diagnoser* computes problems in established dependencies. Dependencies fail not only if the dependee cannot achieve the goal or perform the task (e.g., the nurse cannot support the patient because she’s busy with another patient), but also when changes in the context modify goal applicability and the dependency is not possible anymore (e.g., the patient exits her house and thus cannot depend on a catering service anymore). *Task execution diagnoser* is needed to verify whether the current execution status of tasks is compliant with task applicability. For example, if the patient is preparing breakfast but already had breakfast, something is going wrong and this failure should be diagnosed. This component provides the interface *Failed tasks / goals*. *Goal commitment diagnoser* is in charge of detecting those goals that should be achieved but for whose fulfillment no action has been taken. In our framework, each top-level goal has a commitment time, a timeout within which a commitment to achieve the goal should be taken (i.e., an adequate task should begin). For instance, the patient should have breakfast within two hours since waking up. This component provides the interface *Uncommitted goals*.

The component *Failure diagnoser* requires the interfaces containing the identified failures and *Tolerance policies* provided by component *Policy manager*. The policy manager – handling policies set by system administrators – specifies when failures do not lead to reconfiguration actions. For example, lack of commitment for washing dishes can be tolerated if the patient’s vital signs are good (she may wash dishes after next meal). Diagnoses to be compensated are exposed through *Failure diagnosis*.



The reconfiguration phase – carried out by component *Reconfigurator* – should define compensation / reconfiguration strategies in response to any kind of failure. Its effectiveness depends on several factors: number of tasks that can be automated, available compensation strategies, extent to which the monitor system accepts suggestions and reminders. In our architecture we propose general mechanisms, but the actual success of compensation strategies is scenario-dependent and difficult to assess in a general way. Suppose a patient suddenly feels bad: if she lives in a smart-home provided with a door opener, the door can be automatically opened to the rescue team; otherwise, the rescue team should wait for somebody to bring the door keys.

The component *Prioritize diagnosis* selects a subset of failures according to their priority level and provides them through the interface *Selected Diagnosis*. Common criteria to define priority are failure severity, urgency of taking a countermeasure, time passed since failure diagnosis. Selected diagnoses are then taken as input by the component *Reaction strategy selector*, which is in charge of choosing a reaction to compensate the failure. This component acts as a planner: given a failure, it looks for appropriate reconfigurations, and selects one of them. Three different types of reconfigurations are supported by our architecture, each manifested in a specific interface. *Task reassignment reconfigurations* contains reconfigurations that involve the automated enactment dependencies on support systems. For example, if the patient didn't have breakfast and the commitment time for the goal is expired, the system could automatically call the catering service. *Push system reconfigurations* includes strategies that push the monitored system to achieve its goals (reminding goals or suggesting tasks). A push strategy for the patient that forgot to have breakfast is sending an SMS to her mobile phone. *Actuate reconfigurations* consists of compensations that will be enacted by context actuators. For instance, if the patient feels bad, the door can be automatically opened by activating the door opener. Three components use the interfaces provided by reaction strategy selector: *Task assigner*, *System pushing*, and *Actuator manager*. Their role is to enact the reconfigurations that have been selected, and each component provides a specific interface.

## 4 Creating the Architecture for an Existing System

We describe how the architecture can be used in practice to add self-reconfiguration capabilities to an existing distributed socio-technical system. The required input is a set of interacting sub-systems – sensors and effectors – that compose the distributed system. The following steps should be carried out: (i) define context model (ii) define requirements models; (iii) establish traceability links for monitoring; (iv) select tolerance policies for diagnosis; and (v) choose reconfiguration and compensation mechanisms.

Steps (i) and (ii) output the models we presented in Section 2, that is the context model, Tropos goal models, timed activity diagrams for tasks, and domain assumptions. Step (iii) defines what to monitor for at runtime, by connecting requirements to code. Traceability is ensured by associating events – produced by sensors – to *activities* that are part of a task, to *task preconditions*, to *contexts*, and to *activation conditions* for top-level goals. Events should also be normalized according to the context model defined in step (i).

Step (iv) is carried out to specify tolerance policies for failures. Indeed, some failures have to be addressed through reconfiguration, whereas some others can be tolerated. In step (v) the reaction mechanisms enacting self-reconfiguration are defined. Two sub-steps should be carried out: (i) definition of a *compensation plan* to revert the effects of the failed strategies, and (ii) identification of a *reconfiguration strategy* to retry goal achievement. Both steps exploit the actuation capabilities of the distributed system, i.e. reconfigurations consist of giving commands to effectors (execute a task, enact a dependency, issue a reminder).

## 5 Monitoring and Diagnosis Mechanisms

We detail now monitoring and diagnosis mechanisms included in our architecture. Efficient and sound algorithms need to be defined for successfully diagnosing problems in the running system. Failures are identified by comparing *monitored behavior* of the system to *expected and allowed behaviors*. Failures occur when (a) monitored behavior is not allowed or (b) expected behavior has not occurred.

Table 1 defines expected and allowed goals and tasks. We use first-order logic rules for clarity, but our prototype implementation is based on disjunctive Datalog [11]. We suppose that each goal instance differs from other instances of the same goal class for

**Table 1.** First-order logic rules to define expected and allowed goals and tasks

$\begin{array}{l} \text{goal}(G) \wedge \text{goal\_parameters}(G, P) \wedge \neg \text{done}(G, P) \wedge \text{activation\_evt}(G, P, T) \\ \wedge T \leq \text{current\_time} \wedge \nexists G_p \text{ s.t.} \\ \quad \text{goal}(G_p) \wedge \text{decomposed}(G_p, G) \end{array}$	(i)
<b>should.do(G,P)</b>	
$\frac{\text{should\_do}(G, P)}{\text{can\_do}(G, P)}$	(ii)
$\begin{array}{l} \text{goal}(G) \wedge \text{goal\_parameters}(G, P) \wedge \neg \text{done}(G, P) \\ \wedge \exists G_p \text{ s.t.} \\ \quad \text{goal}(G_p) \wedge \text{goal\_parameters}(G_p, P_p) \wedge \text{decomposed}(G_p, G, Dec) \\ \quad \wedge \text{can\_do}(G_p, P_p) \wedge \text{context\_cond}(Dec) \\ \quad \wedge \forall p \in P \text{ s.t. } (\exists p_p \in P_p \text{ s.t. } \text{name}(p, n) \wedge \text{name}(p_p, n)), \\ \quad \quad \text{value}(p, v) \wedge \text{value}(p_p, v) \end{array}$	(iii)
<b>can.do(G,P)</b>	
$\begin{array}{l} \text{task}(T) \wedge \text{task\_parameters}(T, P) \wedge \text{pre\_cond}(T, P) \wedge \neg \text{done}(T, P) \\ \wedge \exists G \text{ s.t.} \\ \quad \text{goal}(G) \wedge \text{goal\_parameters}(G_p, P_p) \wedge \text{means\_end}(G, T, Dec) \\ \quad \wedge \text{context\_cond}(Dec) \wedge \text{can\_do}(G, P_p) \\ \quad \wedge \forall p \in P \text{ s.t. } (\exists p_p \in P_p \text{ s.t. } \text{name}(p, n) \wedge \text{name}(p_p, n)), \\ \quad \quad \text{value}(p, v) \wedge \text{value}(p_p, v) \end{array}$	(iv)
<b>can.do(T,P)</b>	

actual parameters; for example, a patient's goal *Have breakfast* can be repeated every day, but with different values for the parameter *day*.

Rule (i) defines when a top-level goal should be achieved. This happens if  $G$  is a goal with parameters set  $P$ , the goal instance has not been achieved so far, the activation event has occurred before the current time, and  $G$  is a top-level goal (there is no other goal  $G_p$  and/or-decomposed to  $G$ ). Rule (ii) is a general axiom saying that whenever a goal instance should be achieved, it is also allowed. Rules (iii) and (iv) define when tasks and decomposed goals are allowed, respectively. A goal instance  $G$  with parameters set  $P$  can be achieved if it has not been done so far and exists an achievable goal  $G_p$  with parameters  $P_p$  that is decomposed to  $G$ , the context condition on the decomposition is true, and the actual parameters of  $G$  are compatible with the actual parameters of  $G_p$ . A similar condition holds for means-end tasks, with two main differences: tasks are also characterized by a precondition – which should hold to make the task executable – and are connected to goals through means-end (rather than by and/or decomposition).

Expected and allowed goals and tasks identified by rules (i-iv) are used by Algorithm [1](#) to diagnose goals and tasks failures, comparing the monitored behavior to expected and allowed behaviors. The parameters of COMPUTEFAILURES are the monitored system, the examined goal instance, and the set of failures (initially empty). The algorithm is invoked for each top-level goal of the monitored system and explores the goal tree recursively; all parameters are passed by reference.

Algorithm [1](#) starts by setting the status of goal  $g$  to uncommitted, and the variable *means\_end* to false. Lines 3-10 define the recursive structure of the algorithm. If the goal is and/or decomposed (line 3), the set  $G$  contains all the sub-goals of  $g$  (line 4), and the function COMPUTEFAILURES is recursively called for each sub-goal (lines 5-6). If the status of all the sub-goals is success, also the status of  $g$  is set to success (lines 7-8). If the goal is means-end decomposed (lines 9-10),  $G$  contains the set of tasks that are means to achieve the end  $g$ , and *means\_end* is set to true.

If  $g$  is still uncommitted (line 11) each sub-goal (or means-end decomposed task) is examined (lines 12-39). If  $g$  is and-decomposed (lines 13-23), two sub-cases are possible: (a) if the sub-goal  $g_i$  is not allowed but its status is different from uncommitted, and the status of  $g$  is still uncommitted, the status of  $g$  is set to fail and the cycle is broken, for the worst case – failure – has been detected (lines 14-17); (b) if the status of  $g_i$  is fail (lines 18-23), the status of  $g$  is set to fail in turn, and the cycle is broken (lines 19-21); if  $g_i$  is in progress, the status of  $g$  is set to in\_progress.

If  $g$  is or-decomposed or means-end (lines 24-39), it succeeds if at least one sub-goal (or task) succeeds. If  $g$  is means-end decomposed, the algorithm calls the function MONITORSTATUS, which diagnoses the execution status of a task (lines 24-25). If  $g_i$  is not allowed and its status is different from uncommitted,  $g_i$  is added to the set of failures (line 27), and if  $g$  is not committed its status is set to fail (lines 28-29). If  $g_i$  is allowed or its status is uncommitted (line 30), three sub-cases are possible: (a) if the status of  $g_i$  is success, the status of  $g$  is set to success and the cycle is terminated (lines 31-33); (b) if  $g_i$  is in progress, the status of  $g$  is set to in\_progress and the loop is continued (lines 34-35); (c) if the status of  $g_i$  is fail,  $g_i$  is added to the set of failures, and if  $g$  is still uncommitted its status is set to fail. If  $g$  is a top-level goal that should be achieved, its status is uncommitted, and the commitment condition is true, then the status of  $g$  is set

to fail because no commitment has been taken (lines 40-41). If the status of  $g$  is fail, it is added to the list of failures (lines 42-43).

---

**Algorithm 1.** Identification of goal and task failures.
 

---

```

COMPUTEFAILURES( $s$  : System,  $g$  : Goal,  $F$  : Failure [])
1   $g.status \leftarrow$  uncommitted
2   $means\_end \leftarrow$  false
3  if  $\exists g_1 \in s.goals$  s.t.  $decomposed(g, g_1, dec)$ 
4  then  $G \leftarrow \{g_i \in s.goals$  s.t.  $decomposed(g, g_i, dec)\}$ 
5      for each  $g_i$  in  $G$ 
6      do COMPUTEFAILURES( $s, g_i, F$ )
7      if  $\forall g_i$  in  $G$ ,  $g_i.status = success$ 
8      then  $g.status \leftarrow$  success
9  else  $G \leftarrow \{t \in s.tasks$  s.t.  $means\_end(g, t, dec)\}$ 
10      $means\_end \leftarrow$  true
11 if  $g.status = uncommitted$ 
12 then for each  $g_i$  in  $G$ 
13     do if and  $decomposed(g, g_i, dec)$ 
14         then if  $g_i.can\_do = false$  and  $g_i.status \neq uncommitted$ 
15             and  $g.status = uncommitted$ 
16             then  $g.status \leftarrow$  fail
17             break
18         else switch
19             case  $g_i.status = fail$  :
20                  $g.status \leftarrow$  fail
21                 break
22             case  $g_i.status = in\_progress$  :
23                  $g.status \leftarrow$  in\_progress
24         else if  $means\_end = true$ 
25             then  $g_i.status \leftarrow$  MONITORSTATUS( $g_i, g$ )
26             if  $g_i.can\_do = false$  and  $g_i.status \neq uncommitted$ 
27             then  $F \leftarrow F \cup g_i$ 
28                 if  $g.status = uncommitted$ 
29                 then  $g.status \leftarrow$  fail
30             else switch
31                 case  $g_i.status = success$  :
32                      $g.status \leftarrow$  success
33                     break
34                 case  $g_i.status = in\_progress$  :
35                      $g.status \leftarrow$  in\_progress
36                 case  $g_i.status = fail$  :
37                      $F \leftarrow F \cup g_i$ 
38                     if  $g.status = uncommitted$ 
39                     then  $g.status \leftarrow$  fail
40 if  $g.should\_do = true$  and  $g.status = uncommitted$  and  $g.comm\_cond = true$ 
41 then  $g.status \leftarrow$  fail
42 if  $g.status = fail$ 
43 then  $F \leftarrow F \cup g$ 

```

---

Due to space limitations, the algorithms for diagnosing task failures are only sketched here, but they are fully described and discussed in [12]. We define tasks as workflows of activities each occurring within well-defined time limits (we refer to this formalism using the term “timed activity diagram”). Successful completion of an activity  $a$  is associated to the happening of an event  $e$  ( $\text{happens}(e) \rightarrow \text{success}(a)$ ). Activities can be connected sequentially, in parallel (by fork and join nodes), and conditionally (through branch and merge nodes). A graphical example of this formalism is given in Fig. 4. The allowed branch of a decision point is defined by a branch condition. At any instant, a well-formed model has exactly one allowed branch. `MONITORSTATUS` is invoked by Algorithm 1 to check the status of a particular task; its parameters are a task  $t$  and a goal  $g$  linked through means-end. This algorithm returns task failure if the first activity of the task has happened but beyond the commitment time for goal  $g$ , whereas returns uncommitted if the first activity hasn’t happened so far but the commitment time for  $g$  has not expired. If no failure or uncommitment has been identified, the algorithm retrieves the next node in the timed activity diagram that defines the task, calling the recursive function `CHECKNODE`.

The behavior of Algorithm `CHECKNODE` depends on node type. If the node is an *activity*, we check if the associated event happened within the time limit. If it happened beyond time limit we return failure, if it happened before time limit we recursively check the next node. If the event hasn’t happened so far: (a) if time limit has expired we return failure; (b) if the activity is still within its time limit we return `in_progress`. If a *fork* is found, `CHECKNODE` is recursively called for all the forks. If any fork failed a fail value is returned. If all the forks joined a recursive check is performed on the node that follows the join, with time limit starting from the last joined fork. Otherwise, the algorithm returns `in_progress`. If a *branch* is met, all the branches should be checked. We return failure if any branch that was disallowed happened or an activity in an allowed branch failed. We return `in_progress` if allowed activities occurred within time limits and no disallowed activity happened. If none of these conditions hold, we check the node following the merge construct and return its status.

## 6 Case Study: Smart Homes

We show now a promising application for our architecture, emphasizing how requirements models are used to define and check allowed and expected behaviors, and how the architecture performs the MDC cycle. Our case study concerns smart homes: a patient lives in a smart home, a socio-technical system supporting the patient in everyday activities (such as eating, sleeping, taking medicine, being entertained, visiting doctor). Both smart home and patient are equipped with AmI devices that gather data (e.g., patient’s health status, temperature in the house) and enact compensations (e.g., open the door). The partial goal model in Fig. 3 represents the requirements of the patient; due to space limitations, we present here only the top-level goal “Have breakfast”.

Goal  $g1$  is activated when the patient wakes up (activation event); a commitment to achieve  $g1$  should be taken (either by the patient or by other agents) within two hours since goal activation. Four different contexts characterize the scenario: in  $c1$  the patient is autonomous, in  $c2$  the patient is not autonomous, in  $c3$  the patient is at home, in  $c4$

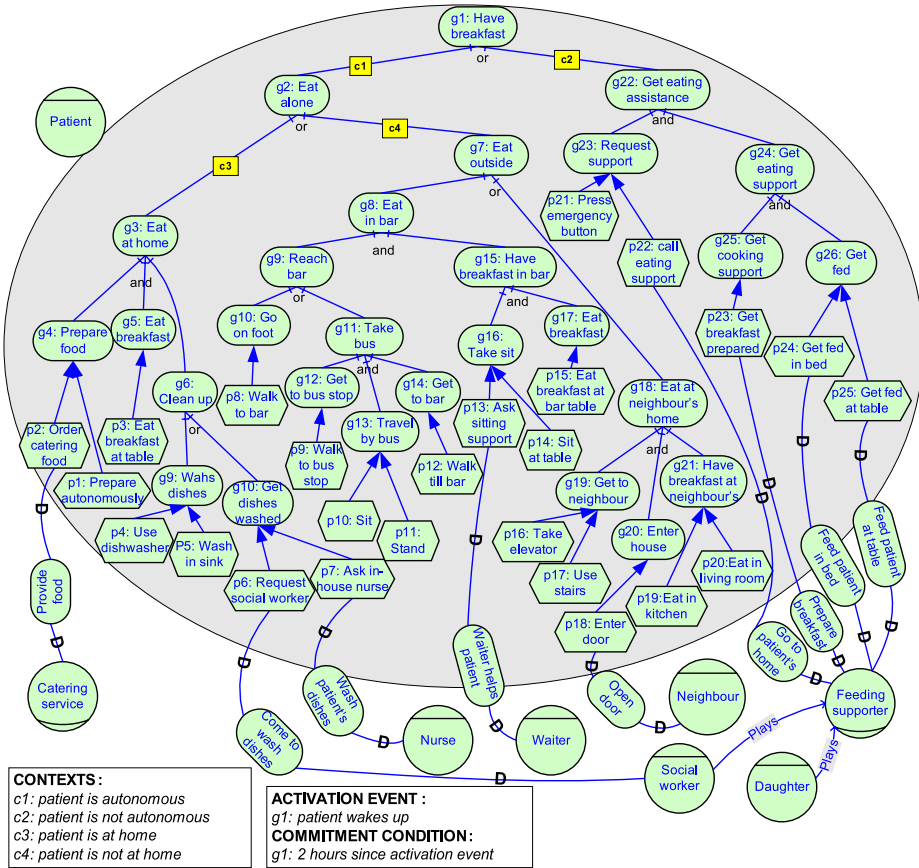
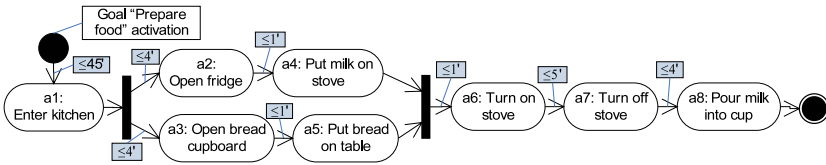


Fig. 3. Contextual goal model describing the patient health care scenario

the patient is not at home. If the patient is autonomous ( $c1$  holds)  $g1$  is decomposed into the subtree of goal “Eat alone” ( $g2$ ); if  $c2$  holds  $g1$  is decomposed into the subtree of goal “Get eating assistance” ( $g22$ ). In the former case,  $c3$  enables the subtree of goal “Eat at home” ( $g3$ ), whereas  $c4$  enables the subtree of goal “Eat outside” ( $g7$ ). When eating at home, the patient has to prepare food ( $g4$ ), eat breakfast ( $g5$ ), and clean up ( $g6$ ). Goal  $g4$  is means-end to two alternative tasks: “Prepare autonomously” ( $p1$ ) and “Order catering food” ( $p2$ ). The latter task requires interaction with actor “Catering service”, which should fulfill goal “Provide food” to execute  $p2$ . The other subtrees of Fig. 3 are structured in a similar way, thus we don’t detail them here.

Our requirements models characterize each task with a precondition that, if false, inhibits the execution of the task. If a task is executed but its precondition is false, a failure occurs (see rule (iv) in Table I). A possible precondition for task “Prepare autonomously” is that there are both bread and milk in the house; a precondition for task “Order catering food” is that the house is provided with a landline phone.

Fig. 4 is a timed activity diagram for task  $p1$ . The task starts as goal “Prepare food” is activated. When the patient enters the kitchen ( $a1$ ), there is evidence that she is going



**Fig. 4.** Timed activity diagram for monitoring the task “Prepare autonomously”

to prepare food. If this doesn’t happen within 45 minutes from goal activation the task fails. After *a1*, a fork node creates two parallel execution processes. In the first fork, the patient should open the fridge (*a2*) and put the milk on stove (*a4*); in the second fork, the bread cupboard should be opened (*a3*) and bread has to be put on the table (*a5*). The forks are then merged, and the next activity is to turn on the stove (*a6*) within a minute since the last completed activity. Sequentially, the task requires the stove to be turned off within 5 minutes (*a7*) and the milk to be poured into the cup (*a8*).

We conclude this section with a description of a possible reconfiguration process. Let’s suppose that patient Mike wakes up at 8.00 am. Mike is autonomous (*c1*) and at home (*c3*); goal *g1* is expected, and the subtree of *g3* is the only allowed one (see rules (i) and (ii) in Table 1). At 8.20 am Mike enters the kitchen: checking the activity diagram for *p1* against this event changes the status of the goal *g4* to *in\_progress*. In turn, this status is propagated bottom-up till *g1* (see Algorithm 1). At 8.25 Mike hasn’t neither opened the fridge nor opened the bread cupboard. This violates the specification of *p1* (Fig. 4). The reconfiguration strategy selector component selects to push the system, and the system pushing component sends an SMS message to remind Mike to have breakfast. The strategy succeeds, for Mike opens both the fridge (*a2*) and the bread cupboard (*a3*), then he puts bread on table (*a5*). These events are compliant with the task specification of Fig. 4, thus the task is evaluated as *in\_progress*. Anyhow, Mike does not put milk on stove (*a4*) within one minute since *a2*, therefore a new failure is diagnosed. The compensation to address this failure is to automate *p2*, and the task assigner component selects a catering service. In an alternative scenario Mike exits house (the context *c4* is true, *c3* is false). This would change the tasks that can happen: the subtree of *g7* becomes the only possible one, and this influences all its sub-goals (rule (iii) in Table 1) and the tasks linked to leaf-level goals (rule (iv) in Table 1).

## 7 Related Work

Self-adaptive software has been introduced by Oreizy et al. [13] as an architectural approach to support systems that modify their behaviour in response to changes in the operating environment. This class of systems performs self-reconfiguration according to the criteria specified at development time, such as under what conditions reconfiguring, open/closed adaptation, degree of autonomy. The building units for self-adaptive software should be components and connectors. Compared to our work, the solution proposed in [13] is generic and flexible to many reconfiguration criteria, whereas ours is focused on particular types of models, that is requirements models.

Rainbow [1] is an architecture-based framework that enables self-adaptation on the basis of (i) an externalized approach and (ii) software architecture models. The authors of Rainbow consider architecture models as the most suitable level to abstract away unnecessary details of the system, and their usage both design- and at run-time promotes the reuse of adaptation mechanisms. Our proposal shares many features with Rainbow; the main difference is that we use higher level models to support the ultimate goal of any software system, that is to meet its requirements. A drawback of our choice is that establishing traceability links between requirements and code is more complex.

Sykes et al. [14] propose a three-layer architecture for self-managed software [15] that combines the notion of goal with software components. This approach is based on a sense-plan-act architecture made up of three layers: *goal management* layer defines system goals, *change management* layer executes plans and assembles a configuration of software components, *component* layer handles reactive control concerns of the components. Our proposal exploits a more elaborate goal representation framework, and differs in planning (predefined plans instead of plan composition) and reconfiguration.

Wang's architecture for self-repairing software [7] uses one goal model as a software requirements model, and exploits SAT solvers to check the current execution log against the model to diagnose task failures. We propose more expressive goal models, accurate specification of tasks based on timed activity diagrams, allow for specifying multiple contexts, and support dependencies on other actors / systems.

Feather et al. [2] address system behaviour deviations from requirements specifications; they introduce an architecture (and a development process) to reconcile requirements with behaviour. This process is enacted by jointly anticipating deviations at specification time and solving unpredicted situations at runtime, and examine the latter option using the requirements monitoring framework FLEA [16]. FLEA is used in conjunction with the goal-driven specification methodology KAOS [3]. Our architecture differs in the usage of different requirements models (Tropos rather than KAOS), support to a wider set of failures ([2] is focused on obstacle analysis), and applicability to scenarios composed of multiple interacting actors (such as AmI ones).

Robinson's ReqMon [17] is a requirements monitoring framework for specific usage in enterprise systems. ReqMon integrates techniques from requirements analysis (KAOS) and software execution monitoring. Although ReqMon's architecture covers all the reconfiguration process, accurate exploration is provided only for the monitoring and analysis phases. Our approach has broader applicability, can diagnose a larger set of failure types, and supports more reconfigurations mechanisms; on the contrary, ReqMon is particularly suitable for enterprise systems.

## 8 Discussion and Conclusion

We have proposed a novel architecture for self-configuring systems founded on principles adopted from Goal-Oriented Requirements Engineering, externalized adaptation, and BDI paradigm. Our approach adds self-reconfiguration capabilities to a wide variety of systems, among which Ambient Intelligence scenarios and Socio-Technical Systems. The architecture is a model-based one, with requirements models used to specify what can, should, and should not happen. We have detailed the main mechanisms for



monitoring and diagnosis, which describe how requirements models are checked against monitored information. We also introduced a case study – smart homes – to show how a realization of the architecture works in practice.

Several aspects will be addressed in future work. Firstly, we need a complete implementation of our architecture, as well as further experimentation on the smart-home case study. We also want to extend our framework so that it deals with a broader class of monitored phenomena, including attacks and failures caused by false domain assumptions. Finally, we propose to introduce mechanisms through which a system can extend its variability space through collaboration with external agents.

## Acknowledgements

This work has been partially funded by EU Commission, through the SERENITY project, and by MIUR, through the MENSA project (PRIN 2006).

## References

1. Garlan, D., Cheng, S.W., Huang, A.C., Schmerl, B., Steenkiste, P.: Rainbow: architecture-based self-adaptation with reusable infrastructure. *Computer* 37(10), 46–54 (2004)
2. Feather, M., Fickas, S., Van Lamsweerde, A., Ponsard, C.: Reconciling system requirements and runtime behavior. In: *IWSSD 1998*, pp. 50–59 (1998)
3. Dardenne, A., Lamsweerde, A., Fickas, S.: Goal-directed requirements acquisition. In: *Science of computer Programming*, pp. 3–50 (1993)
4. Emery, F.: *Characteristics of socio-technical systems*. Tavistock, London (1959)
5. Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F., Mylopoulos, J.: Tropos: An agent-oriented software development methodology. *JAAMAS* 8(3), 203–236 (2004)
6. Rao, A., Georgeff, M.: An abstract architecture for rational agents. In: *KR&R 1992*, pp. 439–449 (1992)
7. Wang, Y., McIlraith, S., Yu, Y., Mylopoulos, J.: An automated approach to monitoring and diagnosing requirements. In: *ASE 2007*, pp. 293–302 (2007)
8. Jureta, I.J., Mylopoulos, J., Faulkner, S.: Revisiting the core ontology and problem in requirements engineering. In: *RE 2008* (2008)
9. Ali, R., Dalpiaz, F., Giorgini, P.: Location-based software modeling and analysis: Tropos-based approach. In: Li, Q., Spaccapietra, S., Yu, E., Olivé, A. (eds.) *ER 2008*. LNCS, vol. 5231, pp. 169–182. Springer, Heidelberg (2008)
10. Clark, J., et al.: Xsl transformations (xslt) version 1.0. *W3C Recommendation* 16(11) (1999)
11. Eiter, T., Gottlob, G., Mannila, H.: Disjunctive datalog. *ACM Transactions on Database Systems* 22(3), 364–418 (1997)
12. Dalpiaz, F., Giorgini, P., Mylopoulos, J.: An architecture for requirements-driven self-reconfiguration. Technical Report DISI-09-010, DISI, University of Trento (2009)
13. Oreizy, P., Medvidovic, N., Taylor, R.N.: Architecture-based runtime software evolution. In: *ICSE 1998*, pp. 177–186 (1998)
14. Sykes, D., Heaven, W., Magee, J., Kramer, J.: From goals to components: a combined approach to self-management. In: *SEAMS 2008*, pp. 1–8. ACM Press, New York (2008)
15. Kramer, J., Magee, J.: Self-managed systems: an architectural challenge. In: *ICSE 2007*, pp. 259–268. IEEE Computer Society, Washington (2007)
16. Fickas, S., Feather, M.S.: Requirements monitoring in dynamic environments. In: *RE 1995*, pp. 140–147. IEEE Computer Society Press, Washington (1995)
17. Robinson, W.N.: A requirements monitoring framework for enterprise systems. *Requirements Engineering* 11(1), 17–41 (2006)

# Automated Context-Aware Service Selection for Collaborative Systems

Hong Qing Yu and Stephan Reiff-Marganiec

University of Leicester, Department of Computer Science, Leicester, UK  
{hqy1, srm13}@le.ac.uk

**Abstract.** Service-Oriented Architecture (SOA) can provide a paradigm for constructing context-aware collaboration systems. Particularly, the promise of inexpensive context-aware collaboration devices and context-awareness for supporting the selection of suitable services at run-time have provoked growing adoption of SOA in collaborative systems. In this paper, we introduce an approach for selecting the most suitable service within a SOA based collaboration system, where suitability depends on the user's context. The approach includes context modelling, generation of context-aware selection criteria and a suitable service selection methodology.

**Keywords:** Context-awareness, SOA, Service Selection, Collaborative Systems.

## 1 Introduction

Collaboration systems are becoming more and more important for facilitating team work and e-activities (e.g. e-business, e-government or e-education). In particular, context-aware and dynamically configured collaboration systems are demanded in order to support collaboration activities, which are progressively flexible due to changes in modern work environments. While context-aware features can not be found in some legacy collaboration systems [Dus04], these systems are static in their architecture and hence the functionality that they offer. With widespread deployment of inexpensive context-aware devices and the innovations brought by the SOA paradigm, we get an opportunity to reconstruct collaboration architectures with context-awareness and dynamic configuration at its centre. For example, the inContext project [1] has defined a platform, called PCSA (Pervasive Collaboration Services Architecture) [RMTC<sup>+</sup>08] to support context-aware collaboration services by working closely with industry users and studying their collaboration needs. The PCSA mainly includes three subsystems:

---

<sup>1</sup> [www.in-context.eu](http://www.in-context.eu)

1. The Access subsystem controls the user and service registration and system access.
2. The service management subsystem, most relevant to this work, is in charge of maintaining the service repository which includes a categorisation of services and details of NFPs (non-functional properties) of the registered services. It also provides functionality to look up services and to obtain service suggestions based on suitability.
3. The Context management subsystem maintains context data of registered users, based on a specified context model.

One of the major challenges of the PCSA is to select collaboration services based on user context information by matching services according to their non-functional properties – clearly the decision is made in the service management subsystem, but is based on the data obtained by the context subsystem.

Before we consider the challenges in more detail and provide an overview of the results of this paper, we consider two motivating examples, which highlight the challenges that modern collaborative systems need to adapt to.

Organising an emergency meeting is a typical e-business and e-government collaboration activity. Notifying all participants to attend the meeting is an important and difficult task because different participants may be in different context situations including different in locations and time zones, they are available on different devices, have diverse contact preferences/rules. For example, a participant may be on holidays in a foreign country and only has a mobile phone with him, or the participant has switched off his mobile phone to save power but is online using IM (Instant Messenger).

In this scenario, the service selection challenge becomes to select the most suitable one. This decision has to be based on dynamically obtain user' context information, with the user being passive in that they cannot be asked upfront which service is most suitable for them.

In contrast to the first scenario, the second case study is concerned with selecting a medical support service during a park fair:

It is expected that a large number of people joins the fair and incidents are expected. Two medical tents are prepared in different locations for providing aid. Efficient collaboration between fair assistants and tents is essential. One tent (Service 1) has more staff and is meant to cope with minor injury cases. The other tent (Service 2) has fewer staff, but more advanced equipment to deal with severe incidents as well as minor injuries. One assistant team scours the park to locate incidents and reports to the most suitable medical support service based on injury level, location of the incident, availability of the tents and response times. For example, if the injury has been reported close to Service 2, but it is not severe then which medical support service should receive the report from the assistant?

Successfully supporting such a collaboration system, requires the service selection method to recognize both users' (member of assistant team) current context and the services' current NFPs.

These two examples highlight a number of challenges to be addressed in context-aware collaboration service selection.

1. The users' runtime context information needs to be dynamically gathered and aggregated in a structured form.
2. The context information needs to be an input to the service selection approach, requiring a link between user context and the relevant non-functional properties of services.
3. The service selection method needs to be automatic.

While services are often considered to be virtual, it turns out that the services that we have to consider do sometimes have physical locations, this is exemplified in the second scenario. In this paper, we are going to illustrate our novel contributions to address these challenges. Specifically we are presenting the following:

1. An OWL/RDF based user context model, with a link connecting to the service NFPs.
2. A method for dynamically generating context-aware service selection criteria based on the service category.
3. A TLE (Type-based LSP Extension) service selection method using the context-aware criteria.

The remainder of this paper is organised as follows. In section 2, we present the context modelling techniques and the details of the model. In section 3, we discuss the connection of the context model to the service's non-functional requirements. In section 4, we explain the TLE method, thus providing a solution to the service ranking issue. Implementation and evaluation results are presented in section 5. We then discuss some current related work and finally draw conclusions.

## 2 User Context Modelling

To allow for selecting the most suitable service for a user, the user's context needs to be evaluated, which is only realistically feasible if it is well defined and organized. In addition, the context information might be distributed and must be easily retrievable. Based on these requirements, we use OWL [OWL] to model user context information and RDF [Gro04] to store context data. By analysing the motivating scenarios and context information in general, our context model has been divided into 4 packages. A simplified top level OWL context model is shown in Figure 1.

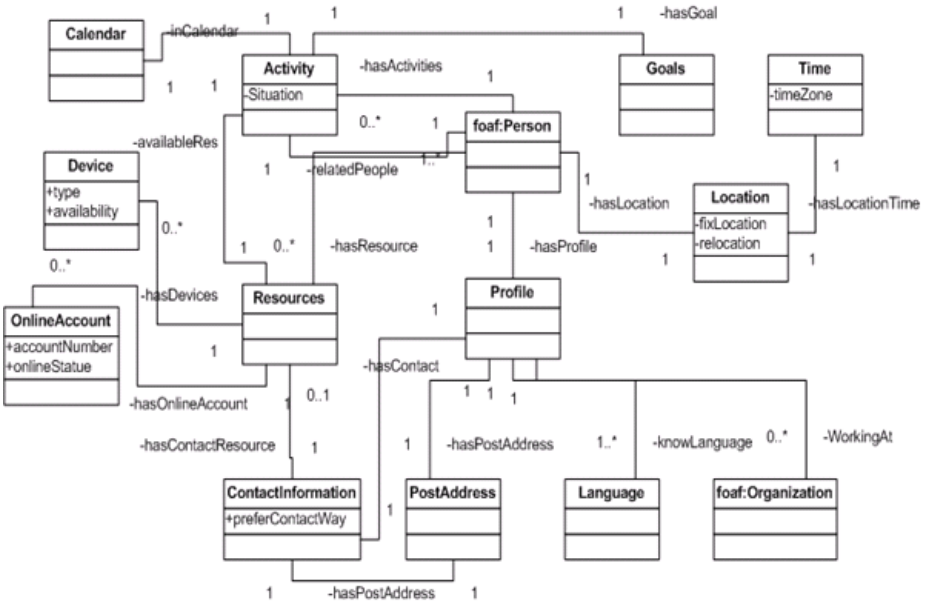


Fig. 1. The integration of 4 packages context model

**User profile context (Profile)** stores a user’s personal data. The profile links to other context properties: Language shows which languages this person knows and their proficiency level; ContactInfo connects to possible contact details of a particular person such as postal address, contact number or online contact details. For service selection, Language can be used to filter services that are usable by the user; ContactInfo might indicate which way of contact is preferred by the person. Person details make use of the FOAF ontology<sup>2</sup>.

**Resource context (Resources)** includes both electronic documents and artifacts, as well as physical resource such as devices available. The resource context allows determination of which devices, software and documents are available.

**Activity context (Activity)** describes everything a person is doing (maybe performed by a service) in order to fulfil a goal. The Goal property uses ontology-based keywords to describe the task and its desired outcomes. It also allows for special variants for e.g. emergency situations. These properties imply the functional requirements of the service. The Situation property influences the importance of different non-functional properties. For example, emergency situation change the weights applied to certain criteria.

<sup>2</sup> <http://xmlns.com/foaf/0.1/>

The physical location context (**Location**) is the detailed ontology for Location property. It indicates the location and time related constraints. A fixed location may have different representations such as GPS Coordinate or PostalAddress; [FipD07] provides more detail.

### 3 Context-Aware Criteria Generation

#### 3.1 Services Categories with Meta Data

Services are traditionally categorised by their functional properties, e.g. in the categorisation system used in UDDI [Org04]. This kind of service categorisation is insufficient for automatic service selection processes because it does not specify NPFs that are essential to differentiate functionally similar services in different situations. We propose to extend the functional properties based categorisation with details of NPFs, following a well-structured data model. We refer to this additional data as service meta data. Different service categories have different sets of relevant meta data. For example, printing services can consider colour options, while communication services might consider the transmission mode (e.g. synchronous).

The service registration process (see Figure 2) builds a link between the service and the category. Meanwhile, the OWL-S description of the service should specify the meta information data which is defined by the category. In the following we provide more details about the category, meta data (Meta) and service.

Each category has a name, which identifies the category (there is also an identifier for computer rather than human use). This is useful for service developers

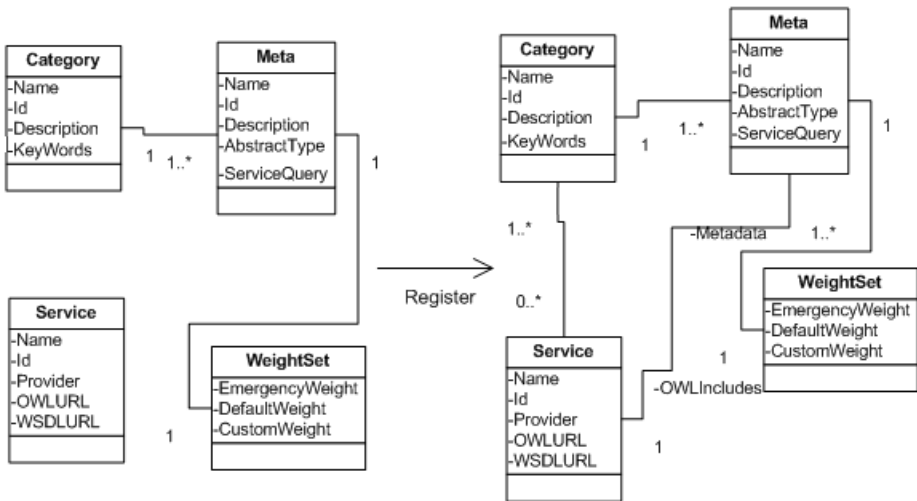


Fig. 2. The conceptual model of category, service and service registration

who wish to register a new service, however for searches and finer grained understanding of what the category represents a number of keywords describing the functional properties of services (or better operations) in the category are provided. A detailed description adds further detail for human use. Each category has a set of meta data associated to it, which captures the non-functional properties.

Each meta data element has an `AbstractType`, which is used to identify the correct evaluation function for this type of data. The associated `WeightSet` reflects the importance of this particular non-functional property in the category, from a service provider point of view. However, different situations require a shift in importance, as do individual users. So weights are more flexible in that they provide the default weight as specified by the provider, they allow for an emergency weight (usually defined for the application domain) and custom weights (usually defined by the end user).

In order to indicate that a small value is desirable, weights take on negative values in the range of  $[0,-1]$ , if a larger value is desirable values come from the  $[0, 1]$  interval. In addition, an absolute value of 1, means that the criteria is a hard constraint (that is it must be satisfied, or we are not interested in the service). Examples for these cases are cost (the smaller the better), speed (the faster the better) and availability in a certain country (e.g. a retail service not shipping to the UK would be of no interest to a UK customer).

While some meta data can be found in the service profile (for example the speed of a printer, or whether it prints in colour), there are some criteria that depend on the service context and need to be more up to date (e.g. the length of a print queue). To obtain such data the `ServiceQuery` specifies a SPARQL [Gro08](#) query statement which can be used to locate this kind of information from the service context.

A service is described by typical elements, such as information about its provider. the service. `OWLURL` is a link to the location of the OWL-S description file of the service, which should contain the required data for the non-functional attributes. `WSDLURL` provides a link to the services WSDL file, as is required for using the service in current web service technologies.

Registering a service involves linking this to the service category model, that is assigning a category for each service (or operation). This has the side effect of linking the service to typical non-functional criteria for which users might require values and this data is populated from the services OWL-S file.

### 3.2 Automated Criteria Generation

Based on the user context model and the service category model, we define context-aware criteria that link the two sides of user context and service non-functional properties and are generated automatically. Context-aware criteria consists of a number of criteria that are initialised from the meta data of the correct service category.

The idea is that this brings together the data required to evaluated the service. For example when considering a transport service from Leicester, it is clear that

the user would specify some values for locations, services would provide to be queried on those and more over the two values need to be available for evaluation. While we have shown this on a location example, the same has to be done for every other criteria of relevance for the application in question to include any appropriate aspects of the context model.

In detail the context-aware criteria consists of data from the service profile as well as data from the user context. It presents itself with an `AbstractType` which is used to identify the related evaluation function. In terms of user data it contains a value and a weight set, which are both derived automatically from the user's context using the context query. The context query is a SPARQL query extracting context information from the user context repository (the repository is structured according to the context model presented earlier). The `AbstractType` and name of criteria as well as the service query are extracted from the service profile. This process of extracting and merging data is completely automatic.

## 4 The TLE Service Selection Method

In sections 2 and 3 we have discussed the context model and how it is linked to the services' meta data. We will now focus on the service selection process. There are two major steps in the service selection process: First we need to evaluate each criterion of each service. Then we need to aggregate all criteria evaluation results to get an overall score for each service in order to select the most suitable service for the user. To complete these two steps, we use a Type-based LSP (Logic Scoring Preference [Duj96]) Extension (TLE) method which has been introduced in our previous work [YRM08]. The TLE method includes a type-based single criterion evaluation process and an extended LSP aggregation function.

### 4.1 Type-Based Evaluation Process

Most current criteria evaluation functions strongly rely on human input; usually this means that evaluation functions are designed and assigned to each criterion by hand, providing excellent results by allowing fine tuning of measurements. However, in the dynamic context, the evaluation function often requires to be adapted at runtime as the relevant criteria change. Therefore, human interaction is not acceptable and the method needs to be atomized.

The type-based evaluation process is designed to automatically match evaluation functions to the criteria at runtime based on each criteria's abstract type. Various types can be defined for different evaluation contexts and environments to extend this type-based evaluation process.

The *Numerical type* is used for criteria which take numerical input to the evaluation method such as cost, time and other quantitative measurement values. The evaluation function is given by Formula [II](#):

$$\varepsilon = \begin{cases} \frac{1-(v_{max}-v)}{v_{max}-v_{min}} & \text{iff } W \geq 0, \\ \frac{v_{max}-v}{v_{min}-v_{min}} & \text{otherwise} \end{cases} \quad (1)$$



where  $w$  is the weight of the criterion.  $v_{max}$  is the maximum value of all competing services,  $v$  is the value for the service under evaluation.  $v_{min}$  is the minimum value of all competitive services (if user context does not indicate a minimum value constraint, in which case that value is used). For example, the price criterion or service response time.

The *Boolean type* is used for criteria which are evaluated to 1 or 0. The function is:

$$\varepsilon = \begin{cases} 1 & \text{if criterion is met,} \\ 0 & \text{otherwise.} \end{cases} \quad (2)$$

The *Set overlap type* is used to define criteria which are measured by matching on instances on an enumerated set:

$$\varepsilon = \frac{\varepsilon_1 + \varepsilon_2 + \dots + \varepsilon_i}{n} \quad (3)$$

with  $\varepsilon_i$  being a score for each element of the set. For example, the constraint value of the available devices criterion is  $C_{ad} = \{mobile, PAD, laptop, IM\}$  in our notification service selection scenario.

The *Distance type* is used to evaluate the criteria which are measured by distance between two locations expressed by latitude and longitude.

$$\varepsilon = \begin{cases} R \times c & \text{iff } c \geq 1 \\ R \times 2 \times \arcsin(1) & \text{otherwise} \end{cases} \quad (4)$$

with  $c = 2 \times \arcsin \sqrt{\sin^2(\frac{|L2-L1|}{2}) + \cos(L1) \times \cos(L2) \times \sin^2(\frac{|G2-G1|}{2})}$ ,  $L1$  = latitude of the first point,  $G1$  = longitude of the first point,  $L2$  = latitude of the second point,  $G2$  = longitude of the second point and  $R$  = the Earth's mean radius of 6371 km. For example, the distance between injured person and the two medical support services is the crucial selection criterion in the medical support service scenario.

Our case studies focussed on some examples from the inContext project, of which the two scenarios mentioned earlier provide snapshots. A more complete description of the case studies can be found in [CMCe08]. We found the four types mentioned sufficed for the case studies however as this is only empirical we do not claim for the types to be complete. Nevertheless, more types can be simply added by specifying an evaluation function and type name; the type name is then used in the service meta data definition.

**Table 1.** Selection Criteria and Service NFPs Meta

Criteria ID	C1	C2	C3	C4	C5	C6
Criteria Name	Location	Devices	Price	Time	Privacy	PCW
Criteria Type	Boolean	Set overlap	Numerical	Numerical	Numerical	Set overlap
Weight	0.9	0.9	0.5	0.3	0.7	0.8

PCW = Prefer Contacting Way.

This method provides a link between the evaluation function and the specific criteria under investigation, and the appropriate function can be chosen at runtime. Table 1 shows the criteria examples of the select a notification service scenario.

## 4.2 Extended LSP Aggregation Function

Having defined how individual criteria can be evaluated, we turn our attention to the criteria aggregation function: vital for computing overall scores for a service. The LSP aggregation function [Duj96] modifies the traditional weighted sum aggregating function, to capture concepts such as replaceability (the fact that one criteria might be replaced, that is ignored, if another criteria is extremely well suited) or an mandatory-ness (the fact that a criteria can under no circumstances be ignored no matter how low the score is compared to other criteria). These concepts are captured in the power ( $r$ ) that is applied to each factor (see 5).

$$E = \left( \sum_{i=1}^n W_i E_i^r \right)^{\frac{1}{r}} \quad (5)$$

LSP was developed for manual evaluation, we extended the LSP function with an automatic process to determine the correct value of  $r$  based on the weight values of different criteria. The extended function is shown in formula 6, where  $W_i$  can be less than 0 to express that a smaller value is desirable for numerical type criteria (e.g. think about minimizing cost).

$$E = \left( \sum_{i=1}^n |W_i| E_i^r \right)^{\frac{1}{r}} \quad (6)$$

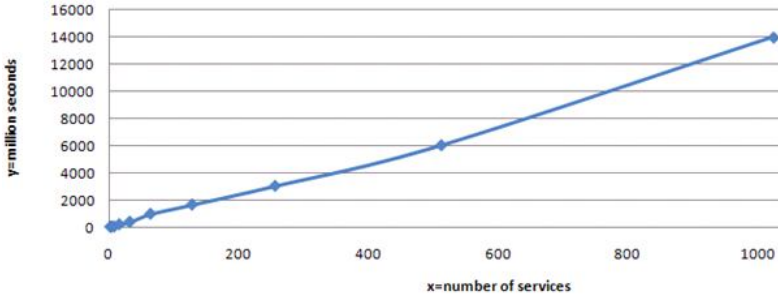
If we refer back to the table 1, then the concrete evaluation function is 7:

$$E = |W_{C1}| E_{C1}^r + |W_{C2}| E_{C2}^r + \dots + |W_{C6}| E_{C6}^r \quad (7)$$

Details of the evaluation function have been introduced in [YRM08]. In this paper we added the link to context information, so that the service selection decision can be made automatically.

## 5 Evaluation

While the ranking approach is implemented in terms of the relevance engine in the inContext platform, the engine itself is developed as a Web service so allows for easy embedding in different environments. For more detailed analysis we have developed a testbed that also considers the generation of the criteria as described here. The testbed includes an OWL/RDF context store, a repository that is organised by service category, but is enhanced with the meta data model and information and the relevance engine performing the TLE selection process. The evaluation reported here considers scalability and was conducted through



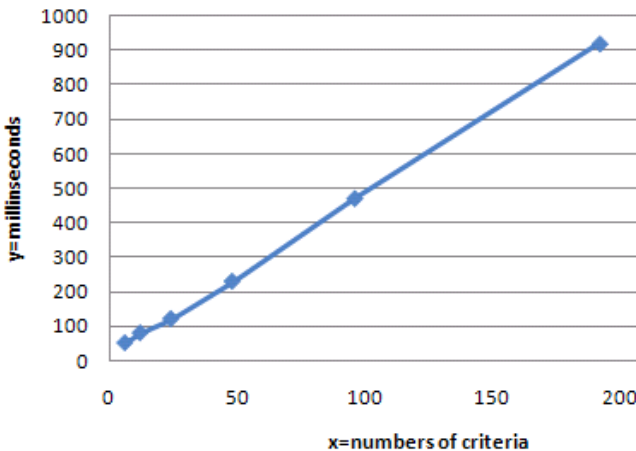
**Fig. 3.** Evaluation results for increasing numbers of services with a fixed number of criteria

3 evaluation cases for notification service selection scenarios. Within inContext the ranking method has been evaluated further on real case studies, focusing on functional correctness rather than scalability.

The first series of tests focuses on measuring the selection time when the number of services increases. There was a fixed number of criteria that was used to evaluate the services here (there were 6 criteria). We considered up to 1000 services.

Figure 3 shows that the approach is essential linear with respect to the number of services.

The second evaluation case was to evaluate the selection time in the light of increasing the number of criteria. We fixed the number of services to 4, but tested up to 192 criteria. The test results are shown in Fig. 4. We can again see that the approach is linear with respect to the number of criteria.



**Fig. 4.** Evaluation results for increasing numbers of criteria with a fixed number of services

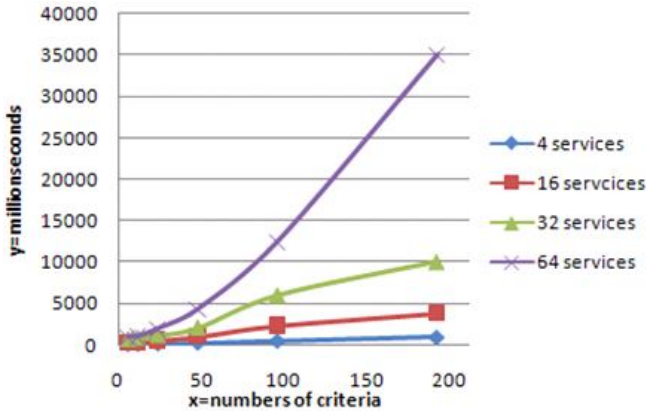


Fig. 5. Results for increasing number of services and criteria

In the last test we evaluated the selection time against both an increasing numbers of criteria and services. We defined a number of test groups with different service numbers and evaluated these against an increase in criteria. The evaluation results are shown in Fig. 5.

From the results, we can see that the scalability will be not dramatically decreased with an increasing number of criteria if the number of services is not too large (e.g. less than 16 services) or if we consider large number of services but smaller numbers of criteria (less than 96 criteria).

This merits some more general discussion: in the real world service selection scenarios, we do not expect there to be vast numbers of criteria, so around 100 seems a good pragmatic upper bound. Also, in terms of competing services, while we expect these to increase with more services becoming available, it seems safe to claim that a choice of 20 services fulfilling our functional requirements and hence been drawn into the comparison should be already a significant number.

Additionally, we should see that the services are chosen in a matter of seconds based on the prototype implementation (which has not been designed with performance in mind, but rather with functional correctness). When considering real service selection scenarios, the runtime of services usually exceed this time by far, and being presented with the best possible service will be ‘worth the wait’, even more so if we consider this selection to form part of a longer running business process possibly containing human tasks.

It would also be interesting to evaluate the selection correctness from a user point of view. However, correctness is difficult to define in general because it depends on different views and concerns. As is typical for multiple criteria decision problems, one can tell which decision is wrong, but it is very hard to say which one is better than others for humans. Large scale user testing would provide an evaluation and this has to some extent be conducted and recently in the inContext project.

## 6 Related Work

Several research approaches have considered using context information for selecting suitable services for the end-user. Location, which is introduced in Cooltown project [Pac04], [RT06] and the Jini [KEKW04] service discovery protocol, is the earliest form of context information used for service selection. These approaches can discovery and select the service nearest to the user. Nerveless, the context information is limited to location context only.

[CKL05] and [LH03] extended the context information by adding so called dynamic and static service attributes. The dynamic service attributes are those characteristics of a service whose values change over time. Other attribute are said to be static. Since there is more than a single context constraint, these works make use of weighted vector based aggregation functions for ranking the services and returning the top matches to the user. However, there are two main drawbacks:

1. the work is reliant on a syntactic representation of contextual information of services. Consequently, it is very difficult to apply more advanced semantic level searching, matching and reasoning techniques.
2. they only focus on modelling services' attributes/context information without specifying the user's context information. Thus, context-awareness means service non-functional properties awareness.

Work in [ESB06] addressed the first drawback by utilising concepts from the Semantic Web. However, it does not address the second problem of considering and modelling user context information. In contrast, [SVC+03] makes a lot of efforts on defining user's context information in details and identifies nine categories expressing user context: User information, Personal Information, Activity Information, Social Information, User Defined Rules, Environment Information, Application Information, Terminal Information and Network information. However, this work then expects service developers to build suitable new services in order to satisfy these context constraints.

In summary, current context-aware service selection methodologies do not bridge the gap between user's context and service's context. Few approaches provide a clear picture of using the user's context information for generating the service selection criteria/constraints. Furthermore, the ranking methods are far more simplistic than what is really required to cope with the context mapping between user and services. The presented work bridges between service and user context and provides a powerful, yet scalable ranking approach.

## 7 Conclusion and Future Work

In this paper, we developed a context model including four aspects typical for collaborative systems: user profile, resources, activities and physical environment. In order to link the user context information to the service selection, we defined a meta data based category system and context-aware criteria which

can be automatically generated by querying both user context and service non-functional meta data. Context-aware service selection also requires a suitable selection method to use the criteria. Therefore, we introduced the TLE selection method, which is based on type-based evaluation functions and an extended LSP aggregation method. Our evaluation results show capability in both increasing numbers of services as well as increasing numbers of criteria. We illustrated the need for this work through two typical scenarios.

The presented selection method has been developed and applied for service selection as presented in this paper, however in its more basic form it can be transferred to other domains where automatic selection based on a number of criteria is required. One such transfer has been made to select appropriate people for a people finder service [YHHRM07].

There are some issues worth future exploration. On one hand, it would be worthwhile to explore how other services selected as part of a workflow provide additional context and requirements. In fact, we have recently started to look at this issue and initial result has been presented in [YRMT08]. On the other hand, it is worthwhile to enable reasoning on context information to determine criteria weights automatically rather than statically providing weights for each user or service domain.

**Acknowledgments.** This work is partially supported by EU inContext (Interaction and Context Based Technologies for Collaborative Teams) project: IST-2006-034718.

## References

- [CKL05] Cuddy, S., Katchabaw, M., Lutfiyya, H.: Context-aware service selection based on dynamic and static service attributes. In: Proceedings of IEEE International Conference on Wireless And Mobile Computing, Networking And Communications, vol. 4 (2005)
- [CMCe08] Yu, H.Q., Schall, D., Melchiorre, C., Reiff-Marganiec, S., Dustdar, S.: Incontext – interaction and context-based technologies for collaborative teams. In: Cunningham, P., Cunningham, M. (eds.) Collaboration and the Knowledge Economy: Issues, Applications, Case Studies. IOS Press, Amsterdam (2008)
- [Duj96] Dujmovic, J.J.: A method for evaluation and selection of complex hardware and software systems. In: Proceedings of 22nd International Conference for the Resource Management and Performance Evaluation of Enterprise Computer Systems, Turnersville, New jersey (1996)
- [Dus04] Dustdar, S.: Caramba—a process-aware collaboration system supporting ad hoc and collaborative processes in virtual teams. *Distrib. Parallel Databases* 15(1), 45–66 (2004)
- [ESB06] El-Sayed, A.-R., Black, J.P.: Semantic-based context-aware service discovery in pervasive-computing environments. In: Proceedings of IEEE Workshop on Service Integration in Pervasive Environments. SIPE (2006)

- [FipD07] European Union FP6 Framework and inContext project Deliveries. Design and proof-of-concept implementation of the incontext context model version 1 wp 2.2 (2007)
- [Gro04] RDF W3C Working Group. Rdf/xml syntax specification (2004), <http://www.w3.org/TR/rdf-syntax-grammar>
- [Gro08] W3C SPARQL Standard Group. Sparql query language for rdf (2008), <http://www.w3.org/TR/rdf-sparql-query>
- [KEKW04] Klimin, N., Enkelmann, W., Karl, H., Wolisz, A.: A hybrid approach for location-based service discovery. In: Proceedings of International Conference on Vehicular Ad Hoc Networks (2004)
- [LH03] Lee, C., Helal, S.: Context attributes: An approach to enable context-awareness for service discovery. In: Proceedings of SAINT 2003 (2003)
- [Org04] OASIS Organisation. Uddi version 3 specification, oasis standard (2004)
- [OWL] Owl web ontology language, <http://www.w3.org/TR/owl-ref/>
- [Pac04] Hewlett Packard. Cooltown project (2004), <http://www.cooltown.com/cooltown/>
- [RMT<sup>+</sup>08] Reiff-Marganiec, S., Truong, H.-L., Casella, G., Dorn, C., Dustdar, S., Moretzki, S.: The incontext pervasive collaboration services architecture. In: Mahonen, P., Pohl, K., Priol, T. (eds.) Proceedings of Service Wave 2008. LNCS, vol. 5377, pp. 134–146. Springer, Heidelberg (2008)
- [RT06] Riva, O., Toivonen, S.: A hybrid model of context-aware service provisioning implemented on smart phones. In: Proceedings of ACS/IEEE International Conference on Pervasive Services (2006)
- [SVC<sup>+</sup>03] Sygkouna, I., Vrontis, S., Chantzara, M., Anagnostou, M., Sykas, E.: Context-aware services provisioning on top of active technologies. In: Horlait, E., Magedanz, T., Glitho, R.H. (eds.) MATA 2003. LNCS, vol. 2881, pp. 67–76. Springer, Heidelberg (2003)
- [YHHRM07] Yu, H.Q., Hong, Y., Heckel, R., Reiff-Marganiec, S.: Context-sensitive team formation: Towards model-based context reasoning and update. In: 6th International and Interdisciplinary Conference on Modeling and Using Context, Doctorial Consortium (2007)
- [YRM08] Yu, H.Q., Reiff-Marganiec, S.: A method for automated web service selection. In: Proceedings of 2nd IEEE International Workshop on Web Service Composition and Adaptation (WSCA 2008) Special Theme: Dynamic Services Composition and User Steering held in conjunction with 6th IEEE International Conference on Services Computing (SCC 2008), Honolulu, USA (2008)
- [YRMT08] Yu, H.Q., Reiff-Marganiec, S., Tilly, M.: Composition context for service composition. In: Proceedings of IEEE International Conference on Web Service, WIP Track (2008)

# Development Framework for Mobile Social Applications

Alexandre de Spindler, Michael Grossniklaus, and Moira C. Norrie

Institute for Information Systems, ETH Zurich  
CH-8092 Zurich, Switzerland  
{despindler,grossniklaus,norrie}@inf.ethz.ch

**Abstract.** Developments in mobile phone technologies have opened the way for a new generation of mobile social applications that allow users to interact and share information. However, current programming platforms for mobile phones provide limited support for information management and sharing, requiring developers to deal with low-level issues of data persistence, data exchange and vicinity sensing. We present a framework designed to support the requirements of mobile social applications based on a notion of P2P data collections and a flexible event model that controls how and when data is exchanged. We show how the framework can be used by describing the development of a mobile application for collaborative filtering based on opportunistic information sharing.

**Keywords:** Mobile Social Applications, Development Framework, Adaptive Middleware.

## 1 Introduction

The increased computational power and storage capacity of mobile phones now makes them capable of hosting a wide range of multimedia services and applications. In addition, the integration of sensing devices such as GPS and connectivity such as Bluetooth and WiFi has made it easier to support location-based services and new forms of information sharing.

As a result of these technical innovations, service providers and application developers are keen to exploit a new potential market for mobile social applications that allow users to interact and share data via their mobile phones. However, programming platforms for mobile phones currently provide little support for flexible forms of information management and sharing. In a rapidly emerging and highly competitive market, this presents companies with a major challenge in terms of the effort required to prototype and validate potential applications.

To address this problem, we have designed an application development framework to support the requirements of mobile social applications. The framework ensures that developers can work at the level of the application domain model, without having to deal with the low-level mechanisms provided in current



platforms for dealing with peer-to-peer (P2P) information sharing, data persistence and location sensing. Instead, applications can be designed around a novel concept of P2P collections of persistent data coupled with a flexible event model that can determine how and when data is exchanged and processed.

In this paper, we present the requirements of mobile social applications along with the limitations of existing platforms with respect to these requirements. We then provide the details of our framework and demonstrate its use by describing how we developed an application for collaborative filtering based on P2P information sharing in mobile environments.

Section 2 discusses the limitations of existing platforms for mobile phones with respect to the goal of supporting the development of mobile social applications. In Sect. 3, we then examine the requirements of mobile social applications in detail and describe how our framework supports these requirements. Details of P2P collections and the event model are given in Sect. 4 and Sect. 5, respectively. In Sect. 6, we describe how the collaborative filtering application was implemented using the framework. Concluding remarks are given in Sect. 7.

## 2 Background

Mobile phones are no longer simply regarded as communication devices, but rather as computing devices capable of, not only hosting a range of applications, but also communicating with each other. This has led to a great deal of interest in mobile social applications which can take advantage of these capabilities to allow users to interact and share information in innovative ways. Applications have been proposed that exploit ad-hoc network connections between phones via Bluetooth or WiFi to support user awareness of social contexts [1,2] or to automatically exchange data between users in shared social contexts [3,4]. In particular, physical copresence has been used as a basis for forming a weakly connected community of users with similar tastes and interests [5,6].

A variety of development toolkits for mobile phones are available. These range from vendor-specific solutions such as iPhone SDK<sup>1</sup>, Windows Mobile Edition<sup>2</sup>, Symbian<sup>3</sup> and Google Android<sup>4</sup> to the platform independent Java WTK (Wireless Toolkit). These provide integrated emulation environments along with support for the development of user interfaces. They also provide access to typical phone features such as personal information management (PIM) data, the camera, Bluetooth, Internet access and GPS. However, the development of mobile social applications using these toolkits still requires considerable effort since they provide no high-level primitives to support vicinity sensing, location awareness, information sharing and data persistence. As a result, developers have to

---

<sup>1</sup> <http://developer.apple.com/iphone>

<sup>2</sup> <http://www.microsoft.com/windowsmobile>

<sup>3</sup> <http://www.symbian.com>

<sup>4</sup> <http://code.google.com/android>

implement components to handle requirements related to these issues for each application and each target platform.

For example, Java WTK uses a simple key-value store for data persistence which means that developers have to define and implement the mapping between Java application objects and key-value pairs for each application. This contrasts with development platforms for PCs such as db4o<sup>5</sup> that support Java object persistence. Support for information sharing is also limited in these platforms and data sharing must be implemented based on sockets able to send and receive binary data. The developer must therefore implement the facilities to serialise and deserialise data, to open and listen to sockets and stream data. Short-range connectivity such as Bluetooth or WiFi can be used to react to peers appearing in the physical vicinity. Using Java WTK, the developer has to implement two listeners, one registered for the discovery of a device and another which is notified about services discovered on a particular device. For each scan of the environment, both listeners must be registered and the developer must also implement the coupling of peer discovery with data sharing.

Frameworks have been developed specifically for P2P connectivity including Mobile Web Services [7] and JXTA [8], but these tend to focus on lower-level forms of data exchange rather than information sharing. For example, JXTA provides the notion of a peer group as a central concept of their metamodel. A group mainly provides facilities for peer discovery, service publishing and message routing. Application development consists of specifying message formats and how they are processed in terms of request and response handling similar to that of service-oriented architectures. This results in a blending of the application logic typically embedded in an object-oriented data model and the collaboration logic specified based on a request-response scheme. Efforts to provide higher level abstractions of P2P networks have either focussed on the allocation and retrieval of identifiers to resources in fixed networks without considering any notion of handling [9] or they offer only a few limited collaboration primitives and lack support for vicinity awareness [10,11].

Within the database research community, a number of P2P database systems, overlay networks and middlewares have been developed including Pastry [12], Pizza [13], PeerDB [14], Hyperion [15], P-Grid [16] and GridVine [17]. However, research efforts have tended to focus on issues of object identity, schema matching and query reformulation, distributed retrieval, indexing and synchronisation as well as transaction management. To date, there has been little effort on supporting developers of mobile applications that utilise P2P connectivity to share information opportunistically with other users in the vicinity.

Based on our own experiences of developing mobile social applications using existing platforms, we realised that there was a need for an application framework that offers functionality for P2P information sharing as high-level primitives. In the next section, we examine the requirements of such a framework in detail before presenting an overview of the framework that we have developed.

---

<sup>5</sup> <http://www.db4o.com>

### 3 Framework

A distinguishing feature of mobile social applications is the notion of collaboration. Each peer follows a set of application-specific rules which determine its behaviour within the collaborative environment. This behaviour includes the local creation, storage and processing of data as well as interacting with other peers by sending, receiving and forwarding data. Such behaviour may be triggered automatically or explicitly by the user. Each peer offers the services of the application to the user independently of the other peers, but the effectiveness of these services depends on the combined effects of local peer behaviour.

To examine the requirements of mobile social applications and illustrate how our framework supports these requirements, we will consider the example of a recommender system. Due to space limitations, a more comprehensive examination cannot be presented here. In previous work [18], we have shown how collaborative filtering (CF) algorithms can be adapted to mobile settings using physical copresence in social contexts as a basis for measuring user similarity. Figure 1 will be used to illustrate how such an application works. Assume users rate items such as music, films or places to go and this data is stored as a collection  $C$  of triples  $(u, i, r)$  where  $u$  is a user,  $i$  an item and  $r$  a rating. Essentially, we can view the collaborative filtering process as some function  $f$  that is applied to  $C$  to return the result recommendation  $R$  as a list of items. The details of the function  $f$  are not relevant to this discussion, but what is important is that each peer has an instance of  $C$  and will locally compute  $f(C)$  when the recommender service is called. We refer to such application-specific services as the *application logic* of the system, and they may be executed either automatically or upon an explicit request by the user.

An application may have multiple data collections defined by a schema shared by all peers, say  $\{C_1, C_2, \dots, C_n\}$  and a set of participating peers  $\{P_1, P_2, \dots, P_m\}$ .

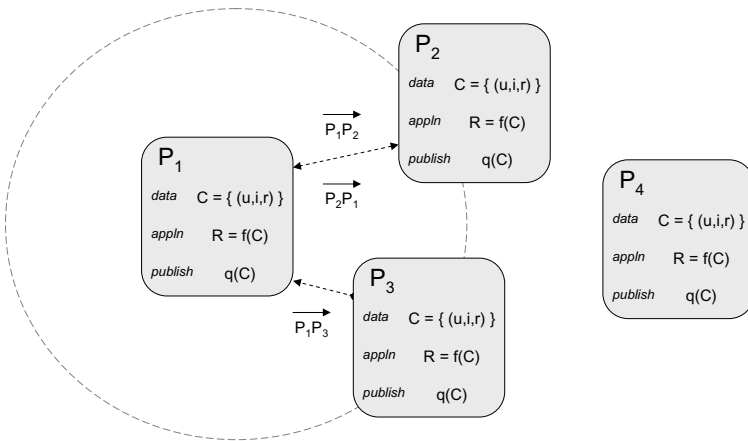


Fig. 1. Collaborative Filtering as a Mobile Social Application

Each peer will have its own instance of each of the application collections and we use  $C_i|P_j$  to denote the instance of the collection  $C_i$  stored on peer  $P_j$ . Note that we prefer to refer to  $C_i|P_j$  as an instance of collection  $C_i$  rather than as a part of some global collection  $C_i$  since the application services running on  $P_j$  will operate only on the locally stored collection of the appropriate name, independently of the collections stored on other peers. In the case of the collaborative filtering application illustrated in Fig. 1, there are four peers each of which has a single data collection  $C$  containing rating triples of the form  $(u, i, r)$  and an application to compute the CF result denoted by  $R = f(C)$ .

Computing user similarity in centralised CF algorithms can be computationally expensive. In mobile settings, a much simpler approach can be used which takes advantage of the fact that local data comes only from the owner of the device, or from users with similar tastes and interests. The underlying assumption is that users who are close enough to exchange data through ad-hoc connections between mobile devices share social contexts and hence are likely to have similar tastes and interests. Detailed studies related to this assumption have been carried out in a number of projects, see for example [5,6,18], and it is beyond the scope of this paper to discuss this aspect in detail. Our interest here is the fact that mobile social applications often involve some form of opportunistic sharing of information based on ad-hoc connectivity between mobile devices, or possibly mobile and stationary devices. It is therefore important that a development framework for mobile social applications supports a notion of vicinity awareness.

At a given time  $t$ , the vicinity of a peer  $P_i$  is the set of peers to which  $P_i$  is connected, and we denote this by  $V_t(P_i) = \{P_1, \dots, P_k\}$ . In Fig. 1 we use a dashed circle to denote the connectivity range of  $P_1$  at time  $t$  and, hence,  $V_t(P_1) = \{P_2, P_3\}$ . If  $P_j \in V_t(P_i)$  then it is possible for peers  $P_i$  and  $P_j$  to exchange data. The *collaboration logic* of an application will specify *if*, *when* and *what* data is shared. We will discuss the details of how the *if* and *when* can be specified later in the paper when we present the details of our framework. The *what* is specified by associating a query expression  $q_i$  with each application collection  $C_i$ . We use  $\overrightarrow{P_j P_k}$  to denote an exchange of data from  $P_j$  to  $P_k$ . This means that if  $P_j$  and  $P_k$  both have instances of collections  $\{C_1, C_2, \dots, C_n\}$  then

$$\overrightarrow{P_j P_k}: \forall C_i \in \{C_1, C_2, \dots, C_n\}, C_i|P_k := q_i(C_i|P_j) \cup C_i|P_k$$

Figure 1 shows a case where  $P_1$  and  $P_2$  exchange data bilaterally, meaning that each peer sends rating tuples to the other peer and adds the data to its local  $C$  collection. The query expression  $q$  acts as a filter on the data to be published. In the case of collaborative filtering, only the data pertaining to the actual user of the device, and hence the user currently in the same social context, will be sent to the other peer. In the case of the connection between  $P_1$  and  $P_3$ ,  $P_3$  sends data to  $P_1$  but  $P_3$  does not receive data from  $P_1$ . This is indicated in the figure by the fact that the connection between  $P_1$  and  $P_3$  has an arrow in only one direction. It could be the case that  $P_3$  had previous encounters with  $P_1$  and found their data unreliable and hence placed them on some sort of black list to indicate that they did not want to receive data from them in the case of future

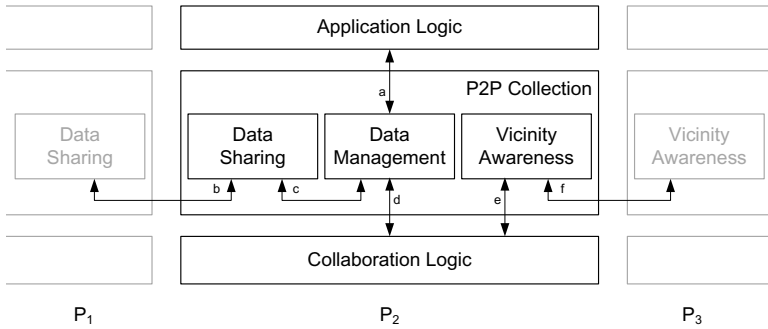


Fig. 2. Framework Overview

encounters. Note that, in practice, it might be that  $P_1$  would publish the data, but  $P_3$  would simply choose not to receive it. Details of this will be given in later sections.

Generally, it should be possible for mobile social applications to have flexibility in determining how and when peers exchange data. For example, there are many ways in which applications might want to control the exchange of data for reasons of privacy. Within our framework, we provide a flexible event processing model that allows applications to determine how and when they share data.

Having looked at the general operation and key requirements of mobile social applications, we see that there are three main functionalities that a framework needs to support. First, it needs to provide basic services for the management and querying of data collections. Second, it should offer developers high-level abstractions to enable data from those collections to be shared via ad-hoc connections to peers. Third, it needs to provide vicinity awareness. At the same time, it is important to separate the concerns of application logic and collaboration logic to ensure maximum flexibility in meeting the requirements of a broad spectrum of users, devices and applications. Figure 2 presents an overview of our framework. The concept of a P2P collection is the central component which encapsulates persistent data storage, data sharing and the ability to sense peers entering and leaving a peer’s physical vicinity. As an interface to application logic and user interaction, the framework offers standard data management facilities such as the creation, retrieval, manipulation and deletion of data (a). These facilities are offered in terms of a database management system which includes transaction management support. Furthermore, it offers a second interface allowing collaboration logic to be specified and executed in terms of events and their handling (d,e). By keeping these interfaces independent, we are able to achieve the required separation of concerns. The actual scanning of the physical environment (f) and data sharing (b,c) is encapsulated by the framework.

In the next section, we will present the concept of P2P collections in detail before going on to describe how the collaboration logic can be specified by means of the event processing system.

## 4 P2P Collections

Programming languages such as Java and C++ have standard libraries that offer various types of collections in terms of interface definitions that declare operations to insert, retrieve and remove data along with concrete implementations that provide the corresponding functionality. Following this paradigm, the central component of our framework—the peer collection—is an alternative collection implementation that provides additional functionality to address the requirements of mobile social applications.

Most programming systems define collections in terms of a collection behaviour and a member type. For example, Java offers collection implementations for sets, lists and maps that, through the use of generics, can be bound to a member type that restricts the possible members of the collection. Our definition of a peer collection follows this approach but extends it to cope with more specific requirements. Generally, a peer collection is characterised by its name  $n$ , its member type  $t$  and its behaviour  $b$ . As we will see, the use of a name to identify the collection is motivated by the requirements of data sharing in a peer-to-peer environment that makes it necessary to identify collections across peers. Our framework introduces additional collection behaviours to support data management. The behaviour  $b \in \{\text{set, bag, sequence, ranking}\}$ , where  $\{\text{set, bag}\}$  are unordered,  $\{\text{sequence, ranking}\}$  are ordered and  $\{\text{set, ranking}\}$  have no duplicates while  $\{\text{bag, sequence}\}$  do.

Similar to common programming environments, methods to add, retrieve and remove data to/from a collection provide basic data management. Peer collections can optionally be marked as persistent with the effect that not only the collection, but also the members are automatically made persistent in a transparent way. In addition, our framework has support for events that get triggered whenever elements are added to, or removed from, peer collections. In Sect. 5, we will discuss how this mechanism can be leveraged to support the decoupling of the collaboration logic.

Our framework also features a low-level query facility that surpasses the data retrieval mechanisms offered by current collection implementations. A query is specified by building a query tree where the inner nodes represent query operations and leaf nodes contain query arguments. Once a query tree has been constructed, its root node is passed to the query evaluator component of the framework which processes the query and returns the result. While a complete presentation of our query facility is outside the scope of this paper, Tab. 1 gives an overview of the most important nodes including those we refer to in this paper. A node may have child nodes and attributes. For example, a selection node has a collection from which members are to be selected as a child and an attribute containing the selection predicates. In order to simplify the task of creating frequently used queries, a query tree builder is provided with the framework. Given the required parameters, it automatically builds the query tree and returns its root node.

Peer collections also address the requirements of data sharing. This additional functionality is provided through a set of methods that can be used to make

**Table 1.** Example query tree nodes, their children and attributes

Node	#Child Nodes	Attributes
Selection	1	predicates
Intersect	2	–
Union	2	–
Map	1	function
Attribute Access	1	attribute
Collection	–	collection

a collection available for sharing, connect it to other peers and exchange its members. In order for two peers  $P_1$  and  $P_2$  to share data, both peers have to make the collection to be shared available. When the two peers enter in each other's vicinity, available collections can be connected if they have the same name  $n$  and member type  $t$ . Once two peer collections are connected, all or some of the collection members from each peer are sent to the other peer. A query expression attached to the collection determines which members are sent.

Based on these basic sharing capabilities, our framework also provides a flexible mechanism to control what data is exchanged. This can be done in two ways. A selection query can be bound to a collection to filter data sent to peers. These filter queries are also expressed and evaluated based on the framework's query facilities presented above. In addition, white and black lists can be used to control with which peers data is exchanged. Thus, a collection that has been made available is associated with a positive and negative neighbourhood of peers. The positive neighbourhood contains those peers with which members are shared if they appear in vicinity, while other peers in vicinity will be ignored. If the positive neighbourhood is empty, members will be shared with any peer appearing in vicinity. The negative neighbourhood optionally contains those peers that should not be considered for data sharing even if they appear in vicinity. Similar to the positive neighbourhood, if that collection is empty, no restrictions are assumed to exist. These two neighbourhoods therefore enable a user to define constraints over the social network within which data is shared.

Our framework offers support for vicinity awareness which is used to react upon the appearance or disappearance of peers in the physical vicinity. As well as triggering events to connect collections and share data as described above, an application may react on such events directly. We will present the event mechanism offered by the framework in more detail in the next section.

Based on the Java programming language in conjunction with Java WTK platform, we will now describe how the framework can be implemented. Other programming languages such as C++ or Objective C as well as other platforms such as Symbian, iPhone SDK or Google Android can be supported analogously.

The implementation of the framework consists of two parts. First, there is the application programming interface (API) visible to the developer of a mobile social application together with the implementation of functionality that is common to all platforms. Then, there is the service provider interface (SPI) which needs to be implemented to support the peer collection framework on a

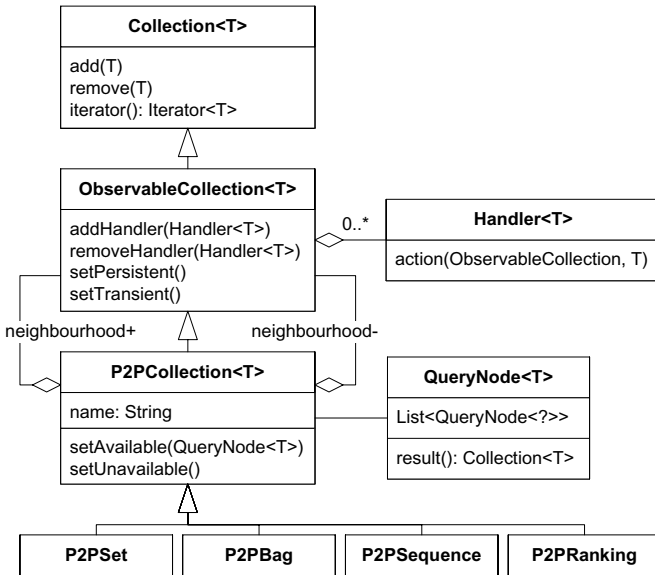


Fig. 3. API of the peer collection framework

given mobile phone and, thus, represents the platform-specific implementation. While other frameworks such as Java WTK address a similar problem, they usually do not cover the entire range of existing devices. In the remainder of this section, we will describe both parts of the framework in turn. Figure 3 gives an overview of the peer collection framework API that is based on the concepts described above. At the top of the figure, a simplified version of the existing Java collection interface is shown, highlighting the methods for adding, retrieving and deleting collection members. Our framework extends the Java collection interface and introduces an interface `ObservableCollection<T>`. Observable collections support the registration of handlers that are invoked whenever an event is triggered through the addition or removal of a collection element. A peer collection is represented by interface `P2PCollection<T>` that defines a collection name as well as methods to make the peer collection available or unavailable. The four different collection behaviours are provided through dedicated implementations of the peer collection interface. Finally, the query facility of the framework is supported by `QueryNode<T>` which serves as the common interface of the various query nodes discussed earlier.

The peer collection framework SPI defines the interfaces for three platform-dependent components and is shown in Fig. 4. One component offers persistent data storage, another the connection technology and a third the scanning of the physical vicinity for other peers. Note that, in contrast to existing platforms, these components have to be implemented once per platform rather than once per application. All persistence mechanisms make use of a single class offering database facilities such as storing, retrieving and deleting objects. This class is



<b>StorageProvider</b>	<b>SharingProvider implements &lt;&lt;Observable&gt;&gt;</b>	<b>Scanner implements &lt;&lt;Observable&gt;&gt;</b>
register(Collection) unregister(Collection) store(Collection, Object) delete(Collection, Object) select(Collection, Predicate): Collection intersect(Collection, Collection): Collection union(Collection, Collection): Collection collection(String): Collection	turnOn(Peer) turnOff() send(Object, Peer, Map<String, Object>) notify(Object, Map<String, Object>)	scan() start(Frequency) stop() notify(Collection<Peer>)

Fig. 4. SPI of the peer collection framework

defined in terms of interface **StorageProvider** and the framework makes use of it based on this interface only which allows it to be adapted to any underlying persistent storage technology such as a record store in the case of Java WTK.

To send and receive data, our framework makes use of a component that is dependent on the connection technology. The implementation of this component is abstracted through interface **SharingProvider**. Developers may turn on and off its availability within the collaborative environment. When turned on, the technology-specific information for reaching the local peer must be provided. In the case of Java sockets, this information includes a host identifier and a port number. Once the peer is turned on, collections may be made available. The peer can be turned off at any time, in which case, no more collections are available and the local peer is no longer available to other peers. This component is defined by a generic interface and can thus be implemented for different connection technologies such as Java sockets, WiFi and Bluetooth.

Finally, connection technologies such as Bluetooth and WiFi are used to scan the physical environment of a peer and discover other peers nearby. As with the other two components, the scanning is implemented by a platform-dependent component which is defined by and used through the interface **Scanner**. This interface declares the means to perform a single scan as well as starting a periodic scan with a frequency that can be specified.

## 5 Event Processing

Within our framework, it is the appearance of a peer in the vicinity of another that drives the sharing process. Thus, data sharing is linked with an event system composed of events for which handlers can be registered to be notified. While events and handlers can be specified by an application developer, predefined system events exist. Table 2 shows these events along with the arguments to which they are attached and the parameters passed to the registered handlers. To register a handler for an event, the developer needs to implement interface **Handler<T>** shown in Fig. 3 and specify an action method to be executed.

As part of the framework, a system collection **Vicinity** is provided. This collection is maintained by the framework and its members represent those peers that are currently in the physical vicinity. Whenever a new peer is detected, a new object is created and added to the **Vicinity** collection. When a peer moves away,

**Table 2.** Events, their arguments and parameters passed to the handlers

Event	Argument Parameters	
New Object	–	Object
Object Changed	Object	Object, Attribute
Member Added	Collection	Collection, Member
Member Received	Collection	Collection, Member, Source Peer
Member Removed	Collection	Collection, Member

the respective member is removed from the collection. Note that this collection is not set to be persistent. Since our event model generally supports the triggering of actions when objects are added to collections, vicinity awareness is realised based on addition events associated with the `Vicinity` collection that will be triggered when a new peer enters the physical vicinity. If a collection is made available, a predefined handler for sending collection members is automatically registered with this addition event.

As explained previously, when a collection is made available, the root node of a selection query is passed along. The query is handed over to the handler. The handler action consists of executing the query and sending the result.

## 6 Collaborative Filtering

To show how our framework is used, we present the implementation of the recommender system introduced in Sect. 3 as a use case. A detailed description of the system has been presented in [18]. A fundamental ability of recommender systems is to infer a rating for a requesting user about a target item unknown to the user. Based on this query, all items known to the system can be sorted according to the inferred rating for a requesting user. In order to recommend an item, the best ranked item(s) can be presented to the user.

Ratings are tuples that contain references to a user and item and the rating value. Consequently, a new tuple is created whenever a user makes a rating. In order to process the fundamental query, a filtering algorithm such as user-based collaborative filtering processes the collection of tuples as follows.

1. Compute the similarity of the requesting user to all other users.
2. Select  $n$  most similar users.
3. Aggregate the rating values of the users selected in step 2 for the target item.

The resulting aggregation is the rating value inferred for the requesting user about the target item. However, as was mentioned in Sect. 3, the first two steps can be omitted in a mobile setting where users exchange their own ratings whenever they are in each other's vicinity. Therefore, the main components of this recommender application can be summarised as follows. The application model consists of user and item entities and a relationship representing rating tuples. The application logic performs the rating inference by retrieving all rating tuples

stored locally which contain the target item and aggregating their rating values. The collaboration consists of sending rating tuples made by the local user whenever that user encounters other users in the vicinity while consuming items. Note that peers do not share tuples as soon as they are in each other's vicinity but wait for a configurable amount of time before starting the data transmission.

For illustration, we now describe how this application is implemented using the Java WTK platform. In a first step, the application model is mapped to the Java object model. The concepts of a user and an item are described by classes `User` and `Item`, respectively. These classes declare at least one identifier attribute allowing their instances to be recognised as equal when they are shared among peers. Additionally, attributes such as names and descriptions can be added to provide the users with meaningful information. Finally, we define the class `RatingTuple` to represent ratings as shown below.

```
public class RatingTuple {
    User user;
    Item item;
    float rating;
}
```

To access and share ratings, we create a peer collection named `RatingTuples` with set behaviour and `RatingTuple` as its member type. The following code shows the creation of this collection and how it is set to be persistent.

```
P2PSet<RatingTuple> ratingTuples =new P2PSet<RatingTuple>("RatingTuples");
ratingTuples.setPersistent();
```

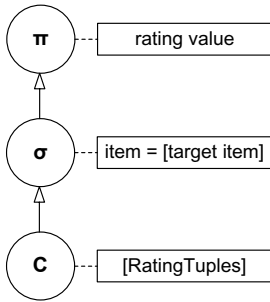
Having modelled the application in Java, the developer uses the data management facilities provided by peer collections to implement the application logic. In our simple example, the application logic consists of two main components. First, it needs to give the user the possibility of generating ratings and storing them persistently. Second, it needs to be able to infer ratings about items unknown to the user. The following example shows how to store a user rating by creating a new member of the `RatingTuples` collection. Note that the amount of code is equivalent to that required for existing Java collections.

```
RatingTuple tuple = new RatingTuple(localUser, item, rating);
ratingTuples.add(tuple);
```

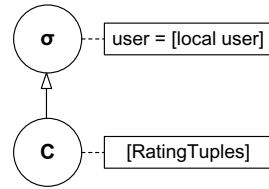
To infer ratings, all members of `RatingTuples` containing the target item must be selected. To do so, the query in Fig. 5 is used. It consists of a selection operation where the attribute comparison predicate constrains the item attribute to point to the target item. An attribute access node performs a projection to obtain the rating values of all tuples returned by the selection node.

Using the query tree builder, the code required to construct this query is given below. At runtime, once this query has been executed, the application logic can simply aggregate the rating values returned by the projection node.

```
QueryNode<RatingTuple> collection = Queries.collection("RatingTuples");
QueryNode<RatingTuple> selection =
    Queries.select(collection, "item", targetItem);
QueryNode<Float> projection = Queries.project(selection, "rating");
```



**Fig. 5.** A query selecting rating tuples containing the target item



**Fig. 6.** A query selecting rating tuples containing the local user

To implement the opportunistic sharing of rating tuples, the query to be executed when a peer appears in the vicinity must be specified. Since we only want to send those rating tuples containing the local user, the selection query shown in Fig. 6 is built using the statements given below.

```

QuerNode<RatingTuple> selection =
    Queries.select(collection, "user", localUser);

```

This selection query is given as an argument when the `RatingTuples` collection is made available.

```
ratingTuples.setAvailable(selection);
```

We now compare the effort required to implement this application with that required if using Java WTK. Figure 7 compares the components needed and the amount of interaction required to implement data management, vicinity awareness and data sharing using Java WTK, on the left hand side, and our framework, on the right hand side. To implement the application logic using Java WTK, a Java collection is used to maintain all rating tuples. Consequently, when all tuples with a particular item must be selected, all tuples would have to be accessed in order to select those having the required attribute value. The program code implementing this behaviour would be part of the application logic whereas, using our framework, it is hidden away from the developer by the query facility. The transparent persistence mechanism is a great improvement compared to Java WTK where application objects have to be serialised manually and stored using key-value records. In order to store objects of a particular type based on key-value pairs, an application developer has to program a database-like component and put a lot of effort into overcoming the impedance mismatch between objects and key-value pairs. If objects of different types must be stored, the required effort increases even further and, allowing stored objects to reference each other, would make this even more challenging. As opposed to the simple vicinity awareness mechanism provided by our framework, the developer of a Java WTK application needs to implement the scanning of the environment based on low-level connection technologies such as Bluetooth or WiFi. Moreover,

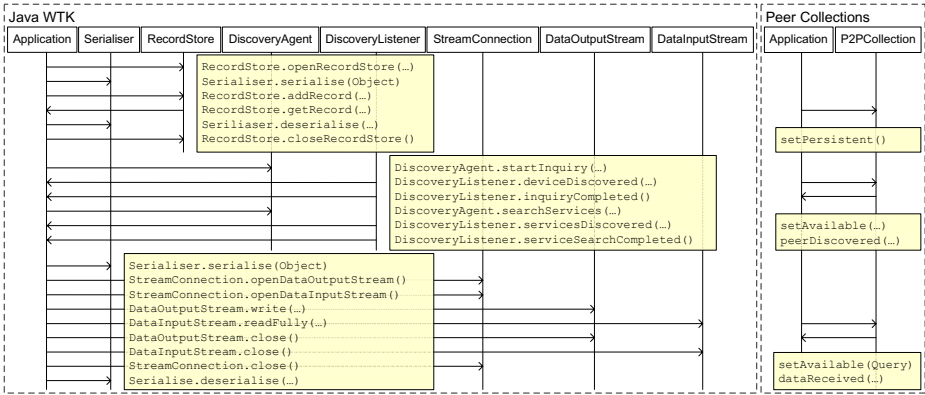


Fig. 7. Comparison of using Java WTK (left) and our framework (right)

using our framework, the application developer does not have to bother with low-level socket-based connectivity and data transmission in terms of serialisation and deserialisation. The fact that the developer can work at the level of the application model by deciding how data collections should be shared presents a significant contribution to the development of mobile social applications.

## 7 Conclusions

We have motivated a set of novel requirements introduced by the emerging class of mobile social applications. Due to the limitations and heterogeneity of mobile phone development platforms, we have proposed to address these requirements with a framework based on a notion of peer collections. A peer collection encapsulates data management, data sharing and vicinity awareness, all of which are recurring issues in the development of mobile social applications. Further, through the provision of both declarative queries and events associated with peer collections, our framework decouples application and collaboration logic. The merits of our approach have been shown by comparing the implementation of a collaborative filtering application based on our framework with one based on an existing mobile phone platform.

We are currently experimenting with extending our framework to accommodate further mobile social application requirements. Vicinity awareness is currently provided in terms of a real-time representation. One extension is to keep track of peers previously encountered which enables applications to take into account frequencies of encounters and to recognise social contexts. We are also considering support for access control by providing the possibility of associating multiple selection queries with a P2P collection to represent user groups or social contexts.

## References

1. Eagle, N., Pentland, A.S.: Reality mining: sensing complex social systems. *Personal Ubiquitous Comput.* 10(4) (2006)
2. Nicolai, T., Yoneki, E., Behrens, N., Kenn, H.: Exploring social context with the wireless rope. In: *OTM 2006 Workshops* (2006)
3. Borcea, C., Gupta, A., Kalra, A., Jones, Q., Iftode, L.: The mobsoc middleware for mobile social computing: challenges, design, and early experiences. In: *Proc. 1st Intl. Conf. on MOBILE Wireless MiddleWARE, Operating Systems, and Applications* (2007)
4. Eagle, N., Pentland, A.: Social serendipity: mobilizing social software. *Pervasive Computing, IEEE* 4(2) (2005)
5. Counts, S., Geraci, J.: Incorporating Physical Co-presence at Events into Digital Social Networking. In: *Proc. CHI 2005* (2005)
6. Lawrence, J., Payne, T.R., Roure, D.D.: Co-presence Communities: Using Pervasive Computing to Support Weak Social Networks. In: *Proc. Intl. Workshop on Distributed and Mobile Collaboration* (2006)
7. Srirama, S.N., Jarke, M., Prinz, W.: Mobile web services mediation framework. In: *Proc. 2nd Workshop on Middleware for Service Oriented Computing* (2007)
8. Traversat, B., Arora, A., Abdelaziz, M., Duigou, M., Haywood, C., Hugly, J.C., Pouyoul, E., Yeager, B.: Project JXTA 2.0 Super-Peer Virtual Network. Technical report, Sun Microsystems, Inc. (2003)
9. Aberer, K., Alima, L.O., Ghodsi, A., Girdzijauskas, S., Haridi, S., Hauswirth, M.: The Essence of P2P: A Reference Architecture for Overlay Networks. In: *Proc. 5th IEEE Intl. Conf. on Peer-to-Peer Computing* (2005)
10. Wang, A.I., Bjornsgard, T., Saxlund, K.: Peer2Me - Rapid Application Framework for Mobile Peer-to-Peer Applications. In: *Intl. Symp. on Collaborative Technologies and Systems* (2007)
11. Kortuem, G., Schneider, J., Preuitt, D., Thompson, T.G., Fickas, S., Segall, Z.: When Peer-to-Peer comes Face-to-Face: Collaborative Peer-to-Peer Computing in Mobile Ad hoc Networks. In: *Proc. Intl. Conf. on Peer-to-Peer Computing* (2001)
12. Rowstron, A.I.T., Druschel, P.: Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In: Guerraoui, R. (ed.) *Middleware 2001*. LNCS, vol. 2218, pp. 329–350. Springer, Heidelberg (2001)
13. Tatarinov, I., Ives, Z., Madhavan, J., Halevy, A., Suci, D., Dalvi, N., Dong, X.L., Kadiyska, Y., Miklau, G., Mork, P.: The piazza peer data management project. *SIGMOD Rec.* 32(3) (2003)
14. Ooi, B.C., Tan, K.L., Zhou, A., Goh, C.H., Li, Y., Liau, C.Y., Ling, B., Ng, W.S., Shu, Y., Wang, X., Zhang, M.: Peerdb: peering into personal databases. In: *Proc. ACM SIGMOD Intl. Conf. on Management of Data* (2003)
15. Rodríguez-Gianolli, P., Kementsietsidis, A., Garzetti, M., Kiringa, I., Jiang, L., Masud, M., Miller, R.J., Mylopoulos, J.: Data sharing in the hyperion peer database system. In: *Proc. 31st VLDB Conf.* (2005)
16. Aberer, K., Datta, A., Hauswirth, M., Schmidt, R.: Indexing data-oriented overlay networks. In: *Proc. 31st VLDB Conf.* (2005)
17. Cudré-Mauroux, P., Agarwal, S., Budura, A., Haghani, P., Aberer, K.: Self-organizing schema mappings in the gridvine peer data management system. In: *Proc. 33rd VLDB Conf.* (2007)
18. de Spindler, A., Norrie, M.C., Grossniklaus, M.: Recommendation based on Opportunistic Information Sharing between Tourists. *Information Technology & Tourism (to appear)*

# Evolving Services from a Contractual Perspective<sup>\*</sup>

Vasilios Andrikopoulos<sup>1</sup>, Salima Benbernou<sup>2</sup>, and Mike P. Papazoglou<sup>1</sup>

<sup>1</sup> ERISS, Tilburg University, Netherlands

<sup>2</sup> LIRIS, Université de Lyon 1, France

{v.andrikopoulos,mikep}@uvt.nl, sbenbern@liris.univ-lyon1.fr

**Abstract.** In an environment of constant change, driven by competition and innovation, a service can rarely remain stable - especially when it depends on other services to fulfill its functionality. However, uncontrolled changes can easily break the existing relationships between a service and its environment (its customers and providers). In this paper we present an approach that allows for the controlled evolution of a service by leveraging the loosely-coupled nature of the SOA paradigm. More specifically, we formalize the notion of contracts between interacting services that enable their independent evolution and we investigate under which criteria can changes to a contract-bound service, or even to the contract itself, be transparent to the environment of the service.

**Keywords:** service evolution, service contracts, compatibility, contract invariance, contract evolution.

## 1 Introduction

A number of serious challenges like mergers and acquisitions, outsourcing possibilities, rapid growth, regulatory compliance needs and intense competitive pressures require changes at the enterprise level and lead to a continuous business process redesign and improvement effort. Service changes that are required by this effort however must be applied in a controlled fashion so as to minimize inconsistencies and disruptions by guaranteeing seamless interoperation of business processes that may cross enterprise boundaries.

In general, we can classify service changes depending on their direct and side effects [1] in *shallow*, where the change effects are localized to the service or are strictly restricted to the clients of that service, and *deep*, that are cascading types of changes which extend beyond the clients of a service, and possibly to its entire value-chain, i.e., to clients of the service clients such as outsourcers or suppliers. Shallow changes characterize both singular services and business processes and require a structured approach and robust versioning strategy to

---

\* The research leading to these results has received funding from the European Community's Seventh Framework Programme FP7/2007-2013 under grant agreement 215483 (S-Cube).

support multiple versions of services and business protocols. Deep changes on the other hand are more intricate and require the assistance of a *change-oriented service life cycle* where the objective is to allow services to predict and respond appropriately to changes as they occur [1]. Due to the complexity and scope of deep changes this paper discusses only shallow changes and more specifically changes to the Structural layer elements of the Service Specification Reference Model introduced in [2], i.e., to the message content, operations, interfaces, and message exchange patterns (MEPs), roughly corresponding to WSDL artifacts.

The setting discussed has a number of similarities with the fields of *evolution transparency* and *interoperability preservation* that have been discussed in different forms in [3] and [4] (among others), and essentially boil down to preventing incompatibility between interoperating (interacting) services. These works though depend mainly on adaptation mechanisms to maintain interoperability, and adaptation approaches are by definition *a posteriori* interventions focusing on incompatibility identification and resolution by modification of a service. In that sense, the adaptation process can not discern between shallow and deep changes and is unable to prevent the propagation of changes throughout the value chain, since the modification of a service may have unforeseen consequences to the parties that interact with it. For that reason we are focusing on identifying under which conditions changes to a service are shallow and discuss an *a priori* approach that aims to prevent or at least predict and confine the necessity for adaptation.

The goal of this work is therefore to allow the *independent* evolution of loosely coupled interacting parties in a *transparent* manner so as to preserve their interoperability. In this context, the parties involved in an interaction can either be services, or services and client (service-based) applications. We only consider bilateral interactions, and for each such interaction we distinguish two roles: that of the *producer* and that of the *consumer*. It must be kept under consideration that the role of a service, unlike that of an application that always acts as a consumer, can vary depending on the interaction. An aggregate service for example plays both roles: that of the producer for its clients, and that of the consumer when it interacts with the aggregated services to compose a result. To achieve meaningful interoperability in this context, service clients and providers must come to a mutual agreement, a *contract* of sorts between them [5]. A contract of this type formalizes the details of a service in a way that meets the mutual understanding and expectations of both service provider and service client. Building around this idea, we are presenting mechanisms to effectively deal with the evolution of the structural aspect of both parties, while preserving interoperability despite the changes that may affect them. After we lay down this foundation we discuss the evolution of interactions and contracts themselves.

The rest of the paper is organized as follows: section 2 presents a notation for service description that leverages the decoupling of service providers and clients through the introduction of the contract construct (section 3). Section 4 shows how the introduced notions can be used to control the evolution of the interacting parties while maintaining a high degree of flexibility. Section 5



will briefly present related works, and section 6 discusses conclusions and future work. To facilitate the conversation, we are using the simple service described below as a point of reference:

*Example 1 (Running Example).* Let's assume the case of a very simple inventory service that checks for the availability of an item and responds that the purchase order can be fulfilled or issues a fault stating that the order cannot be completed. The WSDL file of this service is shown in listing 1.

```

...
<types>
  <xsd:schema targetNamespace="http://e-grocery.com/InventoryService">
    <xsd:complexType name="inventoryItem">
      <xsd:sequence>
        <xsd:element name="orderId" type="xsd:string"/>
        <xsd:element name="itemID" type="xsd:string"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:schema>
</types>
<message name="InventoryRequest">
  <part name="inventoryItem" type="tns:inventoryItem"/>
</message>
<message name="InventoryConfirmation">
  <part name="confirmationMessage" type="xsd:string"/>
</message>
<message name="InventoryFault">
  <part name="faultMessage" type="xsd:string"/>
</message>
<portType name="InventoryServicePortType">
  <operation name="checkInventory">
    <input name="item" message="tns:InventoryRequest"/>
    <output name="confirmation" message="tns:InventoryConfirmation"/>
    <fault name="fault" message="tns:InventoryFault"/>
  </operation>
</portType>
...

```

Listing 1. Inventory Service WSDL specification

## 2 Service Specifications

The WSDL description of the inventory service in listing 1 is far from complete in describing the structural aspect of the service. In specific, apart from providing an unambiguous schema for the service interfaces (the *signature* of the service) to be used by its clients, it lacks completely in providing *a)* any information on the services used by the service itself to fulfill its functionality (if any), and *b)* the means to connect the information *required* and *provided* by its signatures

with that of the signatures of the other services it is using. It is therefore not suitable for describing the interaction of the service with its environment and has to be replaced by a declarative specification that fulfills this role. [2] provides a more exhaustive discussion on the structure and content of such a service specification scheme. For the purposes of this work, we will only define the following constructs:

**Definition 1 (Element).** An element  $e$  of a service  $s$  is defined as a tuple  $(a_1, a_2, \dots, a_n)$ , the set of attributes that characterize the element.  $a_i$  is either an atomic attribute or another element  $e_i$  of the service.

For example, `InventoryRequest`, `checkInventory`, and the rest of the WSDL constructs in listing 1 can be represented as elements  $a_1 = (\text{inventoryItem})$ ,  $a_2 = (\text{item}, \text{confirmation}, \text{fault})$ , etc.

**Definition 2 (Type extension).** The specification  $E$  of a service is defined by the set  $E = \{e_i, i \geq 1\}$  of its elements. We associate to  $E$  the reflexive and transitive relation type extension  $\leq$  on elements  $(E, \leq)$  defined as:  $e \leq e' \Leftrightarrow \{a_1, \dots, a_n\} \subseteq \{a'_1, \dots, a'_m\}, m \geq n \wedge a_i \leq a'_j, 1 \leq i \leq n, 1 \leq j \leq m$ .

If for example  $\exists a'_1 = (\text{orderID}, \text{itemID}, \text{comment})$  (as in listing 1 in section 3.1) then:  $(\text{orderID}, \text{itemID}) \subseteq (\text{orderID}, \text{itemID}, \text{comment}) \Rightarrow a_1 \leq a'_1$ , i.e., the new `(inventoryItem)` is a type extension of the old one.

As discussed in the previous section, the approach discussed by this work assumes that a) both producer and consumer are in the general case services, and therefore use the same notation to describe their specifications, and b) a producer in one interaction can also act as the consumer for another interaction. This latter interaction may or may not be related to the producer’s function in the former. Beyond this generic case, the same paradigm can also be applied to “simpler” cases: autonomous services that implement all of their offered functionalities without using other services act only as producers. Non-service clients (e.g., GUI-supported applications) can be perceived as special cases of exclusive consumers.

We define two orthogonal views on  $E$  (see figure 1a): the expositions/expectations view and the required/provided view:

These views provide us with reference points in disambiguating the roles and functions of elements in a service specification like listing 1. More specifically:

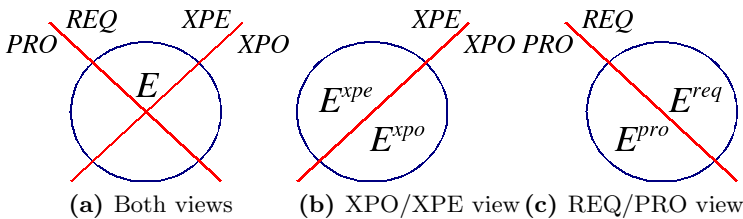


Fig. 1. Views on the service specification

## 2.1 Exposition/Expectation View

This view (figure 1b) classifies the elements within a service specification with respect to whether they are offered as an interface to the environment or they are “imported” into the service specification, referring to interface elements of other services. In the former case, the service acts as a producer; in the latter as a consumer. Elements of a service specification can therefore fall into one of the following categories:

- Exposition  $E^{xpo}$ : the set of elements that describe the offered functionality of the service.
- Expectation  $E^{xpe}$ : the set of elements describing the perceived offering of functionality to the service by other services.

The WSDL file of the inventory service for example in listing 1 contains the information on how to access the elements that constitute the inventory service and what information is exchanged while accessing it. From the perspective of the producer of the service, this file specifies what the producer will offer to the service customers: if the `checkInventory` operation is invoked using the `InventoryServicePortType` and the message payload defined, the generated result or fault message will be a simple string. The elements of the file are in that sense in the exposition subset of the service producer specification.

On the other hand, when a consumer of this service builds and/or uses an application that incorporates an invocation of this service, the consumer refers to what it *perceives* to be a set of elements that allow it to access the service. To put it simply, the client is built on the premise of a particular specification of the provided interface, being bound for example to the service of listing 1. These elements are therefore contained in the expectation subset of the consumer specification. What becomes apparent from this is that the same elements can either be expositions or expectations; it only depends on the adopted viewpoint.

Ideally, this perceived specification and the actual specification of the provided service are the same - and that is so far the fundamental assumption in service interactions. But changes to either side, as we will discuss in the following sections, could lead to inconsistencies - in other terms incompatibilities - between those two.

## 2.2 Required/Provided View

The division enforced by this view (figure 1c) is much more straightforward: it provides the means to cleanly separate input from output in a service specification (irrespective of whether it acts as a producer or a consumer). More specifically:

- Required  $E^{req}$ : contains the input-type elements of the service specification.
- Provided  $E^{pro}$ : contains the output-type elements.

`InventoryRequest` for example is clearly a required element for the producer: it is the input message type for the service. At the same time it is a provided

element for the consumer since it has to be provided to the producer in order to use the respective operation. `InventoryConfirmation` and `InventoryFault` are respectively provided elements for the producer - they are produced as output by the service in one way (normal result) or another (fault message) - and required elements for the consumer (input to it).

### 2.3 Combining the Views

Since the two views are orthogonal, they can be used in conjunction to describe the elements of a service specification:  $E^{xpo} \cup E^{xpe} = E^{req} \cup E^{pro} = E$  (figure 1a).

*Example 2.* Figure 2 shows how an invocation of the inventory service of listing 1 by a Web services client can be described using the classification presented. Due to the request-response messaging pattern of the `checkInventory` operation, the interaction between the service and its client is broken down into two phases: in the first phase, the consumer (client) is using the expectation element (1) to invoke the exposition element (2) of the producer (service). Since (1) is an output for the consumer it belongs to the  $E_{consumer}^{pro}$  set, and (2) is in the  $E_{producer}^{req}$  as the input of the service. The situation is inverted for the second phase, where the producer uses (3) to call back (4) in the consumer side.

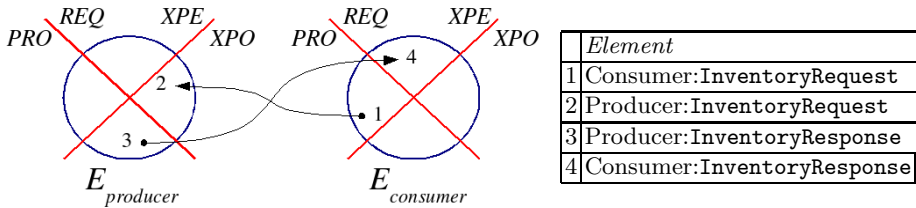


Fig. 2. Service Interaction

## 3 Contracts

This section builds on the notation and classification presented in the previous section to discuss the interaction of parties in a loosely-coupled environment and introduce the notion of *contracts* as the means to leverage the decoupling between producer and consumer.

By the term contract we do not refer to the legal documents that describe a binding agreement, but we use the term in the same manner as the (software) contracts in the Eiffel language [6]. The contracts in this context are documents that record the *benefits* expected by each party from their interaction, and the *obligations* that each party is prepared to carry out in order to obtain these promised benefits. In that sense, the contract protects both sides by clearly defining what is the acceptable contribution and result for a task described by the contract. Our approach applies the same paradigm on services specifications,

using the different views discussed above to distinguish between those benefits and obligations, depending on the role that the service plays.

In specific, there is an important distinction in the way that the producer and the consumer of a service are perceiving a service specification document: the producer promises to offer the service in the manner specified in it (the expositions set), and the consumer accepts this promise and builds a client for it based on this promise (the expectations set). In most contemporary SOA implementations, by using for example Web services technologies, this fundamental difference is bridged by accepting one perspective, that of the producer, and shifting the consumer side perspective accordingly. But in this case the consumer has to adopt any changes and assumptions that are done by the producer. Failure to comply with the producer means that the consumer is unable to use the offered functionality, which explains why producer updates typically fail on the client side.

In order to amend this situation, we propose to use a construct (the contract) that bridges the two perspectives and allows for mapping from and to it by either party. This contract is nothing more than an intermediary specification, containing a set of commonly agreed elements specified in a party-independent way. By providing a neutral mapping procedure from each party to the contract we minimize the producer/consumer coupling. Furthermore, given a contract, we allow for reasoning by each party *in isolation*, enforcing the separation of concerns and responsibilities in service design and operation. In the following we formally define the contract construct and describe how to formulate a contract between two parties.

### 3.1 Contract Definition

In principle only a part of the offered service functionalities may be used by a specific client; on the other hand, a client may depend on a number of disparate services in order to achieve its goals. Thus we need a way to identify and isolate the parts of the interacting parties that actually contribute to the interaction. For that purpose we will denote with  $P \subseteq E_{producer}^{xpo}$  and  $C \subseteq E_{consumer}^{xpe}$  the subsets from the producer and consumer specifications respectively that participate in the interaction.

Following on we define a binding function  $\vartheta$  that reasons *horizontally* between the elements of parties  $P$  and  $C$ :

**Definition 3 (Service Matching).** *A service matching is a binding function defined as  $\vartheta : P \times C \rightarrow U, U = P \cup C$  such that*

$$\vartheta(x, y) = \{z \in U \mid \left\{ \begin{array}{l} x \leq z \leq y, x \in P^{req}, y \in C^{pro} \\ y \leq z \leq x, x \in P^{pro}, y \in C^{req} \end{array} \right\} \} \quad (1)$$

*Example 3.* Let's assume that  $P$  contains the elements of listing [1](#) and let's denote by  $x \in P^{req}$  the `InventoryItem` element:  $x = (a_1, a_2)$ ,  $a_1 = (\text{orderID})$  and  $a_2 = (\text{itemID})$ . A consumer of this service that is bound to listing [1](#) uses all elements as they are defined in the listing (that is:  $P \equiv C$ ) and therefore

$\exists y \in C^{pro}/y = x \Rightarrow \vartheta(x, y) = z = (a_1, a_2)$ . This reasoning holds also for the rest of the elements of  $P$  and  $C$  and the service matching is in that case trivial.

Now consider the case of another consumer  $C'$  that is bound to listing 2 that differs from listing 1 in the definition of `InventoryItem`:  $y' = (a_1, a_2, a_3)$ ,  $a_3 = \text{(comment)}$  to allow for attaching notes to items. By its definition  $\vartheta(x, y') = z'$  returns two possible values:  $z' = (a_1, a_2)$  or  $z' = (a_1, a_2, a_3)$ . By selecting the first value (reflecting the assumption that  $P$  can *ignore* this extra argument in the requests of  $C'$ ) we observe that the previous service matching between  $P$  and  $C$  persists for  $P$  and  $C'$  despite the changes in consumer  $C$ . The actual selection of the binding function value during the contract formulation is a matter of policy (see following section for a further discussion on this subject).

```

...
<xsd:complexType name="inventoryItem">
  <xsd:sequence>
    <xsd:element name="orderId" type="xsd:string"/>
    <xsd:element name="itemID" type="xsd:string"/>
    <xsd:element name="comment" type="xsd:string" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>
...

```

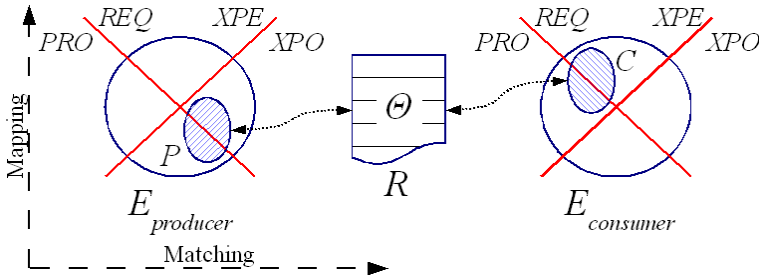
**Listing 2.** Alternative inventory item definition

Binding function  $\vartheta$  is acting in the same manner as a *schema matching* function would. Schema matching aims at identifying semantic correspondences between elements of two schemas, e.g., database schemas, ontologies, and XML message formats [7]. It is necessary in many database applications, such as integration of web data sources, data warehouse loading and XML message mapping. In most systems, schema matching is manual or semi-automatic; a time-consuming, tedious, and error-prone process which becomes increasingly impractical with a higher number of schemas and data sources to be dealt with. In our case though, the matching function relies on the type extension relation to automatically identify elements on either party that are semantically related to each other according to their respective schemata.

Based on the matching function  $\vartheta$  we can define the *Contract R* between two parties as a service mapping:

**Definition 4 (Service Mapping).** *A Service mapping is a Contract R defined by a triplet  $\langle P, C, \Theta \rangle$  between the two parties that is defined as their image under  $\vartheta$ , i.e.,  $\Theta = \{\vartheta(x, y) | x \in P, y \in C\}$ . The elements that comprise R are called the clauses of the contract.*

The mapping therefore consists of the results of the binding function for all possible pairs in the producer/consumer sets and is formulated by reasoning vertically through the parties. The contract that is produced by this mapping



**Fig. 3.** Producer/Consumer/Contract relation

identifies and represents the mutually agreed specification elements that will be used for the interaction of the parties. Figure 3 demonstrates the relation between  $P$ ,  $C$ , and  $R$  graphically.

*Example 4.* Following the previous example, the service mapping between  $P$  and  $C$  (consumer using listing 1) would consist of the contract  $R = \langle P, C, \Theta \rangle, P \equiv C \equiv \Theta$ .

For the service mapping between  $P$  and  $C'$  (consumer using listing 2) we are presented with two options: either we opt for  $z' = (a_1, a_2)$  and since the rest of the elements remain the same then  $\Theta' \equiv \Theta \Rightarrow R' \equiv R$ , or in the case of selecting  $z' = (a_1, a_2, a_3)$  then a new contract  $R' = \langle P, C', \Theta' \rangle$  has to be formulated.

### 3.2 Contract Formulation and Management

The definition of contract  $R$  between two parties as a service mapping  $\langle P, C, \Theta \rangle$  allows for a straightforward formulation of the contract: given the two parties' specifications  $P$  and  $C$ , each of which defines the elements through which the interaction is achieved,  $\Theta$  can be calculated directly by applying the matching function  $\vartheta$  to them. The issue of contract development therefore shifts in producing  $P$  and  $C$  from the service provider  $E_{producer}^{xpo}$  and client  $E_{consumer}^{xpe}$  specifications respectively.

Due to the fact that the service provider is unaware of the internal workings of the service client (represented by the  $E_{consumer}^{xpe}$  set) the process of contract formulation is consumer-driven; more specifically, the steps to be followed are:

1. The consumer decides on the functionality offered by the producer that will be used (if more than one is offered).
2. The set of elements from  $E_{producer}^{xpo}$  that fulfill this functionality (e.g., the port type and the associated structural elements) are identified.
3. The identified elements are either copied to the (initially empty)  $E_{consumer}^{xpe}$  set or the existing  $E_{consumer}^{xpe}$  set is used.
4. The image of  $P$  and  $C$  under  $\vartheta$  set is calculated. If the resulting set is empty then the image is attempted to be re-calculated using alternative values from  $\vartheta$  (or cancelled in case all possibilities have been exhausted); otherwise the contract  $R = \langle P, C, \Theta \rangle$  is produced.

5. The consumer submits the formulated contract  $R$  to the producer for posterity and begins interaction with producer.

The formulating, storing, and reasoning aspects of the proposed solution can be incorporated in the service governance infrastructure that supports each party. Since  $\vartheta$  may return one or more possible values, depending on the type extension 'distance' in the element definition between the producer and consumer specification, a minimum level of 'insight' on the consumer side is required in selecting the appropriate elements from the producer and in assigning values to the binding function  $\vartheta$ :

**Conservative** selection policies would opt for the values contributed by the consumer to the calculation of  $\vartheta$ , trying to protect the consumer from possible changes to the producer.

**Liberal** selection policies on the other hand would pick the values contributed by the producer and allow for the possibility of the consumer evolving in the future.

The type of policy to be followed is therefore largely a design and governance issue and has to be dealt as such. The solution presented assumes that producers and consumers have the means to formulate, exchange, store, and reason on the basis of contracts. In absence of these facilities from one or both parties the interaction between them reverts to the non contract-based modus operandi that, as we have discussed above, can not guarantee interoperability. The exchange of contracts requires the existence of a dedicated mechanism for this purpose that is not part of the service specification.

## 4 Contract-Controlled Service Evolution

The previous section discussed how to leverage the loose coupling of the producer and the consumer by means of the contract construct. The following section discuss how this design solution enables evolutionary transparency that preserves (under certain conditions) the producer/consumer interoperability.

In the initial 'static' state of two interoperating parties  $P$  and  $C$ , and after a contract  $R = \langle P, C, \Theta \rangle$  has been formulated and accepted between them, it holds in general that  $P \equiv \Theta \equiv C$ . For example, when a simple client is using the service described in listing [1](#) it is safe to assume that due to the granularity of the service, the client will be using the one (and only) functionality provided by it. That in turn means that it will refer to all the elements contained in the WSDL file. Therefore,  $P \equiv C$  and by the definition of the contract construct,  $P \equiv \Theta \equiv C$ , as we have seen in the previous section.

But since either party can, or at least should be able to evolve independently of the other, shifts from this state can occur. When changes for example occur to the producer then it may hold that  $P' \not\equiv \Theta \equiv C$ , or for the consumer side  $P \equiv \Theta \not\equiv C'$ , or both. These latter states reflect situations of incompatibility between producer and consumer and they have to be prevented from occurring in order



to avoid the occurrence of deep changes in the context of the interacting parties. The introduction of a contract between them allows us to reason about the contribution of each party to the interaction without directly affecting the other party, ensuring that each party is able to evolve *independently* but *transparently*, that is without requiring modifications, to each other.

For that purpose we will distinguish *shallow* changes occurring to a party in two categories: those that respect the *contractual invariance* and those that require *contractual evolution*. Changes to a party that fall in the former category do not affect the existing contract between the parties. Changes in the latter category require modifications to the contract but nevertheless do not require changes to the other party.

#### 4.1 Contract Invariance

Taking advantage of the ability to reason exclusively on one party given an existing contract, without the need for the other party to participate in this reasoning, exemplifies the notion of independence in evolution. In order to show how this is accomplished we will first formally define what it means for a (modified) party specification to respect, or to be *compliant* with a contract:

**Definition 5 (Compliance to Contract).** *A version of a party, e.g. version  $P'$  of producer  $P$ , is said to be compliant with respect to an existing contract  $R = \langle P, C, \Theta \rangle$  with a consumer  $C$  denoted by  $P' \models_R C$  iff*

$$\forall z \in \Theta / \exists x' \in P', \vartheta(x', y) = z, y \in C \quad (2)$$

**Corollary 1.** *Consequently,  $P'$  violates  $R$ , and we write  $P' \not\models_R C$ , iff  $\exists z \in \Theta / \forall x' \in P', \vartheta(x', y) \neq z, y \in C$ .*

The definition above allows for a simple algorithm to check for the compliance of a new version of a party in the producer-consumer relationship: as long as there is a mapping produced by  $\vartheta$  to *all* clauses of the contract from the elements of the new specification, the two versions are equivalent or *compatible* with respect to the contract - or more formally:

#### Definition 6 (Compatibility w.r.t. existing Contract)

1. *Given a party, e.g. consumer  $C$ , then two versions of the other party,  $P$  and  $P'$ , are called compatible w.r.t. a contract  $R$  denoted by  $P \mapsto_R P'$  iff they are both compliant to  $R$ :  $P \models_R C \wedge P' \models_R C$ .*
2. *Two versions of a party  $S$  and  $S'$  are called fully compatible iff they are compatible for all contracts  $R_i, i \geq 1$  that they participate in, either as producers or consumers:  $S \mapsto_{R_i} S' \forall R_i$ .*

*Example 5.* Consider the modifications applied to the service specification as depicted in listing 3. Let's assume that these changes are applied to  $P$ ; in that case  $P'$  is compatible with  $P$ , since they are both compliant to the same contract  $R$ . To

prove that, we start with the observation that element  $x = (\text{InventoryConfirmation})$  in listing [1](#) is in the  $P^{pro}$  set, and therefore contributes to the second leg of the binding function [\(1\)](#) which means that

$$\exists y \in C^{req}, z \in \Theta / y \leq z \leq x. \quad (3)$$

Let's denote with  $x'$  the changed element from listing [4.1](#). It holds that  $x \leq x'$  and in conjunction with [\(3\)](#) we get:  $\exists y \in C^{req}, z \in R / y \leq z \leq x'$ . Thus,  $\vartheta(x', y) = \vartheta(x, y)$ , and since the rest of the matchings remain unchanged, by [\(2\)](#) we can deduce that  $P' \vDash_R C$ .

If listing 3 though is depicting changes to the consumer side, then by the same reasoning we can easily prove that  $C$  and  $C'$  are not compatible, since  $P \not\vDash_R C'$ .

```

...
<message name="InventoryConfirmation">
  <part name="confirmationMessage" type="xsd:string"/>
  <part name="confirmationDate" type="xsd:date"/>
</message>
...

```

**Listing 3.** New inventory Service WSDL specification

## 4.2 Contract Evolution

The previous section discussed the criteria under which changes to one party can leave the contract between them intact, essentially ensuring that these changes are shallow. This does not necessarily mean that all changes that do not respect this criteria are deep. The existing interaction between the parties can be preserved in certain cases despite the necessity to modify the contract due to changes to one or both of the parties involved, defined as *backward* and *forward* compatibility preserving cases:

**Definition 7 (Backward Compatibility).** *Two contracts  $R = \langle P, \Theta, R \rangle$  and  $R' = \langle P, \Theta', C' \rangle$  are called backward compatible and we write  $R \mapsto_b R'$  iff  $\forall x \in P / \exists z' \in \Theta', \exists y' \in C', z' = \vartheta(x, y')$ .*

In that case changes to the consumer side leave the producer unaffected. The (new) consumer will use the producer in the same manner as the old consumer did.

**Definition 8 (Forward Compatibility).** *Two contracts  $R = \langle P, \Theta, R \rangle$  and  $R' = \langle P', \Theta', C' \rangle$  are called forward compatible and we write  $R \mapsto_f R'$  iff  $\forall y \in C / \exists z' \in \Theta', \exists x' \in P', z' = \vartheta(x', y)$ .*

Forward compatibility therefore allows for the seamless interoperability of the new producer with the old consumer without the former party to have to be modified in any way.

It must be noted that these definitions following [1](#) are using the vantage point of the *consumer* to discuss changes: a change to a contract is backwards

compatible if it allows the consumer to accept input from older devices (versions of the producer). Similarly, a forwards compatible contract means that the consumer can accept input from newer versions of the producer. Consider for example the discussion in section 3.1 on the possibilities for service matching and mapping: if we choose to create a new contract  $R'$  then it can be easily shown that this contract is backward compatible to  $R$  and therefore the new consumer  $C'$  can still use the old producer  $P$ .

Furthermore, by combining the two definitions we can define when two contracts are compatible:

**Definition 9 (Contract Compatibility).** *Two contracts  $R = \langle P, \Theta, R \rangle$  and  $R' = \langle P', \Theta', C' \rangle$  are called compatible and we write  $R \mapsto R'$  iff they are both backward and forward compatible:  $R \mapsto_b R' \wedge R \mapsto_f R'$ .*

Contrary to the case of contractual invariance, evolution of the contract itself requires of the parties to exchange a new contract and replace the old contract with the new one. This creates an additional communication overhead that nevertheless has to be weighted against the cost of possible inconsistencies in the current and future interactions of the parties due to the discrepancy between the contract versions.

## 5 Related Work

The term ‘contract’ and the approach of introducing contracts in software components design stems from the Eiffel language [6], [8]; the core ideas of that work have greatly influenced our approach.

There are a number of works discussing the introduction of adapters between interacting parties to ensure their interoperability: [9], [10], [3], [11], [12], and [4] among others. Of specific interest to us is the work in [13], since they also make a clear distinction between the service producer and service consumer interfaces and protocols and use mappings to bridge them. Then they proceed to describe how to semi-automatically identify and resolve incompatibilities (mismatches) on interface and protocol level. Our approach extends this idea of separating producer and consumer specifications, but discusses how to avoid mismatches altogether instead of resolving them.

Furthermore, the W3C Technical Architecture Group has published an editorial draft on the extensibility and versioning of XML-based languages [14]. Their findings build on a number of previously developed theories and techniques like [15], [16] and draw lessons from the HTML and HTTP standards. They show how compatibility can be defined in terms of set theory, using super-sets and sub-sets to ensure compatibility. Our approach follows a similar way in dealing with the issue of compatibility, but instead of allowing the direct producer/consumer interaction, it introduces the contract as an intermediary to further decouple them.

The notion of service mapping comes from the field of schema evolution, i.e., the ability to change deployed schemas - metadata structures formally describing

complex artifacts such as databases [17, 18, 7], messages, application programs or workflows. Typical schemas thus include relational or object-oriented (OO) database schemas, conceptual ER or UML models, ontologies, XML schemas, software interfaces and workflow specifications. Effective support for schema evolution is challenging since schema changes may have to be propagated, correctly and efficiently, to instance data, views, applications and other dependent system components. Our approach provides the means to identify schema changes that do not result in propagation of changes.

## 6 Conclusions and Future Work

In the work presented in the previous sections we discuss an approach that allows for transparency in the evolution of a service as viewed from the perspective of both clients and providers, in the context of the loosely-coupled nature of the SOA paradigm. For that purpose we introduce the contract construct as the means to leverage the decoupling of the interacting parties. We present a contract constructing function that bridges the gap between service matching and service mapping. Following on, we build on contractual invariance and contractual evolution to show how to effectively deal with shallow changes to the service provider and client interaction - without the need for adaptation which may lead in turn to deep changes.

There are of course a number of issues that are briefly discussed by our approach that we plan to work on in the future. The matter of management of the contracts and its relationship to service governance mechanisms is the most important issue at hand, since it can provide further insights on the proposed solution. Furthermore, the binding function  $\vartheta$  value selection policy has to be further investigated. Using a static selection policy can be very restricting; a balancing mechanism for example can be applied for a more dynamic approach, expressed for example by negotiation between the parties in deciding the terms of the contract. Such a negotiation process during the formulation of the contract could result in the offering of additional or more specialized functionalities by the producer and could add a feedback loop to the presented algorithm for contract formulation. A promising direction when it comes to the implementation of our approach is to see whether it is possible to use techniques like the mapping constraints and tools developed by the schema mapping community like ToMAS [7].

The preservation of interoperability enforced by our approach is only the foundation in discussing the evolution of the interaction of parties. Following on, we plan to investigate how we can build on this work to deal with deep changes and the propagation mechanisms that run through them. On the other hand, another of the limitations of this work, the focus on the structural aspect of the service specification has also to be investigated, and examined if it is possible to apply the same approach to business protocols and policy-related constraints.

## References

1. Papazoglou, M.P.: The challenges of service evolution. In: Bellahsène, Z., Léonard, M. (eds.) CAiSE 2008. LNCS, vol. 5074, pp. 1–15. Springer, Heidelberg (2008)
2. Andrikopoulos, V., Benbernou, S., Papazoglou, M.P.: Managing the evolution of service specifications. In: Bellahsène, Z., Léonard, M. (eds.) CAiSE 2008. LNCS, vol. 5074, pp. 359–374. Springer, Heidelberg (2008)
3. Ponnekanti, S.R., Fox, A.: Interoperability among independently evolving web services, Toronto, Canada, pp. 331–351. Springer, New York (2004)
4. Senivongse, T.: Enabling flexible cross-version interoperability for distributed services, p. 201. IEEE Computer Society, Los Alamitos (1999)
5. Papazoglou, M.P.: Web Service: Principles and Technology. Prentice Hall/Addison-Wesley (E) (2007)
6. Meyer, B.: Applying ”design by contract”. *Computer* 25, 40–51 (1992)
7. Velegarakis, Y., Miller, R.J., Popa, L., Mylopoulos, J.: Tomas: A system for adapting mappings while schemas evolve. In: ICDE, p. 862 (2004)
8. Meyer, B.: Object-Oriented Software Construction, 2nd edn. Prentice Hall PTR, Upper Saddle River (1997)
9. Yellin, D.M., Strom, R.E.: Protocol specifications and component adaptors. *ACM Trans. Program. Lang. Syst.* 19, 292–333 (1997)
10. Evans, H., Dickman, P.: Drastic: A runtime architecture for evolving, distributed, persistent systems. In: Aksit, M., Matsuoka, S. (eds.) ECOOP 1997. LNCS, vol. 1241, pp. 243–275. Springer, Heidelberg (1997)
11. Benatallah, B., Casati, F., Grigori, D., Nezhad, H.R.M., Toumani, F.: Developing adapters for web services integration. In: Pastor, Ó., Falcão e Cunha, J. (eds.) CAiSE 2005. LNCS, vol. 3520, pp. 415–429. Springer, Heidelberg (2005)
12. Kongdenfha, W., Saint-Paul, R., Benatallah, B., Casati, F.: An aspect-oriented framework for service adaptation, 15–26 (2006)
13. Nezhad, H.R.M., Benatallah, B., Martens, A., Curbera, F., Casati, F.: Semi-automated adaptation of service interactions, Banff, Alberta, Canada, pp. 993–1002. ACM, New York (2007)
14. Orchard, D.(ed.): Extending and versioning languages: Terminology. W3C Technical Architecture Group (2007)
15. Orchard, D.: A theory of compatible versions. Published: xml.com article (2006)
16. Hoylen, S.(ed.): Xml schema versioning use cases, Published: W3C XML Schema Working Group Draft (2006)
17. Miller, R.J.: Retrospective on clio: Schema mapping and data exchange in practice. In: Description Logics (2007)
18. Fuxman, A., Hernández, M.A., Ho, C.T.H., Miller, R.J., Papotti, P., Popa, L.: Nested mappings: Schema mapping reloaded. In: VLDB, pp. 67–78 (2006)

# Efficient IR-Style Search over Web Services

Yanan Hao<sup>1</sup>, Jinli Cao<sup>2</sup>, and Yanchun Zhang<sup>1</sup>

<sup>1</sup> School of Engineering and Science, Victoria University  
P.O. Box 14428, Melbourne, VIC 8001, Australia  
{haoyn,yzhang}@csm.vu.edu.au

<sup>2</sup> Department of Computer Science and Computer Engineering, La Trobe University  
Bundoora, VIC 3086, Australia  
j.cao@latrobe.edu.au

**Abstract.** In service-based systems, one of the most important problems is how to discover desired web services. In this paper, we propose a novel IR-Style mechanism for discovering and ranking web services automatically. In particular, we introduce the notion of preference degree for web services and then we define service relevance and service importance as two desired properties for measuring the preference degree. Furthermore, various algorithms are given for computing the relevance and importance of services, respectively. Experimental results show the proposed IR-style search strategy is efficient and practical.

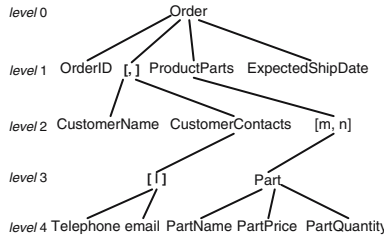
## 1 Introduction

Service-Oriented Computing (SOC) is emerging as a new paradigm for developing distributed applications. Web service discovery, among the most fundamental elements of SOC, provides a way to combine basic web services into value added services to satisfy user needs. As the number of web services and Service Oriented Computing applications increases, there is a growing need for mechanisms for discovering services efficiently.

Web service discovery introduces many new challenges. First, current web service discovery methods are mostly based on the UDDI-registry. To find a service in UDDI, a user needs to browse the relevant UDDI category to locate relevant web services. Considering a large amount of service entries, this process is time consuming and frustrating. So, we need an effective mechanism for automatic web service discovery. Second, a user's requirement for desired web services may not always be precise and a service discovery mechanism can potentially return a large number of results to satisfy the user's requirement, especially when a large service repository is available. Consequently, an important requirement for web service discovery is to rank the discovered results so that the most relevant services appear first. Finally, a good web service discovery mechanism should also be able to assist users in selecting relevant services and compose with them. For example, a typical strategy would allow users to see the services first she can use to start composing her application. Consider the three services shown in Fig. 1. The second and the third services can process the order information for

WS1: Web Service: CreateOrder	
Operation: <i>OrderBuilder</i>	
Input: UserID	DataType: <i>int</i>
Requirement	DataType: <i>ItemList</i>
<b>Output:</b> ProductsList	DataType: <i>BuyingOrder</i>
WS2: Web Service: ProcessPayment	
Operation: <i>CheckoutOrder</i>	
<b>Input:</b> UserProducts	DataType: <i>UserOrder</i>
Output: PaymentConfirmation	DataType: <i>bool</i>
WS3: Web Service: TransportOrder	
Operation: <i>ShippingOrder</i>	
<b>Input:</b> Cargo	DataType: <i>Order</i>
Output: PickupTime	DataType: <i>TimeLimit</i>

**Fig. 1.** Sample web-service operations



**Fig. 2.** XML schema tree of *Order* type

one transaction provided that a buyer’s order has been generated; whereas the first service provides the buyer’s order according to her requirement. Obviously, it is reasonable to say the first service is more important than the others since it contributes indispensable information for both of the other two services to be invoked. So, an ideal ranking strategy should put the first service on top. Also, as we can see, there are two links between the first and the other two services, in which the output of *CreateOrder* service, *BuyingOrder*, is also the input of both *ProcessPayment* service and *TransportOrder* service. This form of link potentially involves more web services and thus are particular useful in web service composition.

To address the problems above, in this paper we propose a novel IR-Style mechanism for discovering and ranking web services automatically, given a textual description of desired services. The contribution of the work reported here is summarized as follows:

1. We introduce the notion of *preference degree* for web services and then we define *service relevance* and *service importance* respectively as two desired properties for measuring the preference degree.

2. We design novel algorithms for computing the relevance and importance degree of services. Our algorithms take into account both textual and structural information of web services.
3. We define *service connectivity*, a novel metric to evaluate the importance of services. Meanwhile, we use our existing schema tree matching algorithm to measure the service connectivity.
4. We do various experiments to search for desired web services. Initial results show the proposed IR-style search strategy is efficient and practical.

The rest of this paper is organized as follows. Section 2 reviews the related work. Section 3 introduces the conception of preference degree for service ranking. Section 4 and section 5 present models and definitions for service relevance and service importance, followed by section 6, in which we present algorithms for ranking web services. In section 7 we describe our experimental evaluation. Section 8 gives some concluding remarks.

## 2 Related Work

Most approaches use text or structural matching to find similar web services for a given web service. The earlier technique tModel presents an abstract interface to enhance service matching process. But the tModel needs to be defined while authors publishing in UDDI [1]. In [2], the authors propose a SVD-Based algorithm to locate matched services for a given service. This algorithm uses characteristics of singular value decomposition to find relationships among services. But it only considers textual descriptions and can not reveal the semantic relationship between web services. Wang etc. [3] discover similar web services based on structure matching of data types in WSDL. The drawback is that simple structural matching may be invalid when two web-service operations have many similar substructures on data types. Woogle [4] develops a clustering algorithm to group names of parameters of web-service operations into semantically meaningful concepts. Then these concepts are used to measure similarity of web-service operations. However, it relies too much on names of parameters and does not deal with composition problem.

Recently, some methods have been proposed to annotate web services with additional semantic information. These annotations are used to match and compose services. For example, in [5] the authors extended DAML-S to support service specifications, including behavior specifications of operations; The Web Service Modeling Ontology (WSMO) [6] is a conceptual model for describing Web services semantically, and defines the four main aspects of semantic Web service, namely Ontologies, Web services, Goals and Mediators. However, most of existing web services currently use WSDL specifications, which do not contain semantics. Annotating the collection of services requires much effort, and it is infeasible in web service discovery. [7] formally defines a behavior model for web service by automata and logic formalisms. However, the behavior signature and query statements need to be constructed manually, which can be very hard for common users.



Some of our algorithms to be presented in this paper are related to keyword search in databases or Internet. For example, Discover [8,9] and DBXplore [10] operate on relational databases and facilitate information discovery on them by allowing users to issue keyword queries without any knowledge of the database schema; Google PageRank [11] uses the Internet's link structure as an indication of each web page's importance value. Also, our work is inspired by the work on XML schema. For instance, in [12] authors propose a syntactic approach to web service composition, given only the input-output schema types of web services available in their WSDL descriptions; [13] introduces the concept of schema summary and suggests importance and coverage as two relevant properties by which to judge the quality of a schema summary. In our previous work [14], we proposed a new schema matching algorithm for supporting web-service operations matching. The matching algorithm catches not only structures, but even better semantic information of schemas.

Other approaches include P2P-based service discovery [15], QoS-based discovery and ranking [16], service crawler [17]. Although these work provide ranking strategies, they overlook user's semantic preferences and can not identify the semantic relationship between services.

### 3 Desired Properties for Service Rank

Our goal is to find services in a more automatic and IR-style way, given a potentially partial specification of the desired service. We need an efficient mechanism to select the preferred services from available ones to satisfy the user's requirement. A natural idea is, firstly, to evaluate the user's preference for available services with respect to the textual service requirement, rank them according to the degree of preference, and then return the top services as search results. But, what makes a good service to the user? What does "preference degree" mean and how do we compute it? In [11], authors pointed that the final rank of a web page appearing in search results pages is determined by both the goodness of the match of the search terms on the page itself (*relevance* of the page) and this page's PageRank (*importance* of the page). Extending this idea to our context, we consider two factors for the user's preference degree for a service, in other words, the rank of the service. First, a good web service should be relevant to the user's requirement, i.e., to a certain extent, similar to the service requirement; Second, we should select services that are important. Intuitively, a service is important if it is employed by many other services in service composition; therefore, services that can be employed by as many as services are worth looking at.

Having seen what we consider to be desired properties for ranking web services, in the next two sections we will define the *service relevance* metric and *service importance* metric respectively, and then we calculate the user's preference degree for available services in reference to a textual description of desired web services provided by the user.

## 4 Service Relevance

Let  $q$  be a natural language description of the desired web services,  $S = \{s_1, s_2, \dots, s_k\}$  be the set of all available services published through UDDI; and  $D = \{D_1, D_2, \dots, D_k\}$  be a document collection containing WSDL specifications for all the services in  $S$ , where each WSDL document  $D_i$  corresponds to service  $s_i$ . Suppose there are  $N$  distinct words in  $D$  after a pre-processing step, including word stemming, removing stop words and expanding abbreviations and acronyms into the original forms. Applying the vector-space model to web services context, we describe each service  $s_i$  as an  $N$ -dimensional vector  $\vec{D}_i$  containing all terms in its specification  $D_i$ , denoted as  $\vec{D}_i = \{(t_1, w_{i1}), (t_2, w_{i2}), \dots, (t_N, w_{iN})\}$ , where each term is assigned a weight. A well-known weighting method is TF/IDF, namely the normalized *term frequency* (TF) and *inverse document-frequency* (IDF). Typically, the weight for each term  $t_j$  in document  $D_i$  is given by  $w_{ij} = tf_{ij} \times idf_j = tf_{ij} \times \log(k/n_j)$ , where  $k$  is the total number of available web services and  $n_j$  is the number of corresponding WSDL documents in which the term  $t_j$  appears. For more details, interested readers are referred to see [18].

Given the weighted vector  $\vec{q}$  for the user's description of desired services and the weighted vector  $\vec{D}_i$  for a service  $s_i$ 's WSDL specification document, we adopt the cosine distance metric to compare their similarity. Formally, the *service relevance* can be defined as follows:

**Definition 1. (Service Relevance).** *The relevance of a service  $s_i$ , denoted as  $R_{s_i}$ , with respect to the user's natural language description of desired web services, written as  $q$ , is defined as:*

$$R_{s_i} = \frac{\sum_{k=1}^N w_{ik} \times w'_{ik}}{\sqrt{\sum_{k=1}^N (w_{ik})^2} \cdot \sqrt{\sum_{k=1}^N (w'_{ik})^2}} \quad (1)$$

where  $w_{ik}$  is the weight for term  $k$  in  $\vec{s}_i$  and  $w'_{ik}$  is the weight for term  $k$  in  $\vec{q}$ .  $R_{s_i}$  ranges from 0 to 1. The higher score  $R_{s_i}$  is, the higher relevance service  $s_i$  has with respect to  $q$ , indicating a closer similarity between the user's description of request  $q$  and the available web service  $s_i$ .

## 5 Service Importance

A web service is in some way not different from a software component or module. Like in a software library, where different functions or modules have different level, not all services have equal importance in a web service repository. For example, consider the services in Fig. 1. Although all their WSDL descriptions contain terms provided by the user to search for desired services, most people would agree that service *CreateOrder* is more important than both the *ProcessPayment* service and the *TransportOrder* service, since the output of the first service, *BuyingOrder*, is also the input of the other two services, and thus a must

prerequisite for the other two services to be invoked. Based on this observation, we argue that a service can have great importance if there are many other services that employ it, or if there are some services that employ it and have a great importance. From this point of view, a service having low relevance may be more important than a service showing high relevance.

In order to reveal the nature of the importance of a service, we need to identify the relationships between the service and other services. Furthermore, we need an appropriate metric to capture how well a service can be used by other services. In the following parts, we describe models and algorithms to evaluate service importance; in particular, we show how to measure *connectivity* between two web-service operations based on schema matching, and how by connectivity we can achieve the importance of a web-service operation, which contributes to part of the importance of the service it belongs to.

## 5.1 Web-Service Operation Modeling

**Definition 2.** A web service is a triple  $ws = (TpSet, MsgSet, OpSet)$ , where  $TpSet$  is a set of data types;  $MsgSet$  is a set of messages (parameters) conforming to the data types defined in  $TpSet$ ;  $OpSet = \{op_i(input_i, output_i) | i = 1, 2, \dots, n\}$  is a set of operations, where  $input_i$  and  $output_i$  are parameters (messages) for exchanging data between web-service operations.

Fig. 1 has given three web-service operations used as examples in this paper. According to definition 1, a web service can be briefly described as a set of operations.

**Definition 3.** Each web-service operation is a multi-input-multi-output function of the form  $f : s_1, s_2, \dots, s_n \rightarrow t_1, t_2, \dots, t_m$ , where  $s_i$  and  $t_j$  are data types in according with XML schema specification. We call  $f$  a **dependency** and  $s_i/t_j$  a **dependency attribute**.

A dependency attribute can be a complex data type or a primitive data type. Complex data types, such as *BuyingOrder* and *UserOrder* in Fig. 1, define the structure, content, and semantics of parameters, whereas primitive data types, like *int* and *bool*, are typically too coarse to reflect semantic information. Since parameters usually can be regarded as data types, we can convert primitive data types to complex data types by replacing them with their corresponding parameters. For example, in Fig. 1 *bool* is converted into *PaymentConfirmation* type while *int* is converted into *UserID* type. Both *PaymentConfirmation* and *UserID* are considered as complex data types with semantics. Therefore, now each data type defined in a web-service operation can carry semantic meaning, according with XML schema specification at the same time.

## 5.2 Connectivity of Web Service Operations

As we can see, data types defined in web-service operations carry semantic information. Intuitively, we can consider two web-service operations, say  $A$  and

$B$ , connected if the output data types/attributes of  $A$  is the same as the input data types/attributes of  $B$ , so service  $B$  could directly employ service  $A$ 's output result and they can potentially collaborate in a user's web-service composition process. Obviously, however, requiring that  $A$ 's output and  $B$ 's input are the same so as to be connected is too strict and not practical in many cases. Generally, the connectivity relationship between two web-service operations can be defined formally as below:

**Definition 4.** (Web-service Operation Connectivity). *Given two web-service operations  $op_1 : s_1, s_2, \dots, s_n \rightarrow t_1, t_2, \dots, t_m$  and  $op_2 : u_1, u_2, \dots, u_l \rightarrow v_1, v_2, \dots, v_k$ , Let  $X = \{t_1, t_2, \dots, t_m\}$  and  $Y = \{u_1, u_2, \dots, u_l\}$ . The connectivity of  $op_1$  with respect to  $op_2$  can be measured as the similarity degree between  $X$  and  $Y$ , denoted as  $Con_{op_2 \rightarrow op_1} = sim(X, Y)$ .*

Service operation  $op_1$  is said to be *connected* to  $op_2$  If the connectivity degree  $Con_{op_2 \rightarrow op_1}$  is greater than some threshold value  $\lambda (0 < \lambda < 1)$ . If  $op_1$  has exactly the same output data type as  $op_2$ 's input, we will have  $Con_{op_2 \rightarrow op_1} = sim(X, Y) = 1$ , indicating the highest possible degree of connectivity of  $op_1$  regarding  $op_2$ . In this case, we say  $op_1$  is *well connected* to  $op_2$ . On the contrary, if the output of  $op_1$  is totally different from  $op_2$ 's input, we will have  $Con_{op_2 \rightarrow op_1} = sim(X, Y) = 0$ , indicating the lowest possible degree of connectivity of  $op_1$  regarding  $op_2$ .

As we have known, a data type used in web service operations presents a structure of schema tree, so  $X$  and  $Y$  are actually two groups of schema trees. Therefore, we can convert the problem of measuring connectivity between two web-service operations to the problem of schema tree matching. Section 6 will detail the algorithms for deriving the connectivity of a web service operation.

### 5.3 Importance of Web Service Operations

Based on the notion of connectivity, the web-service operation importance is given by an iterative equation as below, similar to the technique used in PageRank [11] algorithm:

**Definition 5.** (Web-service Operation Importance). *The importance of a web-service operation  $op$ , written as  $I_{op}$ , is calculated as the following iterative formula until convergence is reached:*

$$I_{op}^r = (1 - p) + p * \sum_{j \in F_{op_j}} Con_{op_j \rightarrow op} * I_{op_j}^{r-1} * 1/N_{op_j} \quad (2)$$

where  $Con_{op_j \rightarrow op}$  is the connectivity degree of  $op$  with respect to service operation  $op_j$ ;  $r$  denotes the number of iterations;  $F_{op_j}$  is the set of service operations connected by  $op_j$ ;  $N_{op_j} = |F_{op_j}|$  is the number of operations in  $F_{op_j}$ ; and  $0 \leq p \leq 1$  is a tuning parameter indicating how well the importance of a web service operation is affected by that of others. For all web-service operations,

the initial importance  $I_0$  is set to  $1/N$ , where  $N$  is the total number of available web-service operations. The computing process of the iterative equation above is shown by the *CompImp* algorithm in section 6.2.

## 6 Algorithm for Ranking Web Services

We now turn to the main focus of this paper, which is efficiently ranking web services. Recall that in section 3, two factors are considered for the rank of a service: service relevance and service importance. Since service relevance has been discussed in section 4, now the key issue remaining is how to compute the importance of services. We start with computing the connectivity of web-service operations by our schema tree matching strategy, then iteratively compute the importance of operations by *CompImp* algorithm. Finally, we combine these two factors to achieve the final rank scores for all web-service operations.

### 6.1 Computing Connectivity Using Schema Tree Matching

In this section, we use our existing schema tree matching algorithm [14] to measure the connectivity of a web-service operation, which is also a key step for evaluating its importance.

Our schema matching algorithm is based on tree edit distance [19,20]. However, traditional tree edit distance methods have three shortcomings:

1. They consider all tree edit operations to have same unit distance.
2. They neglect semantic information carried by the labels of nodes.
3. They do not consider the node difference between tag nodes and constraint nodes, and assign each edit operation unit cost.

To overcome these shortcomings, we presented a new cost model to compute the cost of tree edit operation, by which the tree edit distance of two schema trees is achieved. The new cost model integrates weights of nodes and semantic connections between nodes. Let  $T_1, T_2$  be two schema trees and let  $n$ ,  $node_1$  and  $node_2$  be tree nodes. Formally, the cost model is defined as

$$cost(\rho) = \begin{cases} weight(n)/W(T_1, T_2), & \text{if } \rho = insert(n) \\ weight(n)/W(T_1, T_2), & \text{if } \rho = delete(n) \\ \alpha \times wd(node_1, node_2) & \text{if } \rho \text{ relabels} \\ +\beta \times sd(node_1, node_2) & \text{node}_1 \text{ to } \text{node}_2 \end{cases} \quad (3)$$

where  $\rho$  indicates a tree edit operation.  $weight(n)$  shows the weight of node  $n$ , which is defined in definition 5.  $wd(node_1, node_2)$  and  $sd(node_1, node_2)$  give the weight and semantic difference of  $node_1$  and  $node_2$ , respectively.  $\alpha$  and  $\beta$  are weights of  $wd$  and  $sd$ , satisfying  $\alpha + \beta = 1$ .  $W(T_1, T_2)$  is defined as  $W(T_1, T_2) = weight(T_1) + weight(T_2)$ , where  $weight(T_i)$  is the sum of all node weights of tree  $T_i$  ( $i = 1, 2$ ).  $wd(node_1, node_2)$  is defined as

$$wd(node_1, node_2) = \frac{\|weight(node_1) - weight(node_2)\|}{W(T_1, T_2)} \quad (4)$$

where  $node_1 \in T_1$  and  $node_2 \in T_2$ .

In equation 3,  $weight(n)/W(T_1, T_2)$  explains the cost of inserting or deleting node  $n$ . For the relabel operation, both weight and semantics of  $node_1$  and  $node_2$  can be different, so we use the combination of weight and semantic difference as the relabel cost. All the costs are normalized by  $W(T_1, T_2)$ , i.e. the sum of all nodes weights of tree  $T_1$  and  $T_2$ .

**Definition 6.** Let  $level(n)$  denote the level of node  $n$  in schema tree  $T$ . The weight of node  $n$  is defined by a weight function:

$$weight(n) = 2^{depth(T)-level(n)} (\forall n \in T) \tag{5}$$

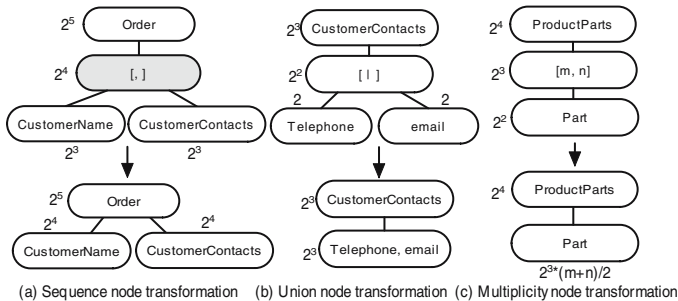
The weights of all nodes fall in the range of  $[2, 2^{depth(T)}]$ . Each weight reflects the importance of a node in schema tree  $T$ .

Having seen that traditional tree edit distance algorithms are not suitable for XML schema trees, three transformation rules: *split*, *merge* and *delete* are proposed to solve this problem. These rules are used to transform constraint nodes, specifically, sequence nodes, union nodes and multiplicity nodes to tag nodes. At the same time, the weights of nodes are reassigned. Fig. 3 gives an example of the three rules.

Note that the definition of complex types can be nested according to XML schema specification. Thus, given a schema tree, we apply the three transformation rules to its nodes level by level, from bottom to top. After the bottom-up transformation, schema tree  $T$  is converted into a new schema tree  $T^*$ . Each node  $n$  of  $T^*$  is a tag node, which contains a few words.

Our idea relies on a hypothesis that two co-occurrence words in a WSDL description tend to have same semantics. We exploit the co-occurrence of words in word bags to cluster them into meaningful concepts. To improve accuracy of semantic measurement, we first carry out the pre-processing step before words clustering, which has been done in the service relevance computation.

Then we use the agglomeration algorithm [21] to cluster words set  $I = \{w_1, w_2, \dots, w_m\}$  into concept set  $C = \{C_1, C_2, \dots\}$ . There are three steps in the clustering process. It begins with each word forming its own cluster and gradually merges similar clusters.



**Fig. 3.** XML Schema tree transformation

Finally, we get a set of concepts  $C$ . Each concept  $C_i$  consists a set of words  $\{w_1, w_2, \dots\}$ . To compute semantic similarity between schema-tree nodes, we replace each word in tag nodes with its corresponding concept, and then use the TF/IDF measure. After schema-tree transformation and semantic similarity measure, the tree edit distance can be applied to match two XML schema trees by the new cost model.

**Obtaining Connectivity.** As it has been mentioned before, we use tree edit distance to match two schema trees. It is equivalent to finding the minimum cost mapping. Let  $M$  be a mapping between schema tree  $T_1$  and  $T_2$ , let  $S$  be a subset of pairs  $(i, j) \in M$  with distinct word bags. Let  $D$  be the set of nodes in  $T_1$  that are not mapped by  $M$ , and  $I$  be the set of nodes in  $T_2$  that are not mapped by  $M$ . The mapping cost is given by  $C = Sp + Iq + Dr$ , where  $p$ ,  $q$  and  $r$  are the costs assigned to the relabel, insertion, and removal operations according to the cost model proposed in section 6.1.2. We call  $C$  the *match distance* between  $T_1$  and  $T_2$ , denoted as  $C = ED(T_1, T_2)$ . Match distance reflects semantic similarity of two schema trees.

Now let us see how to compute the connectivity of a web-service operation.

Given two web-service operations  $op_1 : s_1, s_2, \dots, s_n \rightarrow t_1, t_2, \dots, t_m$  and  $op_2 : u_1, u_2, \dots, u_l \rightarrow v_1, v_2, \dots, v_k$ . Let  $X = \{t_1, t_2, \dots, t_m\}$  and  $Y = \{u_1, u_2, \dots, u_l\}$ . The connectivity of  $op_1$  with respect to  $op_2$  is  $Con_{op_2 \rightarrow op_1} = sim(X, Y)$ . To achieve  $sim(X, Y)$ , for each schema tree  $\in X$ , we find its corresponding schema tree  $\in Y$  with the minimum match distance. We simply identify all possible matches between two lists of schema trees  $X$  and  $Y$ , and return the source-target correspondence that minimizes the overall match distance between the two lists. It does not depends on whether the number of schema trees is the same or not between  $X$  and  $Y$ . This process is illustrated by algorithm 1.

```

input :  $op_1 : s_1, s_2, \dots, s_n \rightarrow t_1, t_2, \dots, t_m$ 
          $op_2 : u_1, u_2, \dots, u_l \rightarrow v_2, \dots, v_k$ 
output: The connectivity of  $op_1$  with respect to  $op_2$ 
1 for  $i \leftarrow 1$  to  $m$  do
2   |  $S_i = \min\{ED(t_i, u_j) | j = 1, 2, \dots, l\};$ 
3 end
4  $Con_{op_2 \rightarrow op_1} = \sum_{i=1}^m S_i$ 

```

**Algorithm 1.** Algorithm for computing web-service operation connectivity

## 6.2 The *CompImp* Algorithm

Based on the strategies proposed in section 5.3, we design the *CompImp* algorithm for automatically computing the importance of a set of given web-service operations  $OP = \{op_1, op_2, \dots, op_N\}$ . The algorithm iteratively computes the importance values for all operations until convergence. It initializes the importance

```

input : A set of web-service operations  $OP = \{op_1, op_2, \dots, op_N\}$ 
output: An array  $I[1 : N]$  to store the importance values of  $OP$ 
1 foreach service operation  $op_i \in OP$  do
2    $I_i^{cur} = 1/N$ ;
3    $convergence[1 : N]=false$ ;
4 end
5 repeat
6   foreach service operation  $op_i \in OP$  do
7     calculate  $I_i^{cur}$  using equation 2;
8     if  $|I_i^{new} - I_i^{cur}|/I_i^{cur} \leq c$  then
9        $convergence[i] = true$ ;
10    else
11       $convergence[i] = false$ ;
12       $I_i^{cur} = I_i^{new}$ ;
13    end
14  end
15 until  $convergence[1:N]=true$  ;
16 Return  $I$ ;

```

**Algorithm 2.** The *CompImp* Algorithm

of each service operation to  $1/N$  and then iteratively applying equation 2 until the importance values converge, i.e., for each operation, the difference between the old and the new importance value is less than some threshold  $c$  (typically, we can choose  $c = 0.1\%$ ). The details of *CompImp* are shown in Algorithm 2.

Once the importance of all available web-service operations has been obtained, we simply define the importance of a web service as the average of the importance values of all operations in the service. The process is straightforward and not presented due to space limitations.

### 6.3 Combining Service Relevance with Importance

Recall that in section 4, each web service  $s$  is assigned a relevance score by similarity measure. In order to reflect the two factors we proposed for characterizing the user's preference degree for  $s$ , we need to incorporate a service importance score into the relevance score of  $s$ . Then, we can rank  $s$  according to its combination score, which is a weighted sum of its relevance score with a query  $q$  and its importance score. Formally, we have

$$Ranking\_Score(q, s) = \begin{cases} w \times R_s + (1 - w) \times I_s & \text{if } R_s > 0 \\ 0 & \text{otherwise} \end{cases} \quad (6)$$

where  $0 \leq w \leq 1$ . Both  $R_s$  and  $I_s$  need to be normalized to between  $[0, 1]$ . A higher rank score indicates a more desirable web service, so the user's top-k search requirement can be satisfied.



## 7 Experiments and Evaluations

We have implemented a prototype system to evaluate the techniques presented in this paper. First, we investigate the time saving issue and present a service index structure that is used in our experiments; second, we evaluate the performance of building the service index; finally, we evaluate the effectiveness and efficiency of our IR-style search strategy.

The experiments were conducted on a P4 Windows machine with a 2GHz Pentium IV and 512M main memory. The data set used in our tests is a web service repository collected from [22,23,24]. Their WSDL specifications are available so we can obtain the textual descriptions and XML schemas of input/output data types. The data contains 223 web services including 930 web-service operations.

In order to improve the performance of searching for desired web services, we design a service index for the service repository. The service index keeps TF/IDF information about each WSDL document. Considering schema tree matching is time consuming, we also included an importance entry in the service index.

We first evaluated the efficiency of building the service index. The time performance of our algorithm is tested with the increase of the number of web-service operations. Taking CPU time as the standard measure, we get time costs of building the service index in Fig. 4(a), in which the time includes the costs of constructing relevance entry for all services, and the computation of the importance entry as well. Fig. 4(a) shows that, as the number of operations increases, the time cost of building the service index increases rapidly. This indicates that, by adding service operations, the number of schema trees increases significantly, leading to more cost of schema tree matching. However, the efficiency is still good since we take a fast tree edit distance method from [25].

We then evaluated the efficiency of searching for desired web services. The time cost is given in Fig. 4(b). It is can be seen that the time increases almost in a linear way with respect to the number of web service operations. This demonstrates that by building the service index, our searching performance is rather high, although building index is a bit time costing. But considering the fact that the web service repository does not change frequently comparing with a user's query request, the service index is effective and practical.

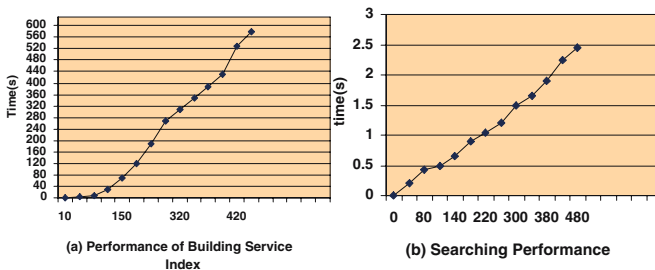


Fig. 4. Performance

## 8 Conclusions

In this paper, we have presented a novel IR-Style mechanism for discovering and ranking web services automatically, given a textual description of desired services. We have introduced the notion of preference degree for a web service, and suggested *relevance* and *importance* as two desired properties for measuring its preference degree. Also, various algorithms are given to obtain service relevance and importance. The key part for computing service importance is our schema tree matching algorithm, which catches not only structures, but even better semantic information of schemas defined in web services. Experimental results show the proposed IR-style search strategy is efficient and practical.

As part of on-going work, we are interested in improving efficiency of the connectivity computation algorithm in terms of running time, since the computation of extended tree edit distance is costly.

## References

1. Booth, D., Haas, H., McCab, F., Newcomer, E., Champion, M., Ferris, C., Orchard, D.: Web Services Architecture (2004), <http://www.w3.org/TR/ws-arch/>
2. Sajjanhar, A., Hou, J., Zhang, Y.: Algorithm for Web Services Matching. In: Yu, J.X., Lin, X., Lu, H., Zhang, Y. (eds.) APWeb 2004. LNCS, vol. 3007, pp. 665–670. Springer, Heidelberg (2004)
3. Wang, Y., Stroulia, E.: Flexible Interface Matching for Web-Service Discovery. In: Proceedings of International Conference on Web Information Systems Engineering (WISE) (2003)
4. Dong, X., Halevy, A.Y., Madhavan, J., Nemes, E., Zhang, J.: Similarity Search for Web Services. In: Proceedings of International Conference on Very Large Data Bases (VLDB), pp. 372–383 (2004)
5. Sycara, K.P., Widoff, S., Klusch, M., Lu, J.: Larks: Dynamic Matchmaking Among Heterogeneous Software Agents in Cyberspace. *Autonomous Agents and Multi-Agent Systems* 5(2), 173–203 (2002)
6. Roman, D., Lausen, H., Keller, U.: Web Service Modeling Ontology (WSMO). WSMO Final Draft 10 (2005)
7. Shen, Z., Su, J.: Web service discovery based on behavior signatures. In: Proceedings of International Conference on Services Computing (SCC), vol. 1, pp. 279–286 (2005)
8. Hristidis, V., Gravano, L., Papakonstantinou, Y.: Efficient IR-Style Keyword Search over Relational Databases. In: Proceedings of International Conference on Very Large Data Bases (VLDB), pp. 850–861 (2003)
9. Hristidis, V., Papakonstantinou, Y.: DISCOVER: Keyword Search in Relational Databases. In: Proceedings of International Conference on Very Large Data Bases (VLDB), pp. 670–681 (2002)
10. Agrawal, S., Chaudhuri, S., Das, G.: DBXplorer: A System for Keyword-Based Search over Relational Databases. In: Proceedings of International Conference on Data Engineering (ICDE) (2002)
11. Brin, S., Page, L.: The anatomy of a large-scale hypertextual web search engine. *Computer Networks* 30(1-7), 107–117 (1998)

12. Pu, K., Hristidis, V., Koudas, N.: Syntactic Rule Based Approach to Web Service Composition. In: Proceedings of International Conference on Data Engineering (ICDE), p. 31 (2006)
13. Yu, C., Jagadish, H.V.: Schema summarization. In: VLDB, pp. 319–330 (2006)
14. Hao, Y., Zhang, Y., Cao, J.: WSXplorer: Searching for desired web services. In: Krogstie, J., Opdahl, A.L., Sindre, G. (eds.) CAiSE 2007 and WES 2007. LNCS, vol. 4495, pp. 173–187. Springer, Heidelberg (2007)
15. He, Q., Yan, J., Yang, Y., Kowalczyk, R., Jin, H.: Chord4s: A p2p-based decentralised service discovery approach. In: IEEE SCC (1), pp. 221–228 (2008)
16. Al-Masri, E., Mahmoud, Q.H.: Qos-based discovery and ranking of web services. In: ICCCN, pp. 529–534 (2007)
17. Al-Masri, E., Mahmoud, Q.H.: Investigating web services on the world wide web. In: WWW, pp. 795–804 (2008)
18. Salton, G., Wong, A., Yang, C.S.: A Vector Space Model for Automatic Indexing. Communications of the ACM (CACM) 18(11), 613–620 (1975)
19. Reis, D.D.C., Golgher, P.B., Silva, A.S.d., Laender, A.H.F.: Automatic web news extraction using tree edit distance. In: Proceedings of WWW Conference, pp. 502–511 (2004)
20. Zhang, K., Shasha, D.: Simple Fast Algorithms for the Editing Distance Between Trees and Related Problems. SIAM Journal on Computing 18(6), 1245–1262 (1989)
21. Kaufman, L., Rousseeuw, P.J.: Finding Groups in Data: An Introduction to Cluster Analysis. John Wiley, New York (1990)
22. <http://www.xmethods.org> (XMethod)
23. <http://www.bindingpoint.com> (BindingPoint)
24. <http://www.webservicelist.com> (WebServiceList)
25. Nierman, A., Jagadish, H.V.: Evaluating structural similarity in xml documents. In: WebDB, 61–66 (2002)

# Towards a Sustainable Services Innovation in the Construction Sector

Sylvain Kubicki<sup>1</sup>, Eric Dubois<sup>1</sup>, Gilles Halin<sup>2</sup>, and Annie Guerriero<sup>1,2</sup>

<sup>1</sup> Public Research Centre Henri Tudor  
29, av. J.F. Kennedy, L-1855 Luxembourg-Kirchberg, Luxembourg  
{sylvain.kubicki, eric.dubois, annie.guerriero}@tudor.lu

<sup>2</sup> Research Centre in Architecture and Engineering  
2, rue Bastien Lepage, 54001 Nancy, France  
gilles.halin@crai.archi.fr

**Abstract.** In this paper, we report on a business case in the construction sector where we have designed and prototyped an innovative Web-based distributed document management application. It supports various exchange and sharing of information services between the different stakeholders involved in a construction project. The development of the application is based on a service-oriented architecture and follows a systematic model-driven engineering approach. Besides the application itself, the paper also reports on a Sustainable Services Innovation Process (S2IP) guiding our activities related to the valorization and the successful technology transfer of a demonstrator into an innovative product. We illustrate how this innovation process has been applied to this business case in the construction sector where a networked value constellation has been identified and realized with professionals of the construction sector (including a standardization body), software houses and our technology transfer centre.

**Keywords:** service oriented architecture, model-driven engineering, science for service systems, open and networked innovation process, networked value constellation, construction domain.

## 1 Introduction

Cooperation between actors is essential for the success of a construction project. The short-lived groups of actors, the heterogeneity of stakeholders and of the local strategies of their firms are the main specificities of AEC (Architecture, Engineering and Construction) sector activities. Indeed, the diversity of projects and architectural realizations is added to the complexity of groups of stakeholders and relations among them. In this context, the improvement and the change of work methods takes time, and there are clear opportunities for innovation through IT services.

In Luxembourg, to answer to this need, the Public Research Centre Henri Tudor (CRPHT) has been engaged in several R&D projects, most of them in a PPP (Public/Private Partnership) approach with different stakeholders active in the sector

as well as with the CRTI-B<sup>1</sup>, the national professional association promoting new usages of ICT in the construction sector and its associated standards. These projects have resulted in several demonstrators and prototypes [1, 2], the latest one, dealing with a document management services system supporting construction projects, is the focus of this paper. Developing demonstrators is clearly an important activity in a global innovation process in order to get the support of early adopters through experiments and validation. However such demonstrators are only one of the elements that is part of a complete innovation chain resulting in a successful technology transfer. From its past experiences, CRPHT has built and continuously improved the elements of this chain, resulting in a so-called S2IP (Sustainable Service Innovation Process). S2IP supports a networked and open innovation approach [3] based on the identification of a networked value constellation [4] making sustainable this innovation. In this paper we illustrate how the S2IP has been applied to the demonstrator developed for the innovative document management services system.

The structure of the paper reflects the twofold orientation of this work, namely the development of the innovative document management services system demonstrator as well as of the technology transfer innovation process associated with this demonstrator itself. The approach for designing new document management services is presented in Section 4 where is detailed the followed model-driven service design and architectural approach used for building a demonstrator (prototype). The work is mainly based on the instantiation of a meta-model that has been built for understanding the nature of cooperation and collaboration activities taking place in the construction sector. The metamodel is presented in Section 2. Transforming an innovative demonstrator into a sustainable innovation is the target of the S2IP whose generic associated activities are introduced in Section 3. Its application based on the identification of a networked value constellation making sustainable the original demonstrator is detailed in Section 5. Section 6 concludes with a summary of the paper and with an overview of our future plans.

## 2 Electronic Cooperation in the Construction Sector

In this section, we first start by motivating the need for research in new electronic platform for supporting cooperation and collaboration dedicated to the specific services and associated constraints of the AEC sector. Then we report on actual research results regarding the definition of a meta-model associated with the nature of the cooperation in the sector and explain the methodological approach followed in the development of the demonstrator.

### 2.1 Cooperative Practices in AEC

The AEC business market is largely driven by building & infrastructure projects demand. Such projects involve temporarily teams of heterogeneous actors (architects,

---

<sup>1</sup> The Resource Centre for Technologies and Innovation in Construction (CRTI-B) is a standardization body involving all the representative building trades in Luxembourg (owners, architects/engineers and contractors) in more than 40 thematic working groups. It was created in 1990. <http://www.crtib.lu>

engineers, contractors, material providers, etc.) able to respond to the customer's requirements (namely its architectural program). Each of these heterogeneous firms has its own internal processes, methods and IT infrastructures. Then, the project's activity today is characterized by a *low-level of integrated design & construction collective processes*.

In this particular context, the "concurrent engineering methods and tools" applied for many years in several other industries (e.g. automotive, aerospace) are not well adapted to the specificities of cooperation in AEC projects. Thus, our research is considering an alternative concept of "cooperative engineering" [5] favoring mutual adjustment, trust between practitioners and enabling the necessary *flexibility of processes realization* and the need for *an adapted IT environment*.

We underline that collective processes have not to be described in the details in order to be flexible. Then we think that numerous process modeling approaches (both methods and IT support) are not well adapted to our case. But "working processes" could be described and agreed at a "high level", on the basis of a common and shared vocabulary (more details in Section 2.2) between actors of the domain. That is what the CRTI-B working groups did (more details in Section 3.2), with the consensus of all the involved representative partners. Then, IT support can be considered from the perspective of these cooperative (best) practices.

## 2.2 A Methodological Approach of Cooperation Support

In the role of technology transfer centre, CRPHT is not developing real software products but more demonstrators (or prototypes) that can be used for the purpose of experiments. The goal is to demonstrate the benefits of the new features to the different stakeholders of the sector. In the targeted AEC sector, for developing rapidly and in a flexible way these demonstrators, we are following a Model Driven Engineering (MDE) approach.

This approach is based on model development, steering both domain analysis and tool engineering. It is largely based on Model Driven Architecture (MDA) for software systems development [6] where the objective is to define a framework of certified industrial standards (MOF, UML). In parallel, the Model Driven Engineering (MDE) research area is an evolution aiming to unify different technical spaces (XML, ontology etc.). It does not focus on a unique technology: it is an integrative approach [7]. MDE recommends the use of meta-models to define domain languages. Models represent real systems. Each model has to be conformed to its meta-model [8, 9]. Finally the transformation concept is a central one, a transformation being itself described with a model.

We use this methodological framework and propose two levels of modeling for the cooperative activity in the AEC domain [10, 11]. First, a Cooperation Context Meta-Model (CCMM) allows us to describe the cooperative activity at a high level of abstraction. This meta-model is used to construct a specific model representing the particular context of a real construction project. The instantiation MOF architecture ( $M2 > M1 > M0$ ), which we base this reasoning on, nicely fits with in the approach based on models and meta-models from MDE.

Our CCMM (M2) takes into account the existing relations between the different elements of a project (See Figure 1). We identify three main categories of elements

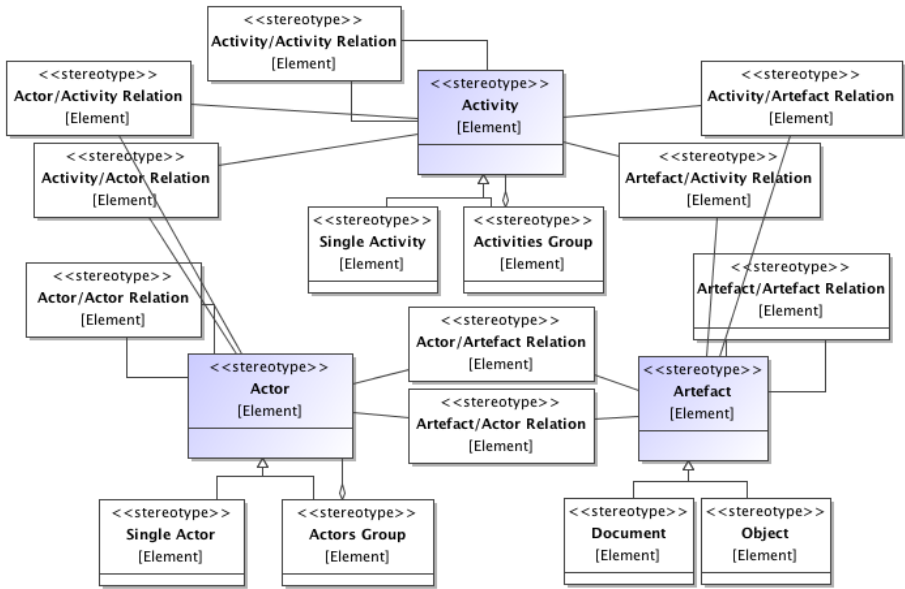


Fig. 1. Cooperation Context MetaModel CCMM – M2 (extract)

existing in every cooperation project: the activity, the actor and the artefact (associated with documents and objects related to an activity). CCMM strengthens the relationships existing between these elements of cooperation.

More details about the CCMM can be found in articles where we have M1 models that have been instantiated to represent specific architectural design context [12] and building construction activity dedicated context [11]. Finally, these Cooperation Context Models (CCM, M1) also enable the description of particular project contexts (M0) representing the business knowledge in which actors cooperate. Note also that an additional benefit of having several M1 models instantiated from the same M2 is to guarantee a better interoperability between different information systems possibly located in different organizations (e.g. engineering offices, architectural agencies...).

In this paper, we will report on a specific new M1 model - focusing on the specific early building design/construction activities and the need for extensively exchanging documents (plans, 3D views, etc.).

### 3 Towards a Sustainable Service Innovation Process (S2IP)

#### 3.1 An Introduction to the S2IP

In the previous section, we have introduced the technological and scientific ingredients that are at the basis of the demonstrator offering new services related to AEC document management. Those services and their underlying architecture will be further detailed in Section 4. At this point, we would like to present how the design and the validation of these services are part of a general process applied by CRPHT in a rigorous management of its innovation activities.

CRPHT is the Luxembourg R&D centre dedicated to the support of technology transfer and innovation in different technological domains including ICT, health, environment and materials. In the ICT domain, most of these applications are services oriented. This is in line with the nature of the national economy where the service sector accounts for above 85% percent of total value added in 2006 granting Luxembourg with the first place in the European landscape. According to the EARTO terminology ([www.earto.org](http://www.earto.org)), CRPHT is a RTO, a public Research and Technology Organization whose is a “specialized knowledge organization dedicated to the development and transfer of science and technology to the benefit of the economy and society”. The main mission of a RTO is therefore to provide research, development and innovation services both to private and public beneficiaries according to an open approach (or Public-Private-Partnership) where it acts as interface between universities and firms [13].

In order to perform its public mission, transfer of technologies but also of knowledge, ideas and concepts, CRPHT has defined and applied an innovation management process targeting the support to innovation in services within open partnerships with the targeted beneficiaries [3]. This process is called “Sustainable Service Innovation Process” (S2IP) [14]. It is based on a participatory and collaborative innovation approach in order to sustain deep involvement of the network’s actors in the development of innovation services. Targeted services are mostly based on ICT services but packaged into business services, i.e. also including the organizational (processes) and the human (skills and competencies) perspectives. This view is in line with the new research domain of Service Science [15]. The overall structure of S2IP is depicted in Figure 2.

Although the figure may suggest that the S2IP is lifecycle oriented, the reality is that each box corresponds to a process that has to be performed and may be pursued in parallel with other processes in a non strict sequence.

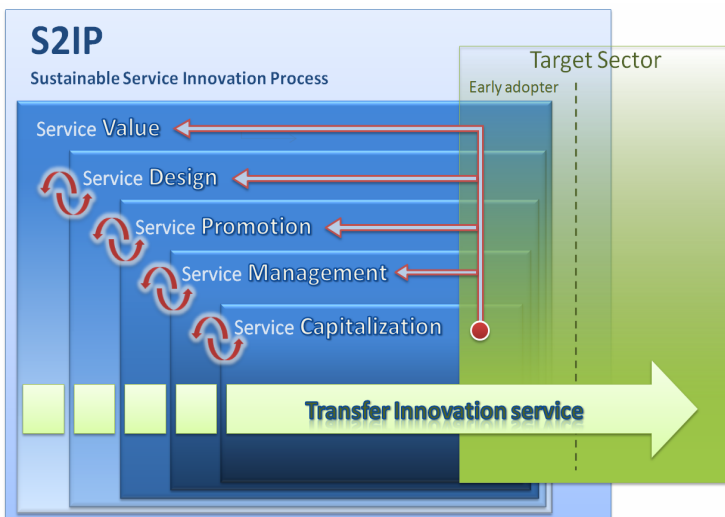


Fig. 2. The S2IP Innovation Process



The main processes are:

1. *Service value and business strategy*: This process covers the activities associated with the identification of an opportunity for a new service innovation. They cover a study of the technological feasibility of the service (which, in most cases, requires the building of a demonstrator for the purpose of experiments with early-adopters) as well as a preliminary identification of the business model associated with the value proposition (both expressed in terms of tangible financial elements and of intangible assets).
2. *Service design and engineering*: This process is associated with the definition of the service not only in terms of its business functional objectives but also in terms of all its required (non-functional) qualities. Thus requires to elicit the strategic goals of the different early-adopters stakeholders involved in the final acceptance of the service as well as to understand the constraints associated with the environment (like specific regulations associated with a sector). From this initial elicitation, requirements have to be formally expressed in terms of properties of the services that can be organized in terms of a service contract (or a service level agreement).
3. *Service promotion*: Once the service contract has been validated by early adopters, it is important to promote the service to other potentially interested parties. This can be done within an organization through some marketing regarding the socio-economical sustainability of the service. In a network of organizations or for a sector, this promotion can also include initiatives regarding the branding of the new service through some label definition and associated certification scheme. Ultimately standardization activities run for example at the national or international levels (like e.g. ISO) definitively help in a successful promotion of the service.
4. *Service management*: This is out of the scope of CRPHT's mission to deploy by itself the service with an organization or within a sector. This is where the market should play its role. However we define and provide tools that can be used by those that will deploy the service for checking and measuring the correctness of its implementation. In particular for each new service we propose metrics associated with the measurement of the quality of the services implementation with respect to the services contract.
5. *Service capitalization*: Once a services system is deployed within organizations, we can start to collect the feedbacks associated with the measures as well as from assessment performed with the end-users. The analysis of this feedback indicates the possible evolution of the service in terms of new requirements, new business model, etc. Thus this is where new iterations associated with the different processes described above are starting.

### **3.2 Applying the S2IP Value Proposition Process to the Building Construction Sector**

Starting from the generic presentation of the S2IP introduced above, we now illustrate its application to our business case in the construction sector by first considering the initial *Service Value* process and then detailing the four other processes in Section 5.

The *Service Value* process has the twofold objective of “*inventing*” new services together the *strategic business model* underlying their acceptance and sustainability. The invention itself is difficult to formally describe but always results from the matching between the knowledge about an innovation opportunity in a sector or a firm, and the knowledge acquired about the potentialities of new technologies, new processes, new methods, etc. In our case, knowledge about the sector was gained through our long-term relationship with the CRTI-B (the national professional association promoting new usages of ICT in the construction sector) in Luxembourg, while knowledge about technology and scientific advances was acquired through our cooperation with the co-author of this paper (namely the MAP-CRAI laboratory in Nancy).

At the centre of our innovation idea was the decision to address the issue of document management through two viewpoints: the human one (i.e. human practices related to document management) and the technological one (i.e. existing software solutions).

The issue of document management in weakly integrated activities needs to focus on the structuring of metadata related to these documents [16, 17] rather than on cross-organizational workflows approaches. Metadata management should be considered both the human viewpoint (Why do I share a document? How to document the flow of documents?) and the technological one (How to represent metadata? How should the users fill in metadata?). Regarding such approach, it should be noted that Turk & Björk suggested one of the first model describing document-related concepts in AEC: presentation, document lifecycle, organization and especially the link with building product models [18].

In this context, designing new document management services for the AEC sector is not really challenging but designing those which really answer to the demand of the CRTI-B is really the topic of the innovation. To do so the following activities were performed with the overall objective of demonstrating the existence of a business model (further detailed in Section 5) for new software services fitting a clearly identified market.

1. During the first activity, enquiries were performed and showed that most of the users were not really satisfied with the existing solutions. Some reasons were collected through brainstorming with practitioners. Interesting synthesis papers also introduced some metrics and indicators to understand the factors of success or failure of AEC groupware solutions [19, 20]. The complexity of their common functionalities and associated services is one important reason [21]. Their low adequacy to the AEC projects specificities (in particular organizational, processes and actors' skills ones) is another reason of the failures in introducing such new IT information systems. Finally technical reasons also have been underlined. Actors have to use numerous IT solutions (one project – one tool) because it is often that the owner or main contractor decides and forces to use such a system in its project.
2. Then, as the decision was taken regarding the development of a new solution, initial needs have been formulated by the end-users themselves through a second enquiry/interview stage. Working practices have been collected by CRPHT who transformed them into a comprehensible set of best working practices. A dedicated working group allowed the practitioners to agree on it in a consensus way.

3. During the third activity, six releases of the demonstrator have been incrementally developed and regularly validated with 6 working groups (more details in the next Section). These working groups were constituted of 15 AEC practitioners representing several fields (i.e. architects/engineers, owners). The CRPHT team frequently presented the developments' progress. This enabled a validation of the progress but also an early appropriation of the application by its future users.
4. Experiments begun early with only some basic services in order to rapidly debug the system and to let the users better formulate their needs.

## 4 A Model-Driven Service Design Approach to a Document Management IT Service in AEC

The incremental development of the successive releases of the demonstrator took direct benefits of the MDE approach introduced in Section 2 and of its tailoring to the cooperation issues in AEC. The flexibility and the efficiency of this approach for a systematic derivation and evolution of new specific cooperation platforms was already proved by our previous experiences regarding the development of open-source applications, and of their related business services [2].

### 4.1 Document Management Business Services

In our new application, as explained in the previous section, the objective is to support cooperative practices related to the exchange of documents which fits the specificities of the building construction working practices. From interviews and working group meetings with practitioners involved in construction projects (architects, engineers, owners, contractors), we collected information about their practices. After structuring it, and agreeing on it with the involved practitioners, we organized it around seven high level practices. Two examples are:

- Practice 2: A standard naming of project documents is setup and agreed by all the actors involved. It is developed on the basis of the actual norm used in public buildings projects.
- Practice 4: When a document is shared, it is necessary to inform the interested participants of its availability and of all its related modifications.

From the identified cooperation practices (Px) related to document exchange between practitioners, the following services have been incrementally developed:

- A *file name management service* enables the use of a standard for naming the plans, contracts or meeting reports. It is defined for each construction project and enables to obtain the metadata of a document when it is uploaded. It also ensures the classification of documents in the standard structuring of the project. Standard name is composed of fields and separators. A web service parses the name of the document submitted by the user and interprets its content. If the user uses standard names for his documents, he has just to check if the name is correct. If he does not, he can rename his file using the field contents that the tool suggests him.
- A *notification service* allows the users to receive notification when a document is uploaded, updated or when various actions are performed (such as leaving a

reaction, assigning a request and so on). For example an engineer designing its plan on the basis of the architect's one is automatically informed of the upload of a new architect's index, which could be of interest for him.

- A *request management service* has been developed to manage and keep trace of the interactions between the users. The author of a document can inform someone else that a document has been uploaded. He can also make a request for validation or ask for a reaction. Requests, "due date" and "accomplished date" are then stored and enable to trace the state of a document (validated, waiting for a validation, rejected).
- The *reaction service* traces the discussions between users about a document and allows them for example to directly inform the owner of a change.
- Finally, a *privacy management service* enables to manage privacy areas in which the users are authorized to access or not: a first area is restricted to the designers (architects, engineers), a second one allows the owner to see the plans he has to validate, and a third one is completely public and accessible for all the contractors.

## 4.2 Development of the Demonstrator

Today most of the solutions are based on a classical client/server architecture enabling the description, storage and retrieval of cooperation context data in a database. In our case, for the purpose of openness and ease of integration with existing tools, we choose to orient our developments towards a service-based architecture. Examples of such architectures in AEC include the solution developed in the ISTforCE EU project [22] and the one implemented in the e-Nvision project [23].

Our services' platform manages the cooperation context, based on a specific construction business-domain model instantiated from the cooperation context meta-model CCMM) presented in section 2.2. Following the Model-Driven Engineering approach summarized in the right part of Figure 3, we have built a specific cooperation context (M1) associated with the work practices described in the previous sub-section and derived from CCMM. This modeling framework enables to define an adaptable and flexible business context which was useful in the incremental development of the different demonstrator's releases. On our case, M1 represents the specific context of specific early building/construction activities characterized by many exchanges of documents (plans, 3D views, texts) between different actors (firms, agencies, control bodies, etc.). For example, in the M1, we can manage explicitly the relationships existing between the author of a document (e.g. a plan) and the other actors who have to validate it.

The implementation of the business domain context through the use of IT services requires to use the same modeling framework to guarantee the alignment between the concepts introduced at each modeling level. The IT Service metamodel (left part of Figure 3) describes existing visualization modes, designed business processes, usable functional resources and the configuration elements. The metaconcept matching at M2 level enables to define IT services adapted to specific business cooperation contexts by using business concepts when configuring the IT service (M1 level).

The business-oriented services described above have been developed through the use of Web services. The aim is to facilitate their integration in existing software infrastructures used by some of our partners. These Web services are described in the

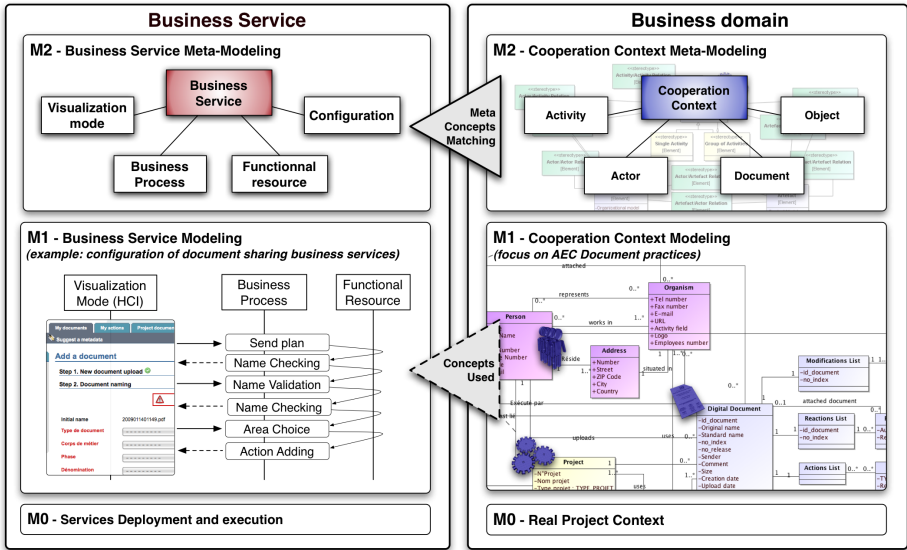


Fig. 3. A MDE approach for the design of IT services fitting a Business Domain

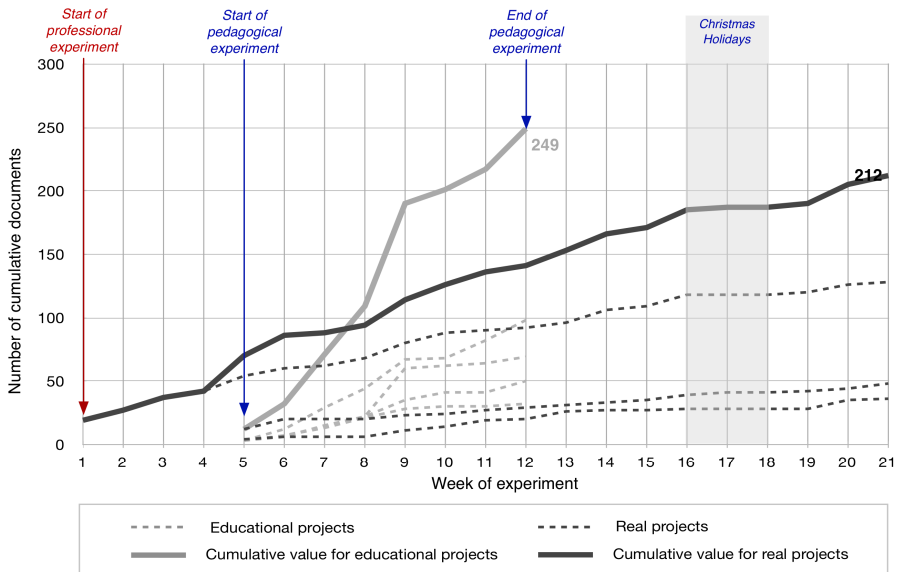
REST protocol [24] and are also available in SOAP. REST is a Web services technology based on the Web architecture and its basic technologies: HTTP, URI and XML. We structured these Web services using the ROA approach (Resource Oriented Architecture) [25]. It describes a set of good practices for REST Web service design and is very adapted to our Agile development process, involving business experts, technical experts and final users.

### 4.3 Validation of the Demonstrator

The prototype implementing these document management business services is under validation in two different experimental contexts. The first one consists of real projects' experiments (3 running at the time of writing this article), where early-adopters use the tool to support the exchange of electronic documents instead of classical ones (such as fax and email). The second one is an education context (an Architecture Master curriculum [26]), where distant architecture students working on architectural design projects (4 projects performed in fall semester 2008) use the tool to share their design documentation.

The figure 4 shows the evolution of the number of e-documents shared each week via the demonstrator. It distinguishes between real construction projects and education projects.

Figure 4 shows that the usage of such an innovative tool is increasing during the duration of each experiment. This reflects the necessary appropriation time, during which users become aware of the possibilities of the tool and progressively trust it. It corresponds also to the increasing amount of documents produced and exchanged in a construction project.



**Fig. 4.** Statistics of usage of the document management tool by week of experiment

Functional feedback is also targeted by the experiment stage. The cooperative practices supported by the IT services implemented seem to fit well the needs of all the practitioners of construction projects. Some particular demands came from architects and engineers (daily users of the tool), especially in order to gain time in uploading documents (addition of metadata) and to improve visualization of the documents (consultation and validation flows). Minor bugs have also been fixed.

## 5 Applying a Sustainable Service Innovation in AEC

As the result of the incremental evolutionary development and experiments of the demonstrator presented in the previous section, there has been an agreement of professional early adopters from the construction sector on the usefulness of the tool and more importantly on the added *value* of its offered services. Thus, the first process of the S2IP (see section 3.1) being accomplished, our role was to make sustainable this innovation by applying the next processes of the S2IP.

The challenge was thus to build a networked value constellations of actors that *jointly satisfy a consumer need, where each actor contributes its own expertise and services* [4]. The consumer's needs and requirements are here represented by professionals (both individuals and companies) from the AEC sector. The actors are the CRTI-B, software houses interested in selling and supporting the software services and the CRPHT. Regarding the latter, in its role of RTO, its objective is not to commercialize the developed software product derived from the demonstrator by itself but more to find partners for doing this and motivate them through the identified value proposition.

Hereafter we describe how the networked value constellation of actors has been put in place through the realization of the remaining four processes (numbered 2-5 in Section 3.1) of the S2IP.

2. *Service design and engineering process:* The first demonstrator was not robust, neither reliable enough to be transferred to a commercial software house. In our case, the benefit of the demonstrator was to help in the elicitation of the features required from the end-users and also to convince them from the usefulness of the application in support to their practices. The transfer (or selling) of a new release more stable to commercial entities is not sufficient for them. What is required is also to precisely define the set of requirements (both functional and non-functional) associated with the future commercial application. This precise set of requirements is required from the software house to help it in the engineering of the final product, but also from the end-users who want to check that the nice features shown through the demonstrator are also part of the final product.
3. *Service promotion process:* One of the mechanisms that can be used for the promotion of new services within a sector is the standardization approach. In other words, if the sector promotes an innovation as being the standard to be followed, it also strongly boosts this innovation. In our case, the CRTI-B has played this role of standardization body at national level. This was first through the endorsement of the proposed meta-data information structure for the documents. Then, from the experiments we conducted with the demonstrator, the CRTIB recognizes the importance to promote its usage in future construction projects. More importantly, the Ministry of Public Construction (one of the partners of the CRTI-B) decided to foster the use of such software services in its future public building construction projects. The CRTI-B also imposes that only a services system respecting the set of requirements formally expressed in the requirements document produced according to the process described above was eligible under its trademark "CRTI-weB".
4. *Service management process:* As indicated above, the role of CRPHT is not to deploy and manage the new designed services by itself. However its role is to support partners in a successful commercialization of the innovation. In our case there is today 5 consultancy software houses which are interested to develop a software system offering the services defined according to the requirements document produced by CRPHT. What they are also interested in is to demonstrate that their developed software is compliant with requirements endorsed by CRTI-B (see above). This is also what the CRTI-B expects from the proposed software. To this end it has mandated the CRPHT in its quality of neutral actor, to check for the compliance of the different proposed solutions with the requirements document. So only the companies, which will have produced a 'certified' product, will get the CRTI-B label. To do this certification approach, the CRPHT is applying a service/software procurement approach based a systematic test of both functional and non functional requirements in order to detect and track inconsistencies.
5. *Service capitalization process:* CRTI-B has also asked to CRPHT for a follow-up of construction projects managed with the new document management services system. To this end a working group will be set up in order to get feedbacks coming both from the professional construction actors as well as from software houses. The objective is here to enter a Plan-Do-Check-Act (PDCA) model aiming

at improving the set of delivered services through an agreement and the publication of new releases of the requirements document.

## 6 Conclusion and Future Work

This article presents the design of IT services for the construction sector in Luxembourg. In particular, it focuses on a specific approach based on a high implication of the AEC sector professionals through the CRTI-B national standardization body. The design of an innovative document management IT services system demonstrator has been realized according to the Model-Driven Engineering, paradigm enabling the modeling of the business domain, and the rapid prototyping of the services. This development is carried out in the framework of a Sustainable Service Innovation Process (S2IP) driving the various R&D projects of CRP Henri Tudor.

Besides a continuous improvement and formalization of the S2IP through the capitalization of other innovations performed in the context of open networks of stakeholders, more technical work is pursued in two directions.

- First, we are opening the document management services to other information systems, especially the ones used internally by some of our partners (e.g. engineering firms). The required interoperability between the global document management solution and these particular services is an essential key to guarantee a large usage of the application in Luxembourgish construction projects.
- Second, we are also addressing several research issues related to the development of these services. In particular we plan to improve the description of IT services (modeling) and of the business domain use cases (cooperation context models, M1). Our aim is to setup a repository of M1 business models, enabling the selection/discovery of IT services closely related to the specificities of these business contexts. In the construction domain it will allow us to provide IT services' offers fitting the particular context of each architectural project.

## Acknowledgements

The authors would like to thank the CRTI-B network and more especially the practitioners who participated in working groups and took part in real-world experiments.

## References

1. Kubicki, S., Guerriero, A., Halin, G.: Model-based eServices for supporting Cooperative Practices in AEC. In: ECPPM 2008 Conference. E-Business and e-work in Architecture, Engineering and Construction. Sophia-Antipolis, France (2008)
2. Kubicki, S., Guerriero, A., Halin, G., Hanser, D.: IT services design to support coordination practices in the Luxembourgish AEC sector. In: Luo, Y. (ed.) CDVE 2007. LNCS, vol. 4674, pp. 396–403. Springer, Heidelberg (2007)



3. Chesbrough, H.: The era of open innovation. *MIT Sloan Management Review* 44(3), 35–41 (2003)
4. Tapscott, D., Ticoll, D., Lowy, A.: *Digital Capital: Harnessing the Power of Business Webs*. Nicholas Brealy Publishing, London (2000)
5. Kubicki, S.: *Assister la coordination flexible de l'activité de construction de bâtiment. Une approche par les modèles pour la proposition d'outils de visualisation du contexte de coopération*, PhD Thesis, Université Henri Poincaré, CRAI - Centre de Recherche en Architecture et Ingénierie, Nancy (2006)
6. Soley, R.: *OMG: Model Driven Architecture*, Object Management Group (2000)
7. Bézivin, J.: On the Unification Power of Models. *Software and Systems Modelling (SoSym)* 4(2), 171–188 (2005)
8. Favre, J.M.: Towards a Basic Theory to Model Driven Engineering. In: *Workshop on Software Model Engineering, WISME 2004, joint event with UML2004*. Lisboa, Portugal (2004)
9. Frankel, D.: *Model Driven Architecture: Applying MDA to Enterprise Computing*. OMG Press (2003)
10. Halin, G., Kubicki, S.: Une approche par les modèles pour le suivi de l'activité coopérative de construction d'un bâtiment. In: Augeraud, M. (ed.) *Une interface multivue et des services métiers orientés gestion de chantier*, in RSTI série ISI (Ingénierie des Systèmes d'Information). Modèles, formalismes et outils pour les systèmes d'information, Lavoisier, Paris (2008)
11. Kubicki, S., Bignon, J.C., Halin, G., Humbert, P.: Assistance to building construction coordination. Towards a multi-view cooperative platform. *ITcon Electronic Journal of Information Technology in Construction* 11, 565–586 (2006); (Special Issue Process Modelling, Process Management and Collaboration, edited by Katranuschkov, P.)
12. Hanser, D.: *Proposition d'un modèle d'auto coordination en situation de conception, application au domaine du bâtiment*, PhD Thesis, Institut National Polytechnique de Lorraine, CRAI - Centre de Recherche en Architecture et Ingénierie, Nancy (2003)
13. Dodgson, M., Gann, D., Salter, A.: *Think, Play, Do: Technology, Innovation and Organization*. University of Oxford Press, Oxford (2005)
14. Absil, F., Dubois, E., Grein, L., Michel, J.-P., Rousseau, A.: Trust in the Heart of the Open Innovation: Lessons by the Resource Centre for Information Technologies for the Building Industry. In: *International Society for Professional Innovation Management (ISPIM)* (2008)
15. Chesbrough, H., Spohrer, J.: A research manifesto for services science. *Communications of the ACM* 49(7), 35–40 (2006)
16. Caldas, C., Soibelman, L.: Automating hierarchical document classification for construction management information systems. *Automation in Construction* 12, 395–406 (2003)
17. Forcada, N., Casals, M., Roca, X., Gangolells, M.: Adoption of web databases for document management in SMEs of the construction sector in Spain. *Automation in Construction* 16, 411–424 (2007)
18. Turk, Z., Bjork, B.-C.: Document Management Systems as an Essential Step towards CIC. In: *CIB-W78 Conference 1994*, Helsinki, Finland (1994)
19. Nitithamyong, P., Skibniewski, M.J.: Web-based construction project management systems: how to make them successful? *Automation in Construction* 13, 491–506 (2004)
20. Nitithamyong, P., Skibniewski, M.J.: Key Success/Failure Factors and their Impacts on System Performance of Web-Based project Management Systems in Construction. *ITcon Electronic Journal of Information Technology in Construction* 12, 39–59 (2007)

21. Björk, B.-C.: The Impact of Electronic Document Management on Construction Information Management. In: CIB-W78 Conference 2002. Aarhus School of Architecture (2002)
22. Katranuschkov, P., Scherer, R., Turk, Z.: Intelligent services and tools for concurrent engineering? An approach towards the next generation of collaboration platforms. *ITcon Electronic Journal of Information Technology in Construction* 6, 111–128 (2001); (Special Issue Information and Communication Technology Advances in the European Construction Industry)
23. Angulo, I., Garcia, E., Peña, N., Sanchez, V.: E-nvisioning the participation of European construction SMEs in a future e-Business scenario. In: *ECPPM 2006, E-Business and e-work in Architecture, Engineering and Construction*, Valencia, Spain (2006)
24. Fielding, R.T.: *Architectural Styles and the Design of Network-based Software Architectures*, PhD Thesis, University of California, Irvine, USA (2000)
25. Richardson, L., Ruby, S.: *RESTful Web Services. Web Services for the Real World*. O'Reilly, Sebastopol (2007)
26. Kubicki, S., Bignon, J.C., Leclercq, P.: Cooperative Digital Studio: IT-supported Cooperation for AEC students. In: *CIB-W78 2008 Improving the management of construction projects through IT adoption*. Santiago de Chile, Chile (2008)

# P2S: A Methodology to Enable Inter-organizational Process Design through Web Services

Devis Bianchini<sup>1</sup>, Cinzia Cappiello<sup>2</sup>, Valeria De Antonellis<sup>1</sup>,  
and Barbara Pernici<sup>2</sup>

<sup>1</sup> University of Brescia

{bianchin,deantone}@ing.unibs.it

<sup>2</sup> Politecnico of Milan, Milan (Italy)

{cappiell,pernici}@elet.polimi.it

**Abstract.** With the advent of Service Oriented Architecture organizations have experienced services as a platform-independent technology to develop and use simple internal applications or outsource activities by searching for external services, thus enabling inter-organizational interactions. In this scenario, services are units of work provided by service providers and offered to the other organizations involved in a collaborative business process. Collaboration should be facilitated by guaranteeing a homogeneous description of services at the right level of granularity. We propose a methodology to support the designer of a business process in the identification of services that compose the process itself. The methodology should allow collaborative partners to standardize process modelling through component services, enabling effective inter-organizational service discovery. The methodology is presented by means of a running example in a real case scenario.

**Keywords:** service-based process decomposition, service-based collaborative processes.

## 1 Introduction

Internet and related technologies demonstrate that information systems and IT can provide a strategic platform to support the collaboration among Internetworked Enterprises (IE) [1], that is, borderless organizations that share applications, services and knowledge and whose processes are transformed and integrated with the ones of their partners. Heterogeneity of such collaborative environments implies the adoption of standards and infrastructures to communicate. With the advent of Service Oriented Architecture (SOA), organizations have experienced services as a platform-independent technology to develop and use simple internal applications or outsource activities by searching for external services, thus enabling inter-organizational interactions. In particular, IE can share their own applications by using the Software as a Service paradigm

(SaaS). They can place their own implemented functionalities at the other organizations' disposal by creating services and providing them across the Internet, thus designing operating information systems able to connect IEs each other. In this scenario, services are units of work provided by service providers and offered to the other organizations involved in a collaborative business process. By eliminating the need to install and run the application on the customer's own computer, SaaS alleviates the customer's burden of software maintenance, ongoing operation and support. In such a way, the adoption of SOA technology also enables small and medium enterprises (SMEs) to join IE networks, since it promises to reduce costs and complexity for connecting systems and business processes. At the heart of the SOA paradigm there is a catalog of available services that can be shared across IE and reused to build up collaborative processes. Collaboration should be facilitated by guaranteeing a homogeneous description of services at the right level of granularity. A uniform level of granularity enables the comparability among different services. In fact, a coarse-grained service (e.g., a service associated with many tasks of a process) could not be compared with an elementary service (e.g., a service that corresponds to a single task).

We propose the P2S (from process to services) methodology to support the designer of a business process in the identification of the component services. The methodology starts from a process represented by means of a workflow-based language (e.g., BPMN) and supports the designer through the semantic annotation of the business process elements and the identification of component services on the basis of the notion of values produced for the actors that collaborate in the process. The methodology also provides metrics to evaluate the quality of the performed decomposition and to guide the designer in improving the solution found, given the widely accepted definition of services as platform-independent, self-contained, loosely coupled units of work. The P2S methodology should allow collaborative partners to standardize process modelling through component services, enabling effective inter-organizational service discovery.

The paper is structured as follows: in Section 2 basic definitions are provided; in Section 3 the proposed methodology is presented and discussed by means of a running example in a real case scenario; in Section 4, the described approach is compared with related solutions proposed in the literature; finally, Section 5 gives some hints about future work and concludes the paper.

## 2 Basic Definitions

Roughly speaking, a business process  $BP$  can be defined as a combination of a set  $\mathcal{T}$  of simple tasks through control structures (e.g., sequence, choice, cycle or parallel) to form composite tasks, also denoted as sub-processes. Each simple task  $t_i \in \mathcal{T}$  of the business process  $BP$  is described through the operation that it performs and by its I/O data. Data exchanged between tasks and control flows connecting them are modelled as data dependencies and control flow dependencies, respectively. Processes are usually defined at a business level using a workflow-based notation (e.g., BPMN<sup>1</sup>), independently from implementation

<sup>1</sup> <http://www.bpmn.org/>

technology and platforms [2]. As usual, a process has an entry point (start event) and one or more stop events. Furthermore, actors participating in the process must also be considered. Actors are represented as abstract entities that interact with the business process as responsible of one or more simple tasks. Actors can be grouped into organizations. In workflow-based notation, task responsibilities are represented through *swimlanes*. For example, BPMN supports swimlanes with two main constructs: *pools*, to represent organizations participating in the process, and *lanes*, that constitute sub-partitions within pools and are used to organize activities which are logically related to each other (e.g., when they are performed by the same department). We can provide the formal definition of a simple task  $t_i \in \mathcal{T}$  as follows:

$$t_i = \langle n_{t_i}, IN(t_i), OUT(t_i), f_{t_i}, A_{t_i}, O_{t_i}, \{t_{i-1}\} \rangle \quad (1)$$

where:  $n_{t_i}$  is the task name:  $IN(t_i)$  and  $OUT(t_i)$  are the sets of task inputs and outputs, respectively;  $f_{t_i} : IN(t_i) \rightarrow OUT(t_i)$  is the transformation associated with the task;  $A_{t_i}$  and  $O_{t_i}$  are the actor and the organization responsible for the task, respectively;  $\{t_{i-1}\}$  is the set of tasks that precede the task  $t_i$  according to control flow dependencies. According to widely accepted data structure diagrams (e.g., UML, XML complex data types) each input  $in \in IN(t_i)$  can be described as  $\langle n, \mathcal{P} \rangle$ , where  $n$  is the input name and  $\mathcal{P} = \{p_i\}$  a set of properties or attributes that further detail the input data. Outputs can be described in the same manner.

Given two simple tasks  $t_i$  and  $t_j \in \mathcal{T}$ , we say that there is a data dependency from  $t_j$  to  $t_i$  if  $f_{t_i}(OUT(t_j)) \neq f_{t_i}(IN(t_j))$ , that is, execution of  $t_i$  depends on the execution of  $t_j$ . A task  $t_i \in \mathcal{T}$  is defined as *dependent* on another task  $t_j \in \mathcal{T}$  ( $t_j \mapsto t_i$ ) if all the following conditions holds:

1. there is a path of control flow dependencies from  $t_j$  to  $t_i$ , that is, there exists a set  $\{t_k\} \subseteq \mathcal{T}$  of simple tasks, for  $k = 1, \dots, n$ , such that  $t_j \equiv t_1$ ,  $t_i \equiv t_n$  and, for each  $t_k, t_{k-1} \mapsto t_k$ ;
2. there is a data dependency directly from  $t_j$  to  $t_i$ .

We introduce the *task dependency graph*  $\mathcal{T}DG$  as  $\langle \mathcal{T}, \mathcal{T}D \rangle$ , where  $\mathcal{T}$  is the set of simple tasks and  $\mathcal{T}D : \mathcal{T} \times \mathcal{T}$  is the set  $\{t_j, t_i\}$  of task dependencies such that  $t_j \mapsto t_i$ .

Considering cooperation among actors that participate in the process, we define the *data exchange graph*  $\mathcal{D}EG = \langle \mathcal{A}, \mathcal{E} \rangle$ , where  $\mathcal{A}$  is the set of actors,  $\mathcal{E} : \mathcal{A} \times \mathcal{A} \times Data$  is the set  $\{\langle A_i, A_j, d \rangle\}$  of data exchanges between actors,  $Data$  is the union set of inputs and outputs of the simple tasks in the overall process and  $d \in Data$  represents a data value that is transferred from actor  $A_i$  to actor  $A_j \neq A_i$ . Exchanged data are those associated with control flows crossing swimlanes boundaries. Among actors we also include the final user, that is the beneficiary of the overall process execution.

Also a service  $\mathcal{S}$  can be defined as a collection of tasks. However, the definition of a service imposes a series of additional constraints with respect to that of a business process: (i) services are self-contained units of work (that is, they do not require context or state information of other services) and are connected each other using standard, dependency reducing, decoupled message-based elements

such as XML document exchanges; (ii) each service takes one or more inputs and creates an output perceived as a tangible value for the service requester. More specifically, in [3] a service is recognized as a recurrent communication pattern, where the service requester sends a request and receives a response that is of value for the requester himself/herself. On the other hand, the provider can invoke other services to execute his/her tasks, becoming requester for those services. Therefore, the overall business process can be viewed as a complex structure of nested and chained service invocations connected by control structures. The aim of P2S methodology is to support the process designer in a semi-automatic identification of component services, as well as to give metrics for evaluating the decomposition that has been identified.

### 2.1 Case Study

As a motivating example, we present a case study based on the cooperative scenario in which a sofa manufacturer produces all the textiles sofa components and purchases backbones from trusted suppliers (Figure 1). Different actors belonging to the sofa manufacturer enterprise have an active role in the analyzed process. In details, administrative activities are performed by the sales and purchasing offices, while production and delivery activities are performed by the manufacturing and shipping departments. Among the participating actors, the final user that sends the order and receives the sofa is implicitly considered.

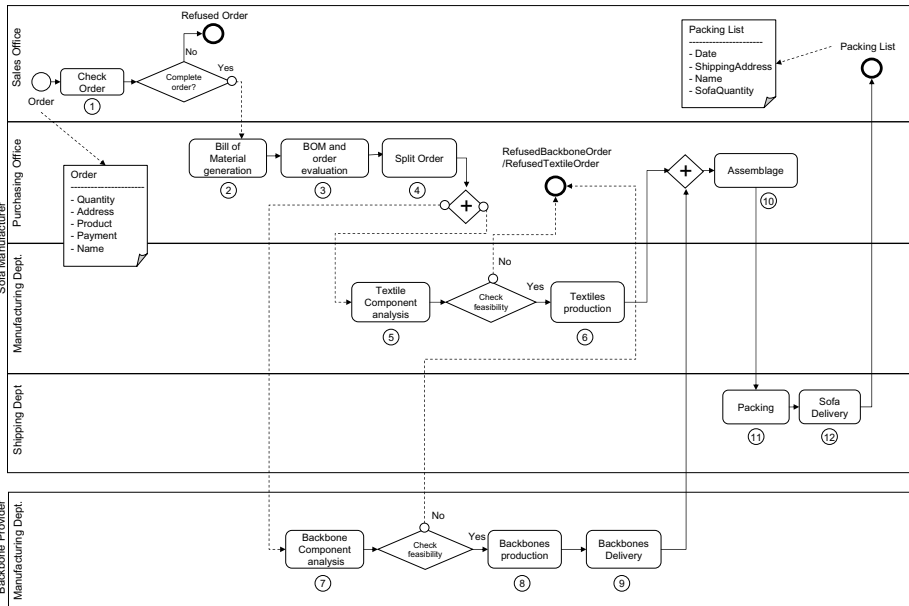


Fig. 1. Business process used as running example

**Table 1.** Complete task representation (details about I/O have been omitted)

Name	Input	Output
1. Check order	Order	CompletenessEvaluation
2. BoM Generation	Order	BoM
3. BoM and order Evaluation	Order, BoM	SuitableProviders, Schedule
4. SplitOrder	Order, BoM, SuitableProviders, Schedule	BackboneComponentOrder, TextileComponentOrder
5. Textile Component Analysis	TextileComponentOrder	OrderFeasibility
6. Textile Production	TextileComponentOrder	TextileComponent
7. Backbone Component Analysis	BackboneComponentOrder	OrderFeasibility
8. Backbones Production	BackboneComponentOrder	BackboneComponent
9. Backbones Delivery	BackboneComponent	DeliveredBackboneComponent
10. Assemblage	TextileComponent, DeliveredBackboneComponent	Sofa
11. Packing	Sofa	Pack
12. Delivery	Pack	PackingList

Name	Operation	Organization.Actor
1. Check order	CheckCompleteness (Order, Constraints)	SofaManufacturer.SalesOffice
2. BoM Generation	BoMDefinition (Order)	SofaManufacturer.PurchasingOffice
3. BoM and order Evaluation	Planning (Order, BoM, ProvidersList, ProductionPlan)	SofaManufacturer.PurchasingOffice
4. SplitOrder	SplitOrder (Order, BoM, SuitableProviders, Schedule)	SofaManufacturer.PurchasingOffice
5. Textile Component Analysis	Evaluate (TextileComponentOrder, ProductionPlan)	SofaManufacturer.ManufacturingDept
6. Textile Production	TextileProduction (RawMaterials)	SofaManufacturer.ManufacturingDept
7. Backbone Component Analysis	Evaluate (BackboneComponentOrder, ProductionPlan)	BackboneProvider.ManufacturingDept
8. Backbones Production	BackboneProduction (RawMaterials)	BackboneProvider.ManufacturingDept
9. Backbones Delivery	Delivery (BackboneComponent)	BackboneProvider.ManufacturingDept
10. Assemblage	Assemblage (TextileComponent, BackboneComponent)	BackboneProvider.ManufacturingDept
11. Packing	Packing (Sofa)	SofaManufacturer.ShippingDept
12. Delivery	Delivery (Pack)	SofaManufacturer.ShippingDept

The event that activates the process is the reception of an order by the sales office. The office checks the order and refuses it if it is affected by incompleteness or inaccuracy problems. If the order is well defined, then the sales office forwards it to the purchasing office, that is responsible for the relationships with raw materials and components providers. The purchasing office generates the Bill of Material (BoM) and evaluates it together with the received order to identify the required components and the providers to contact. Thus, the order is split in sub-orders for each component required. In the example, two sub-orders are created and sent to the internal manufacturing department for the textile production and to an external provider for the backbone production. In both cases, the production units check the received document: if the order is considered as feasible, they start with the production phase and, at the end of the production step, they commit the delivery of the realized components. The assemblage of the components and thus the realization of the final product is in charge of the purchasing office. The process finishes when the shipping department receives the product and delivers it to the final user.

The most of the workflow-based notation tools allow the designer to represent the process in terms of tasks and control structures. Information about the data flow is not provided. In order to identify services, a preliminary step before applying the P2S methodology is the completion of the task representation with the definition of inputs and outputs, the associated operation and responsible actors and organizations (Equation 1). The completed representation of tasks in the running example is summarized in Table 1. Each input/output can be associated to a complex type definition, that includes the I/O name and the list of properties/attributes. For example, Figure 1 also shows the definitions of Order and PackingList information at the beginning and at the end of the process.

According to basic definitions given in Section 2 and task descriptions in Table 1, the task dependency graph and the data exchange graph for the running example are the following:

$$\begin{aligned}
 TDG &= \langle T, TD \rangle \\
 T &= \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12\} \\
 TD &= \{ \langle 2, 3 \rangle, \langle 2, 4 \rangle, \langle 3, 4 \rangle, \langle 4, 5 \rangle, \langle 4, 6 \rangle, \langle 4, 7 \rangle, \langle 4, 8 \rangle, \langle 6, 10 \rangle, \langle 8, 10 \rangle, \langle 10, 11 \rangle, \langle 11, 12 \rangle \} \\
 DEG &= \langle \mathcal{A}, \mathcal{E} \rangle \\
 \mathcal{A} &= \{ \mathcal{A}_0 = \text{FinalUser}, \mathcal{A}_1 = \text{SofaManufacturer.SalesOffice}, \mathcal{A}_2 = \text{SofaManufacturer.PurchasingOffice}, \\
 &\mathcal{A}_3 = \text{SofaManufacturer.ManufacturingDept}, \mathcal{A}_4 = \text{SofaManufacturer.ShippingDept}, \mathcal{A}_5 = \text{BackboneProvider.ManufacturingDept} \} \\
 \mathcal{E} &= \{ \langle \mathcal{A}_0, \mathcal{A}_1, \text{Order} \rangle, \langle \mathcal{A}_1, \mathcal{A}_0, \text{RefusedOrder} \rangle, \langle \mathcal{A}_1, \mathcal{A}_2, \text{Order} \rangle, \langle \mathcal{A}_2, \mathcal{A}_5, \text{BackboneComponentOrder} \rangle, \\
 &\langle \mathcal{A}_2, \mathcal{A}_3, \text{TextileComponentOrder} \rangle, \langle \mathcal{A}_5, \mathcal{A}_2, \text{RefusedBackboneOrder} \rangle, \langle \mathcal{A}_3, \mathcal{A}_2, \text{RefusedTextileOrder} \rangle, \\
 &\langle \mathcal{A}_5, \mathcal{A}_2, \text{DeliveredBackboneComponent} \rangle, \langle \mathcal{A}_3, \mathcal{A}_2, \text{DeliveredTextileComponent} \rangle, \langle \mathcal{A}_2, \mathcal{A}_4, \text{Sofa} \rangle, \langle \mathcal{A}_4, \\
 &\mathcal{A}_0, \text{PackingList} \rangle \}
 \end{aligned}$$

### 3 Methodology

According to the definitions given in Section 2, we consider the business process as the combination of *component services* that execute the set of simple tasks. The proposed methodology (see Figure 2), that guides the business process designer in the identification of component services, is organized into four main phases:

1. *semantic process annotation* - in a distributed heterogeneous environment, where different SMEs provide independently developed process representations, business process elements (inputs and outputs, task names) must be semantically annotated with concepts extracted from shared ontologies; in fact, process designer could use synonymies or terms that are semantically related to ontological concepts and tools and methods to figure out semantic similarities between terms should also be proposed to uniform adopted terminology;
2. *identification of candidate services* - according to an external perspective, process actors involved in the task execution and the values they expect from the business process are identified to determine a first set of candidate



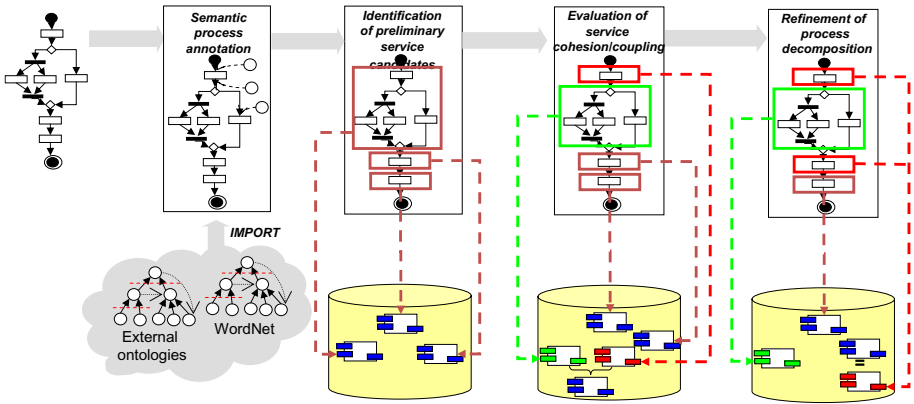


Fig. 2. Methodology phases

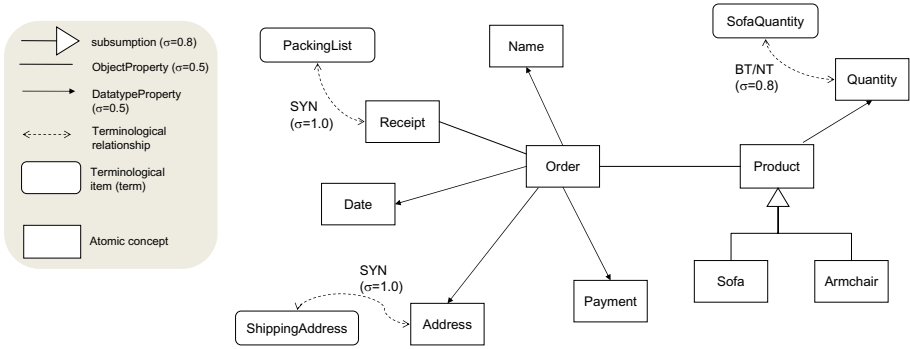
component services; in particular, the data exchange graph will be exploited to figure out service invocations and value exchanges between requesters and providers;

3. *evaluation of service cohesion/coupling* - according to an internal perspective, evaluation of service dependencies in terms of cohesion/coupling criteria is useful to better define service structure and granularity; to this aim, the task dependency graph will be exploited;
4. *refinement of process decomposition* - an additional phase is performed to detect multiple invocations of the same service throughout the process workflow by means of proper coefficients; such coefficients will be applied to each pair of identified services to check if they perform the same operations or operate on the same data.

Methodological phases will be detailed in the following sections, with reference to the motivating example.

### 3.1 Phase 1: Semantic Process Annotation

We suppose that SMEs aiming at collaborating through their business activities agree on a shared reference ontology  $\mathcal{O} = \langle \mathcal{C}, \mathcal{R} \rangle$ , where  $\mathcal{C}$  is the set of ontological concepts (atomic concepts) and  $\mathcal{R}$  is the set of semantic relationships (e.g., equivalence, subsumption) between concepts. In the semantic process annotation phase, after the completion of the task representation as shown in Section 2.1, concepts names are associated with business process elements (i.e., task names, input and output names and properties). Since terms used for business process elements do not necessarily coincide with atomic concepts, the reference ontology is extended with pre-existing, domain independent terminological knowledge extracted from an underlying lexical system (e.g., WordNet [4]), that relates each term that is not already included in the reference ontology to atomic concepts by means of synonymy (SYN), broader/narrower term (BT/NT) and related



**Fig. 3.** A portion of shared ontology for the running example

term (RT) relationships. This approach is common and it has been successfully adopted in other contributions [5,6]. A weight  $\sigma \in (0, 1]$  is associated with each kind of relationship. For example,  $\sigma = 1.0$  for equivalence or synonymy relationships,  $\sigma = 0.8$  for subsumption or BT/NT relationships,  $\sigma = 0.5$  for the other kinds of relationships. Two generic terms  $n_1$  and  $n_2$  (either atomic concepts or not) can be related by a chain of weighted relationships. We define the *name affinity* value between  $n_1$  and  $n_2$  (denoted with  $NAff(n_1, n_2)$ ) as the product of weights associated to the chain of relationships that relates  $n_1$  to  $n_2$ . During the semantic annotation phase, the name affinity evaluation between a term  $n_1 \notin \mathcal{C}$  adopted as business process element name and the atomic concepts in  $\mathcal{C}$  is used to suggest to the designer the ontological concepts that better match with  $n_1$ . Moreover, given two data items  $d_1 = \langle n_1, \mathcal{P}_1 \rangle$  and  $d_2 = \langle n_2, \mathcal{P}_2 \rangle$  (either input or output of simple tasks), we define the *structural affinity*  $SAff(d_1, d_2)$  as the evaluation of their semantic similarity, that is:

$$SAff(d_1, d_2) = \frac{1}{2} \cdot \left[ NAff(n_1, n_2) + \frac{2 \cdot \sum_{p_1, p_2} NAff(p_1, p_2)}{|\mathcal{P}_1| + |\mathcal{P}_2|} \right] \quad (2)$$

where  $p_1 \in \mathcal{P}_1, p_2 \in \mathcal{P}_2, |\mathcal{P}_i|$  is the cardinality (that is, the number of elements) of the set  $\mathcal{P}_i$  of properties. For the running example, let consider the descriptions of **Order** and **PackingList** data items represented in Figure 1 and the portion of reference ontology and domain-independent terminological knowledge shown in Figure 3. The following affinity values can be evaluated:

$$\begin{aligned} NAff(\text{Order}, \text{PackingList}) &= 1.0 * 0.5 = 0.5 \\ NAff(\text{Quantity}, \text{SofaQuantity}) &= 0.8 \\ NAff(\text{Address}, \text{ShippingAddress}) &= 1.0 \\ SAff(\text{Order}, \text{PackingList}) &= \frac{1}{2} \left[ 0.5 + \frac{2 * (0.8 + 1.0 + 1.0)}{9} \right] = 0.56 \end{aligned}$$

$Aff_{TOT}(D_1, D_2)$  is the total structural affinity between two sets of data items and is defined as the sum of structural affinity for each pair of items  $d_1 \in D_1$  and

$d_2 \in D_2$ , normalized with respect to the cardinality of  $D_1$  and  $D_2$ . The  $Aff_{TOT}$  coefficient will be exploited in the next phases of the methodology. Note that the proposed use of domain independent terminological knowledge can be exploited to bridge the gap between different reference ontologies (as shown in [7]), thus avoiding to constrain collaborating partners to adhere to the same reference ontology.

### 3.2 Phase 2: Identification of Candidate Services

For the identification of candidate services that are combined in a business process, we propose some heuristics that are derived from empirical observations about the features and the definition of a service found in literature. Services constitute units of work that are logically decoupled. Workflow-based tools and languages usually collect in the same swimlane (either pool or lane in BPMN process representation) logically coupled sets of tasks. Moreover, services are invoked by actors participating to the process to obtain tangible values from other actors. We exploit the data exchange graph defined in Section 2 to identify value exchanges. According to definitions given in [3], a value for one of the process actors can be associated to a service invocation request. For each node  $A$  of the data exchange graph (that is, an actor), the outgoing and incoming edges (that is, data transfers towards and from other actors, respectively) are considered as service requests/invocations and responses/values, respectively. The total structural affinity evaluation is performed for each pair  $\langle Dreq, Dres \rangle$ , where  $Dreq$  is a data item associated with one of the outgoing edges and  $Dres$  is a data item associated with one of the incoming edges. Only pairs such that  $Aff_{TOT}(Dres, Dreq) > 0$  are maintained, that correspond to the service invocation and the corresponding value produced through the service execution. Three cases should be distinguished: (i) an outgoing data value  $Dreq$  is not associated with any incoming data, that is,  $\langle Dreq, \emptyset \rangle$ ; (ii) a match is found between outgoing data  $Dreq$  and incoming data  $Dres$ , that is,  $\langle Dreq, Dres \rangle$ ; (iii) an incoming data value  $Dres$  is not associated with any outgoing data, that is,  $\langle \emptyset, Dres \rangle$ . The first and the third cases seem correspond to borderline situations, in which an actor sends a request without receiving any value from service invocation or an actor receives a service response/value without sending any request, respectively. This could be due to the impossibility for the supporting tool to find out semantic correspondences between outgoing and incoming data items; in these cases, the process designer must intervene to explicitly set, modify or exclude this kind of matches, according to his/her own domain or process knowledge. The second case corresponds to the invocation of a service, that produces the corresponding value. This leads to the following heuristics. For each process actor that is associated with a pair  $\langle Dreq, Dres \rangle$ , a candidate service  $\mathcal{S}$  is recognized whose first task  $t_i$  is such that  $IN(t_i) \equiv Dreq$  and whose last task  $t_j$  is such that  $OUT(t_j) \equiv Dres$ . In particular, the service  $\mathcal{S}$  is invoked from the service (if exists) containing the task  $t_k$  such that  $OUT(t_k) \equiv Dreq$ .

The list of candidate services is proposed to the process designer, that can validate or refuse them. Let consider the sofa manufacturer example. Applying

the heuristics described above, the following pairs  $\langle \mathcal{D}req, \mathcal{D}res \rangle$  are identified, with corresponding candidate services:

- $\langle \text{Order,RefusedOrder} \rangle \Rightarrow \mathcal{S}_1 = \{t_1\}$  (requester: final user)
- $\langle \text{Order,PackingList} \rangle \Rightarrow \mathcal{S}_2 = \{t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_9, t_{10}, t_{11}, t_{12}\}$  (requester: final user)
- $\langle \text{BackboneComponentOrder,DeliveredBackboneComponent} \rangle \Rightarrow \mathcal{S}_3 = \{t_7, t_8, t_9\}$   
(requester: purchasing office)
- $\langle \text{BackboneComponentOrder,RefusedBackboneOrder} \rangle \Rightarrow \mathcal{S}_4 = \{t_7\}$  (requester: purchasing office)
- $\langle \text{TextileComponentOrder,DeliveredTextileComponent} \rangle \Rightarrow \mathcal{S}_5 = \{t_5, t_6\}$   
(requester: purchasing office)
- $\langle \text{TextileComponentOrder,RefusedTextileOrder} \rangle \Rightarrow \mathcal{S}_6 = \{t_5\}$  (requester: purchasing office)
- $\langle \text{Sofa,PackingList} \rangle \Rightarrow \mathcal{S}_7 = \{t_{11}, t_{12}\}$  (requester: purchasing office)

### 3.3 Phase 3: Evaluation of Service Cohesion/Coupling

Once the candidate services have been identified as set of tasks, services are further analyzed in terms of cohesion/coupling criteria in order to better define service structure and granularity. The adopted cohesion/coupling metrics have been inspired by their well-known application in software engineering field [8]. In our scenario, task dependencies inside and across services are identified in terms of common used data and are exploited to better aggregate tasks or to figure out parallelism among them. In this phase, the task dependency graph is exploited.

Given two tasks  $t_i$  and  $t_j$ , we define the task coupling coefficient as follows:

$$\tau(t_i, t_j) = \begin{cases} Aff_{TOT}(OUT(t_j), IN(t_i)) & \text{if } t_j \mapsto t_i \\ Aff_{TOT}(IN(t_j), OUT(t_i)) & \text{if } t_i \mapsto t_j \\ \frac{Aff_{TOT}(IN(t_i), IN(t_j)) + Aff_{TOT}(OUT(t_i), OUT(t_j))}{2} & \text{otherwise} \end{cases} \quad (3)$$

The third member in the equation 3 is used for tasks  $t_i$  and  $t_j$  that are executed on parallel branches, but work on semantically related data. In details, two tasks  $t_i$  and  $t_j$  are considered: (i) *loosely coupled*, if  $0 < \tau(t_i, t_j) < \delta$ , where  $\delta$  is a threshold set by the designer ( $\delta \in [0, 1]$ ); (ii) *strongly coupled*, if  $\tau(t_i, t_j) \geq \delta$ ; (iii) *decoupled*, if  $\tau(t_i, t_j) = 0$ . Given a service  $\mathcal{S}$  identified in phase 2, the service cohesion coefficient is defined as:

$$coh(\mathcal{S}) = \begin{cases} 2 \cdot \frac{\sum_{i,j} \tau(t_i, t_j)}{|\mathcal{S}| \cdot (|\mathcal{S}| - 1)} \quad \forall t_i, t_j \in \mathcal{S} & |\mathcal{S}| < 1 \\ 1 & |\mathcal{S}| = 1 \end{cases} \quad (4)$$

where  $|\mathcal{S}|$  is the number of tasks in  $\mathcal{S}$ . Given two services  $\mathcal{S}_1$  and  $\mathcal{S}_2$ , the coupling coefficient between them is defined as:

$$coup(\mathcal{S}_1, \mathcal{S}_2) = \frac{\sum_{i,j} \tau(t_i, t_j)}{|\mathcal{S}_1| \cdot |\mathcal{S}_2|} \quad \forall t_i \in \mathcal{S}_1 \wedge t_j \in \mathcal{S}_2 \quad (5)$$

Service cohesion and coupling coefficients are used to evaluate the average cohesion and coupling of the overall process  $\mathcal{BP}$ , respectively:

$$pcoh(\mathcal{BP}) = \frac{\sum coh(\mathcal{S}_i)}{|\mathcal{BP}|} \quad (6)$$

$$pcoup(\mathcal{BP}) = \frac{\sum_{i,j} coup(S_1, S_2)}{|\mathcal{BP}| \cdot (|\mathcal{BP}| - 1)} \tag{7}$$

combined in the coupling/cohesion ratio  $\Gamma$ :

$$\Gamma = \frac{pcoup(\mathcal{BP})}{pcoh(\mathcal{BP})} \tag{8}$$

where  $|\mathcal{BP}|$  is the number of services identified in second phase. Task coupling coefficient is used to establish the goodness of the service cohesion and coupling of the set of candidate services identified in the second phase of the methodology. Tasks are aggregated according to their level of coupling by applying a hierarchical clustering algorithm. Firstly, each task constitutes a cluster with only one element (singleton). Two clusters  $C_1$  and  $C_2$  are merged together if there exist two tasks  $t_1 \in C_1$  and  $t_2 \in C_2$  that are coupled (loosely or strongly) or viceversa. Clusters are iteratively merged until a unique cluster is obtained for the overall process or there is no coupling between the tasks of clusters that have been identified. Clusters that present tasks with highest value for  $\tau$  are merged first. At each merging step the coupling/cohesion ratio  $\Gamma$  is evaluated. The best clustering level is the one that minimized the  $\Gamma$  value. After clustering has been applied and optimized, the system proposes to the designer to further split services identified in the previous phase if they contain tasks belonging to different clusters to allow their parallel execution. On the other hand, the process designer may be suggested to merge services if they contain coupled tasks. The designer can evaluate the results and then decide to adopt different business process re-engineering actions.

If we apply the clustering algorithm to the running example, we obtain the tree shown in Figure 4, where, for example, the task  $t_{10}$  is decoupled from the other ones, while from phase 2 it is included with the other simple tasks as part of the  $S_2$  candidate service. The process designer may decide to split this candidate

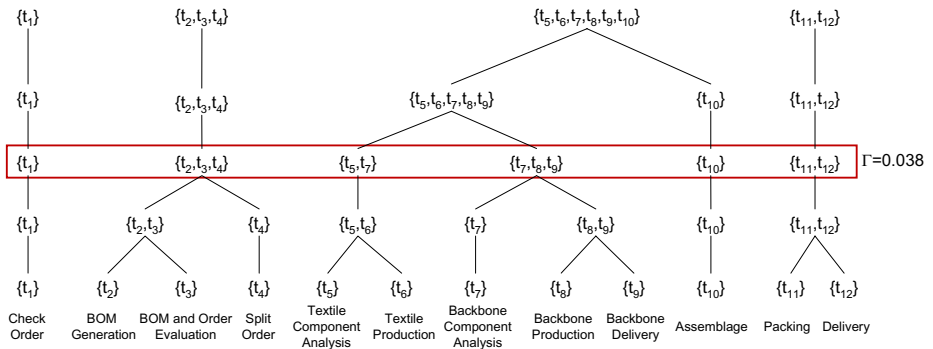


Fig. 4. Clustering of coupled simple tasks

service, thus obtaining three more services  $\mathcal{S}_{21} = \{t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_9\}$ ,  $\mathcal{S}_{22} = \{t_{10}\}$  and  $\mathcal{S}_{23} = \{t_{11}, t_{12}\}$ .

### 3.4 Phase 4: Refinement of Process Decomposition

Services identified in the previous phases are sub-processes constituted by one or more tasks that identify service operations. However, there could be similar services that are invoked multiple times throughout the process. For example, services that check the textile component and backbone component feasibility could be recognized as similar services and, in particular, they could be viewed as different invocations of the same service. To detect possible overlapping services in the business process, coefficients already introduced in [9] are applied. In particular, the *Entity-based similarity coefficient* between two services  $\mathcal{S}_1$  and  $\mathcal{S}_2$ , denoted with  $ESim(\mathcal{S}_1, \mathcal{S}_2)$ , is used to state if  $\mathcal{S}_1$  and  $\mathcal{S}_2$  work on the same data. Denoting with  $\mathcal{S}^{IN}$  and  $\mathcal{S}^{OUT}$  the union sets of inputs and outputs of the tasks in a service  $\mathcal{S}$ , the  $ESim$  coefficient is computed as:

$$ESim(\mathcal{S}_1, \mathcal{S}_2) = Aff_{TOT}(\mathcal{S}_1^{IN}, \mathcal{S}_2^{IN}) + Aff_{TOT}(\mathcal{S}_1^{OUT}, \mathcal{S}_2^{OUT}) \in [0, 2] \quad (9)$$

The *Functionality-based similarity coefficient* between two services  $\mathcal{S}_1$  and  $\mathcal{S}_2$ , denoted with  $FSim(\mathcal{S}_1, \mathcal{S}_2)$ , is used to state if  $\mathcal{S}_1$  and  $\mathcal{S}_2$  perform the same operations.  $FSim$  is based on the *Operation Similarity coefficient* between two tasks  $t_1$  of  $\mathcal{S}_1$  and  $t_2$  of  $\mathcal{S}_2$ , denoted with  $OpSim(t_1, t_2)$  and computed as:

$$OpSim(t_1, t_2) = NAff(n_{t_1}, n_{t_2}) + Aff_{TOT}(IN(t_1), IN(t_2)) + Aff_{TOT}(OUT(t_1), OUT(t_2)) \quad (10)$$

where  $OpSim \in [0, 3]$ , since it is the sum of three elements in the range  $[0,1]$ . The  $FSim$  coefficient is evaluated as:

$$FSim(\mathcal{S}_1, \mathcal{S}_2) = \frac{2 * \sum_{h,k} OpSim(t_h, t_k)}{|\mathcal{S}_1| + |\mathcal{S}_2|} \in [0, 3] \quad (11)$$

Entity-based and functionality-based service similarity coefficients support the designer to recognize multiple invocations of the same service throughout the process workflow.

Considering the running example, it is possible to find some similarities between the services identified in the previous phase. In details, two pairs of services are selected:  $\langle \mathcal{S}_3 = \{t_7, t_8, t_9\}, \mathcal{S}_5 = \{t_5, t_6\} \rangle$ ,  $\langle \mathcal{S}_4 = \{t_7\}, \mathcal{S}_6 = \{t_5\} \rangle$ . The designer will discard the pair  $\langle \mathcal{S}_3, \mathcal{S}_5 \rangle$  since they are characterized by input, output and terms similarities but the task operation has a different implementation in the two services and it is not possible to merge them in a unique component. The pair  $\langle \mathcal{S}_4, \mathcal{S}_6 \rangle$  can be instead considered for a service reconciliation, since both services have as input the same document and on the basis of the production plan use the same algorithm in order to evaluate the order feasibility.

## 4 Related Work

Links between the world of business processes and Web services have been thoroughly analyzed in different contexts and from different perspectives in the literature. Many contributions focus on the analysis of existing Web services and

address the issue of their composition to obtain the desired process. Nowadays, organizations more and more implement collaborative businesses through component services over the Internet.

In the literature, in order to bridge the link between services and business processes some contributions consider existing Web services and define enriched process representation. In particular, they start from a business process that is manually represented as a composition of services semantically enriched with ontologies. In [10] an abstract process represents a Web process whose control and data flow are defined at design time, but the actual services are not chosen till at run-time. Run-time service selection can be automated with the semantic representation of the knowledge of the domain experts in ontologies and rules (semantic Web processes). In [11] the notion of process template is introduced. Process templates are reusable business process skeletons that are devised to reach particular goals and are made up of states and transitions. A state corresponds to the execution of a service (called component service) that is member of a Web service community. A community is a collection of services with a common functionality, but different non functional properties such as different QoS parameters, that are exploited to select the right service at run-time. These approaches are especially useful in environments where component services are relatively fixed, while the methodology presented in this paper does not require any knowledge about existing services.

Moreover, there are approaches that attempt to assist service providers and service aggregators in multi-party business processes [12,13,14]. In particular, [13] discusses how business process should be described so that services can be properly identified and provides strategies and principles regarding functional and non-functional aspects of Web service design. Furthermore, in [14] authors define a methodology that aims at defining a foundation of development principles for Web services based on which business processes can be assembled into business scenarios. In [12] a goal-based approach for the identification of service composition is proposed. All these approaches provide guidelines for the service identification without giving an operational support. The use of coupling and cohesion metrics to evaluate process decomposition into sub-processes or activities has been suggested in [8,15,16], but these approaches mainly propose techniques to compare different decompositions to choose the best one. This issue also relates to existing contributions in literature to support the transformation of legacy applications into component services [17,18,19]. For example, [17] describes the Service-Oriented Migration and Reuse Technique (SMART) as a technique that helps organizations to analyze legacy systems to determine whether their functionalities, or a subsets of them, can be reasonably exposed as services in a Service-Oriented Architecture. Other valuable guidelines in this context are provided in [18,19]. Moreover, a considerable number of research efforts have been devoted to the composition of services both in academia and industry [20,21,22]. However, all these valuable contributions do not offer operational support for the identification of process tasks that can be aggregated and exposed as services. For this purpose, the methodology described in this paper

tries to uniform component services identification at the same granularity, in order to improve and speed up the following discovery phase. To the best of our knowledge, there are no approaches in literature that propose a semi-automatic methodology that supports the designer in identifying the component services in a business process.

## 5 Conclusion

The methodology presented in this paper aims at constituting a semi-automatic approach for the identification of the subset of functionalities that can be exported as services to implement a collaborative business process. The methodology starts from a process represented by means of a workflow-based language (e.g., BPMN) and supports the designer for the identification of component services. Future work will focus on the design of a CASE tool able to support the process designer through the phases of the methodology. Currently, single modules that implement the methodological phases have been developed and it is necessary to integrate them and properly test the resulting application. Additional features of the developed system will be investigated to deal with a multi-knowledge environment, where the system relies on distinct reference ontologies and different process representation languages are used and must be integrated. Finally, the proposed methodology will be further studied in the field of service orchestration and composition to implement efficient solutions.

## Acknowledgements

This work has been partially supported by the TEKNE (Towards Evolving Knowledge-based internetworked Enterprise) FIRB Project (<http://www.tekne-project.it/>), founded by the Italian Ministry of Education, University and Research.

## References

1. O'Brien, J.A.: Introduction to Information Systems: Essentials for the Internetworked Enterprise. McGraw-Hill Education, New York (2000)
2. Chang, S.H., Kim, S.D.: A service-oriented analysis and design approach to developing adaptable services. In: Krämer, B.J., Lin, K.-J., Narasimhan, P. (eds.) ICSOC 2007. LNCS, vol. 4749. Springer, Heidelberg (2007)
3. Dietz, J.L.G.: The atoms, molecules and fibers of organizations. Data & Knowledge Engineering (2003)
4. Fellbaum, C.: Wordnet: An Electronic Lexical Database. MIT Press, Cambridge (1998)
5. Pedersen, T., Patwardhan, S., Michelizzi, J.: Wordnet:Similarity - Measuring the Relatedness of Concepts. In: Proc. of Nineteenth Conf. of Artificial Intelligence (AAAI 2004), Intelligent Systems Demonstration, San Jose, CA, pp. 1024–1025 (2004)



6. Corley, C., Mihalcea, R.: Measuring the Semantic Similarity of Texts. In: Proc. of the ACL Workshop on Empirical Modeling of Semantic Equivalence and Entailment, Ann Arbor, Michigan, pp. 13–18 (2005)
7. Bianchini, D., De Antonellis, V., Melchiori, M.: Flexible Semantic-based Service Matchmaking and Discovery. *World Wide Web Journal* 11(2), 227–251 (2008)
8. Vanderfeesten, I., Reijers, H., van der Aalst, W.: Evaluating workflow process designs using cohesion and coupling metrics. *Computer in Industry* 59(5), 420–437 (2008)
9. Bianchini, D., De Antonellis, V., Pernici, B., Plebani, P.: Ontology-based methodology for e-service discovery. *Journal of Information Systems, Special Issue on Semantic Web and Web Services* 31(4-5), 361–380 (2006)
10. Mulye, R., Miller, J., Verma, K., Gomadam, K., Sheth, A.: A semantic template based designer for Web processes. In: Proc. of 2005 IEEE Int. Conf. on Web Services (ICWS 2005), Orlando, Florida, USA, pp. 461–469 (2005)
11. Sheng, Q., Benatallah, B., Maamar, Z., Dumas, M., Ngu, A.: Enabling Personalized Composition and Adaptive Provisioning of Web Services. In: Persson, A., Stirna, J. (eds.) CAiSE 2004. LNCS, vol. 3084, pp. 322–337. Springer, Heidelberg (2004)
12. Kaabi, R.S., Souveyet, C., Rolland, C.: Eliciting service composition in a goal driven manner. In: Proc. of the 2nd Int. Conf. on Service Oriented Computing, New York, NY, USA, pp. 308–315 (2004)
13. Papazoglou, M.P., Yang, J.: Design Methodology for Web Services and Business Processes. In: Buchmann, A., Casati, F., Fiege, L., Hsu, M.-C., Shan, M.-C. (eds.) TES 2002. LNCS, vol. 2444, p. 54. Springer, Heidelberg (2002)
14. Papazoglou, M.P., van den Heuvel, W.J.: Business process development life cycle methodology. *Communications of ACM* 50(10), 79–85 (2007)
15. Castano, S., De Antonellis, V., Melchiori, M.: A Methodology and Tool Environment for Process Analysis and Reengineering. *Data and Knowledge Engineering* 31(3), 253–278 (1999)
16. Baresi, L., Casati, F., Castano, S., Fugini, M., Mirbel, I., Pernici, B.: WIDE Workflow Development Methodology. In: Proc. of Int. Joint Conf. on Work Activities Coordination and Collaboration, pp. 19–28 (1999)
17. Lewis, G., Morris, E., O'Brien, L., Smith, D., Wrage, L.: SMART: The Service-Oriented Migration and Reuse Technique. Technical Note CMU/SEI-2005-TN-029, Carnegie Mellon University, Software Engineering Institute (2005)
18. Lawrence, C.: Adapting legacy systems for SOA. Technical report, IBM (2007)
19. Microsoft: The Business Value of Legacy Modernization. Technical report, Microsoft (2007)
20. Baresi, L., Bianchini, D., De Antonellis, V., Fugini, M., Pernici, B., Plebani, P.: Context-aware Composition of e-services. In: Proc. of Fourth VLDB Workshop on Technologies for E-Services (TES 2003), Humboldt-University zu Berlin, Germany, pp. 49–58 (2003)
21. Benatallah, B., Sheng, Q.Z., Dumas, M.: The Self-Serv environment for Web services composition. *IEEE Internet Computing* 7(1), 40–48 (2003)
22. Rao, J., Su, X.: A Survey of Automated Web Service Composition Methods. In: Cardoso, J., Sheth, A.P. (eds.) SWSWPC 2004. LNCS, vol. 3387, pp. 43–54. Springer, Heidelberg (2005)

# Composing Time-Aware Web Service Orchestrations

Horst Pichler, Michaela Wenger, and Johann Eder

University of Klagenfurt, Department of Informatics-Systems, Austria\*

**Abstract.** Workflow time management deals with the calculation of temporal thresholds for process activities, which allows forecasts about looming deadline violations. We present a novel approach to transform a web service orchestration into a time-aware orchestration, that contains temporal assessment and intervention logic. During process execution intervention strategies are triggered pro-actively to speed up a late process and to avoid upcoming violations of temporal constraints.

## 1 Introduction

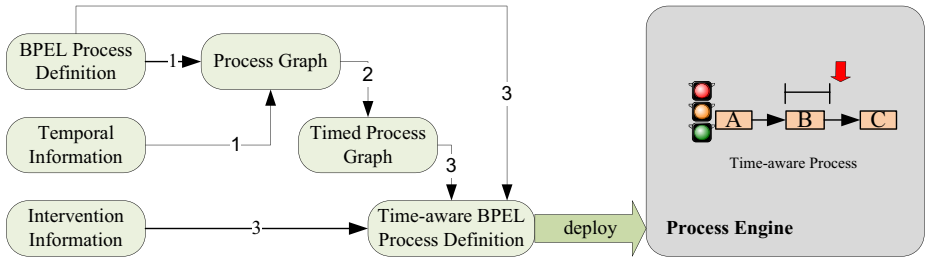
Web service orchestrations are used to assemble processes from external processes and web-services to implement business processes. Expected process execution times and compliance to agreed upon deadlines rank among the most important quality measures [5,11]. To speed up processes and decrease the number of deadline violations should therefore be among the major objectives of business process management. This can be achieved by the application of workflow time management [17]. It deals with temporal aspects of time-constrained processes and aims at optimized, timely, and violation-free process execution. Based on explicit knowledge about process structure, activity durations, and deadlines it is possible to calculate temporal thresholds (internal deadlines) for each activity of a process. During run time, these thresholds are utilized to monitor the progress, forecast looming deadline violations, and to pro-actively trigger intervention strategies which speed up the remainder of the process, like skipping of optional activities, choosing alternative shorter paths, substituting activities, etc.

Time management approaches have mainly dealt with modelling of temporal aspects, checking satisfiability of temporal constraints, scheduling, and so on. The basic concepts of intervention strategies have been described (see Section 2), but how to model, implement, and apply them on process definitions and within process instances is still open. Suggested realization of time management functionality requires new process definition elements and additional logic within the process engine.

Our main goal is to close the gap between build and run-time concepts and enable time management for long-running web service orchestrations. The novel contribution of our approach is that we do not propose an extension of the process enactment service but make the process itself time aware and capable of triggering pro-active measures. So time-aware information systems can be realized without waiting for vendors to complement their process enactment services with up-to-date time management capabilities.

---

\* Part of this work has been supported by EU Commission within the FET-STREP Project WS-Diamond.



**Fig. 1.** Generation of a Time-aware Process Definition

The architecture of our approach is shown in Figure 1. Inputs to the process are a process model (in our realization a BPEL process definition), a representation of explicit temporal information (durations and deadlines) and a representation of intervention strategies. From the process model, augmented with temporal information, we calculate particular internal deadlines for all activities. Then we use this timed process graph and the intervention information to extend the process model with 2 aspects: (a) temporal state assessment and (b) intervention logic. Temporal state assessment monitors the execution of the process and compares the execution times with pre-calculated thresholds. If the process is not running within its tolerance, exceptions are raised. The intervention logic reacts to these exceptions and enables changes to the original process logic with the goal to regain a safe execution state. This results in an extended time-aware process with self-healing capabilities which can then be executed on any enactment service of that process model type.

To be more concrete we instantiated this general approach for WS-BPEL. However, this approach can be easily ported to any commercial workflow system, which supports exceptions and timed triggers in its proprietary control flow language. Some more assumptions: we focus on long-running processes with asynchronous communication structures that take hours, days or even weeks, where the execution time of the process logic is negligible compared to the processing time of the individual activities.

The remainder of the paper is organized as follows: Section 2 provides an overview of related work and describes time management basics. Section 3 gives an overview of elements in a BPEL process definition and shows how to transform it into a graph representation, a prerequisite for the calculation of the timed graph in Section 4. In Section 5 we show how to define intervention and state assessment information. Section 6 describes how to transform the original definition into a time-aware process definition, based on the timed graph and intervention information. Finally we discuss our prototype and complexity in Section 7 and conclude the paper with Section 8.

## 2 Related Work

Workflow time management architectures, as sketched in [9] or [14], consist of several components. [9] proposes a *build time component* that takes a control flow model and temporal information about activity duration and time constraints as input, and

calculates thresholds for each activity in the process. The calculation of thresholds is frequently rooted in techniques based on temporal constraint networks to model and verify temporal information [12,19], others utilize project planning methods [13,11,19]. All these techniques apply variants of interval-timed graph-based models, which can also be used to calculate schedules for workflow execution. A node represents an activity or its start/end-event, edges represent precedence, constraints between nodes, and intervals are used to describe valid time frames for the execution of an activity or occurrence of an event. In this paper we adopted the approach of [10]. They extend the temporal model of [11] for asynchronous messaging patterns, an important prerequisite in web service environments (see also [18]). During run time a *prediction component* [9] correlates and compares time stamps of start and end-events with precalculated thresholds of corresponding activities. Based on this comparison the temporal state can be assessed, for example with the traffic-light model introduced in [17], or the duration and instantiation space model introduced in [15]. Delays, caused by unexpected waiting times or longer execution durations, will change the temporal state of the process. According to this state a *proactive component* [9] has to select intervention strategies that trigger actions within the process engine to speed up the process. Proactive intervention strategies, aiming at speeding up the process, may be applied according to the current temporal state to avoid looming deadline violations.

In this paper we apply intervention strategies that can be realized by implementing them within the process definition, like skipping optional tasks, execution of alternative paths or the parallel execution of sequential activities (see [22,11,8,16] for basic descriptions). Furthermore, we utilized an adaptation of the technique described in [6]: early escalation, which terminates a late process immediately, if the cost of finishing it, is higher than immediate escalation followed by termination. More escalation strategies, which can not be realized by changing the process definition, can be found in [8]. This includes load balancing strategies like resource redeployment (e.g., add resource capacity) or grouping similar tasks to batches to decrease the task preparation time. The closest approach to our's is [8] where time-awareness is also integrated into the process definition itself, by extending it with hard-coded conditional structures. Such a structure could for example be "if (process late) then {perform two reviews} else {perform three reviews}". Naturally, the statement "process late" must be specified as a state-assessment expression, that compares time stamps of events (start or end of an activity) with a precalculated thresholds. The disadvantages of such an approach are obvious. Defining the process gets more complex as the designer has to specify state assessment mechanisms and temporal exception handling parts. New duration estimations or changed deadlines alter the thresholds, resulting in manual adaptations of the assessment expressions. Furthermore, intervention strategies for late processes may be added, changed or removed, which alters the process structure. And finally, with such a "passive" approach it is not possible to handle activities that block process execution.

Specific temporal aspects of web service environments were examined in, e.g., [20,21]. [20] uses temporal abstractions of business protocols in a finite state machine formalism and [21] exploits an extension of a timed automata formalism for modeling time properties of web services. However, these approaches aim primarily at service compatibility, therefore proved to be unsuitable for our purposes.

### 3 Process Representations

In this section we describe the elements of BPEL (the process definition language of our realization), show how to transform it to a graph representation and augment it with temporal information, and describe subsequent timed graph calculations.

#### 3.1 BPEL Representation

The original process is defined in WS-BPEL, also known as BPEL 2.0, which provides the following elements:

**Declarative Elements** to specify the environment of the process, like links to web services (*partner links*), process variables, or *correlation sets*.

**Basic Activities** for the communication with web services: invoking or sending a message to a web service with *invoke*; *receive* a message from an external service; send a *reply* to an external service (as answer to a prior receive). Furthermore, activities to *assign* a value to a variable, delaying execution with *wait*, doing nothing with *empty*, and explicit termination with *exit*.

**Structured Activities** which define the control flow of the process by nesting of basic or structured activities: *sequence* for sequentially executed activities; *if* for conditional exclusive execution of activities; *flow* for parallel execution; *while*, *repeatUntil* and *forEach* for iterative execution; and *pick* for conditional execution based on the type of a received message. Additionally it is possible to declare *event* and *fault-handlers* within *scopes*, which handle thrown faults and raised events.

Some activity-types delay process execution because they wait for something (e.g., an incoming message). Therefore we consider the basic activities *receive* and *pick* as blocking. Furthermore, every structured activity is considered as blocking, if it contains a blocking basic or structured activity. The left-hand side of Figure 2 shows the skeleton of a BPEL process definition. To identify specific activities within a process definition, we use the *name*-attribute, an attribute that can be added to any BPEL-element. Note that this paper focuses on the regular flow in block-structured processes, therefore we do not consider flows with links and exception handlers.

#### 3.2 Process Graph

According to [10] and as visualized in Figure 2, a *process graph* consists of named nodes (rectangles with labels) connected by edges, which describe precedence constraints (solid arrows). Each node has a type (label above or below a node), which can either be activity (act) or an opening or closing node-type for structured elements: sequential execution of elements (seq-start, seq-end), 1-out-of-n exclusive conditional execution (xor-split, xor-join), parallel execution of a several paths (and-split, and-join), and the process itself (proc-start, proc-end). Furthermore, asynchronous communication relationships between invoking and receiving activities are represented as dashed arrows, augmented with service response times (angle-brackets on top of dashed arrows). The example process contains several nested structures: a sequence *s0* that executes *a-i*, followed by an if-conditional element *i*, which selects either *b-i* and *b-r* or *c-i*

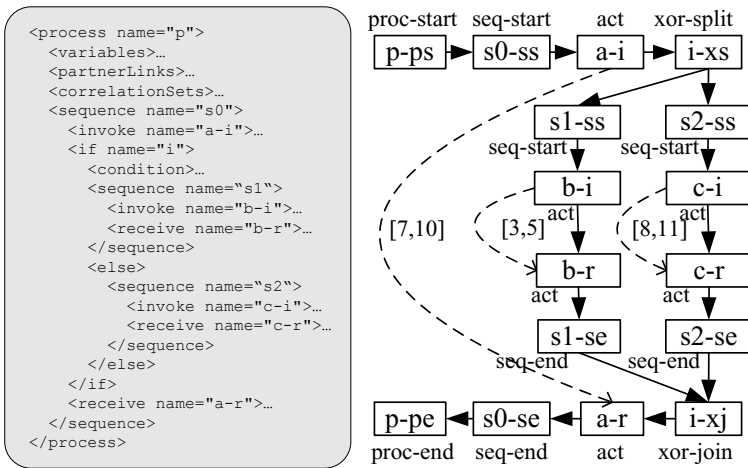


Fig. 2. BPEL process and Graph-representation - Generation Step 1

and  $c-r$ , followed by a receiving  $a-r$ . Although the details are not specified in this shortened example, assume, that  $a-i$  asynchronously sends a message to an external service, whose response is received by  $a-r$ ; furthermore,  $b-i$  and  $c-i$  send messages to services, whose response is received by  $b-r$  and  $c-r$ . Due to space limitations we omitted the XML-representation of the process graph and temporal information, which we used in our prototypical implementation.

### 3.3 Temporal Information

Additionally we need explicit *temporal information*, which are a maximum process duration (assume [15,15] for our running example) and response times of asynchronous relationships between invoking and receiving nodes. We apply [min, max]-intervals for temporal information (durations, deadlines), given in a specified time-unit, which will be days or hours (as applied in the running example) for long running processes. Expected service response times may stem from empirical knowledge (extracted from logs) or be estimated by experts. Especially when dealing with web services this information may also come from the service provider [5] or service directories, which may offer temporal information as part of their service descriptions [3]. The algorithm in [10] additionally requires explicit information about the execution duration of every node (ranging from millis to a few minutes at maximum), which we considered, compared to service response times, negligible. So we set all node durations to [0,0].

### 3.4 Transformation

How to transform a hierarchical block-structured process, which BPEL is, to a flattened graph representation is described in [4]. For the transformation we had to add the following BPEL-specific rules:

- every basic activity is represented as a node of type *act*
- the structured activities *sequence*, *if* and *flow* are represented as two nodes of corresponding type, which embrace inner nested activities: *seq-start* and *seq-end*, *xor-split* and *xor-join*, *and-split* and *and-join*
- the structured activity *pick* must be encapsulated in a sequence, which contains one node (the message receiver), followed by an xor-structure with a branch for each message type handler
- structured scope activities are interpreted as sequences

The following exceptions to above stated rules had to be considered: (a) iterative activities are represented by two nodes of type *act* connected by a precedence edge. Eventually nested elements are omitted [1]. Additionally these two nodes are connected with an asynchronous relationship edge, augmented with the specified, estimated (or calculated) execution time of the activity, specified in the temporal information file. (b) The same applies for *wait* activities, but here the duration can be extracted from the duration expression in the BPEL-definition. (c) Event, fault and compensation handlers of the original process definition are not considered, as time management focuses on the regular flow, and are therefore omitted in the graph representation.

It is basically possible to extract asynchronous, and therefore temporal, relationships (dashed arrows) between invoking and receiving activities from the BPEL process definition by interpreting the declaration parts (*partnerLinks*, etc.). However, this is outside the scope of this paper, and therefore we demand, that information about these relations must be specified by the process designer within the temporal information file. Asynchronous relationship edges and their response times can now be added to the graph in the final transformation step.

## 4 Timed Graph Calculation and State Assessment

Now we have a basic graph representation, augmented with a maximum duration and service response times (in hours) between invoking and receiving activities. Equipped with this information we can calculate the timed graph. Based on the process structure and explicit temporal information, it is possible to determine remaining durations for each node in the process graph. The remaining duration interval represents the expected minimum and maximum execution duration of the path between node  $n$  and the end of the process. The calculation algorithm utilizes the graph specified above and explicit temporal information, and yields remaining durations for each node as visualized in Figure 3. The remaining duration of the first node *proc-start* also depicts the expected overall process duration. Due to space limitations we can not explain the details of this calculation (refer to [10]). Furthermore note that remaining durations or thresholds may be calculated with any interval-based approach that is capable of dealing with above mentioned control structures, where some even allow more complex time constraints (like [11], which introduces upper and lower-bound constraints).

<sup>1</sup> As cyclic structures are problematic for time management calculations, since the actual number of iterations will not be known in advance, we applied a common solution for now: interpreting the loop as one complex activity with an estimated or calculated overall duration.

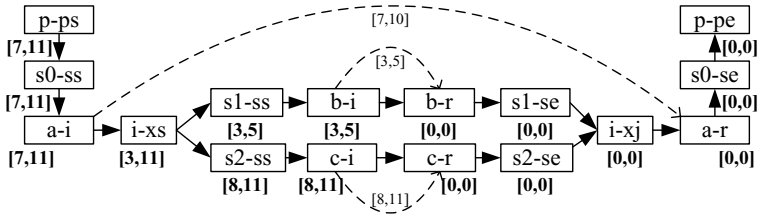


Fig. 3. Timed Graph - Generation Step 2

For run-time purposes we need thresholds, relative to the start time of the process, that shall not be exceeded in order to meet the process deadline. For this we adapted the idea of the traffic light model, described in [17]. Its lights represent the temporal states: (1) *Green* indicates, that the process can be finished within the calculated process duration. (2) *Yellow* indicates, that the process can be finished within the specified maximum duration. Future delays should be avoided, as the process already started consuming buffer time. Proactive intervention is advised. And otherwise it is (3) *Red*, which indicates, that all available buffer time has been consumed, and that missing the deadline is likely. Proactive intervention is inevitable, the process must be sped up.

For illustration purposes we applied a very simple (rather pessimistic worst-case) approach for temporal assessment: we calculate two state switching thresholds for each node/activity  $n$ , based on the upper bound of a specified maximum process duration interval  $maxduration.ub = 15$  hours, the upper bound of the calculated process duration interval  $calcduration.ub = 11$  hours, and the upper bound of the node's remaining duration interval  $n.rduration.ub$ , as follows:

- $n.greenToYellow := calcduration.ub - n.rduration.ub$
- $n.yellowToRed := maxduration.ub - n.rduration.ub$

Note, that state assessment for blocking structured activities must use the remaining durations of the corresponding end node. This calculation yields, e.g., for the node  $a-i$  the following thresholds:  $a-i.greenToYellow = 0$  hours after process start and  $a-i.yellowToRed = 4$  hours after process start. As  $a-i$  is the first basic activity in the process, it will be reached within (milli)seconds (rounded to 0 hours) after process start, therefore the temporal state will most probably never switch to yellow or red at this position. However, theoretically  $a-i$  could consume up to four hours before the state switches to red - this time is also called *buffer time*. For the receiver activity  $b-r$ , we determine the following values:  $b-r.greenToYellow = 11$  and  $b-r.yellowToRed = 15$ . If, for example,  $b-r$  did not receive its message until 11 (hours after process start) then the state switches to yellow and we could start an intervention, e.g., interrupt the waiting activity  $b-r$  and invoke an alternative fast (more "expensive") service which returns a message immediately: the process is in time, but the costs increased. You will notice, that the threshold-values of  $b-r$  are equal to the specified and calculated process duration (and equal to the thresholds of  $c-r$  and  $a-r$ ). This means that each of these activities is allowed to consume the whole buffer time of the process, leaving no buffer for subsequent activities. For descriptions of fairer buffer distribution techniques refer to [7].



## 5 Interventions

Proactive time management needs information about how to intervene when the temporal state changes. We support the following intervention strategies, which may be applied on any BPEL-activity (basic or structured).

**Optional Execution.** Skipping optional activities can be used on any element, which is not absolutely necessary for the successful completion of the process. Therefore the element, or activities nested within this element, should not have communication relationships to another element in the process (e.g. skipping an invoke activity may block the process at the matching receive activity).

**Parallelization of Sequence.** Parallelization of sequences forces the parallel execution of elements of a sequence. Note, that elements nested in the sequence (or their subelements) may have communication relationships between each other. In the worst case, such an execution will again be sequential.

**Alternative Path.** A late process can be sped up by executing a faster alternative (basic or structured) activity, instead of the original one. Again, eventually existing communication relationships between elements must be considered: e.g., an alternative for an invoke-activity which calls another service may block the process at the matching receive-activity.

**Dynamic Service Selection.** This strategy is a variant of Alternative Path that offers multiple alternatives. In case the process is late, the fastest of several variants must be selected and executed. We assume, that the list of matching candidates has been preselected. Alternatively one could also apply a QoS-based adaptive service-retrieval technique, which automatically finds compatible candidates [23].

**Early Termination.** Early termination of a late process, depicted by *terminate*, aims at the avoidance of costs resulting from further process execution and exception-handling actions at the end. Although we do not consider the cost factor in this paper (cp. [6]), we offer this policy as a last resort, only to be used in extreme cases, as it does not consider side-effects on integrated processes or services.

The specification of interventions binds activities of the process, depicted by their name, to a certain intervention-behavior, which shall be invoked instead of the activity, if the process is in the given temporal-state. We defined an XML-structure for intervention information, specified by the following DTD.

```
<!ELEMENT interventions (intervention)+ >
<!ELEMENT intervention (intervene)+ >
<!ATTLIST intervention activity CDATA #REQUIRED >
<!ELEMENT intervene ((terminate|optional|parallelize|alternative|dynamic), bpe1?) >
<!ATTLIST intervene when (yellow|red) #REQUIRED >
<!ELEMENT terminate EMPTY >
<!ELEMENT skip EMPTY >
<!ELEMENT parallelize EMPTY >
<!ELEMENT alternative (bpe1) >
<!ELEMENT dynamic (variant)+ >
<!ATTLIST dynamic objective (green|yellow|red) #REQUIRED >
<!ELEMENT variant (bpe1) >
<!ATTLIST variant duration CDATA >
<!ELEMENT bpe1 (declarations,actions) >
<!ELEMENT declarations (#PCDATA) >
<!ELEMENT activity (#PCDATA) >
```

A set of *interventions* for a specific process may contain several *intervention* elements. Each refers to a basic or structured *activity*-name in the BPEL-process and contains *intervene* elements, that define which intervention strategy to apply, *when* a certain temporal state is assessed. An Intervene-element must contain one element of type *terminate*, *skip*, *parallelize*, *alternative*, or *dynamic*, which specifies the strategy to apply. Furthermore, an intervene-element may contain an optional *bpel* element, which defines a BPEL-activity (basic or structured) and necessary BPEL-declarations, which we do not further specify here (variables, partnerLinks, etc.). This BPEL code will be executed before the intervention itself takes place; it may for instance be used to notify the process-owner about temporal state changes. The following example defines interventions for two activities of a (fictitious) process.

```
<interventions>
  <intervention activity="a_sequence" >
    <intervene when="red"> <optional/> </intervene>
    <intervene when="yellow"> <parallelize/> </intervene>
  </intervention>
  <intervention element="an_activity">
    <intervene when="yellow"> <optional/> </intervene>
    <intervene when="red" >
      <terminate />
    <bpel>
      <declarations> ... BPEL declarations ... </declarations>
      <activity> ... BPEL activity (e.g. notify owner)... </activity>
    </bpel>
  </intervene>
</intervention>
</interventions>
```

Intervene-elements of type *alternative* must contain *bpel* declarations and code of the alternative. For interventions of type *dynamic* we must specify multiple execution *variants*, augmented with information about the (expected) duration for each alternative. The attribute *objective* defines the desired temporal state after the execution of a variant. The *duration* of a variant (may be a structured activity) can be calculated with the calculation algorithm explained above. The following intervention specification refers to activities within our example process. The sequence *s1* shall be skipped if the process enters state *red*, and if it enters state *yellow* alternative code shall be executed (e.g. invoking a fast service and receiving its message). If the process waits too long at the receiving activity *a-r* it shall either select a faster variant (*yellow*) or even terminate (*red*). Note, that both activities are considered 'blocking', as they wait for an incoming message.

```
<interventions>
  <intervention element="s1">
    <intervene when="red"> <optional/> </intervene>
    <intervene when="yellow">
      <alternative>
        <bpel> declarations and activity for alternative </bpel>
      </alternative>
    </intervene>
  </intervention>
  <intervention element="a-r">
    <intervene when="red"> <terminate/> </intervene>
    <intervene when="yellow">
      <dynamic objective="green">
        <variant duration="[6,9]">
          <bpel> declarations and activity for alternative1 </bpel>
        </variant>
      </dynamic>
    </intervene>
  </intervention>
</interventions>
```

```

        </variant>
        <variant duration="[3,5]">
            <bpel> declarations and activity for alternative3 </bpel>
        </variant>
        <variant duration="[4,7]" >
            <bpel> declarations and activity for alternative2 </bpel>
        </variant>
    </dynamic>
</intervene>
</intervention>
</interventions>

```

## 6 Generation of a Time-Aware Process Definition

A time-aware process definition is an extension of the original process definition with state assessment and intervention mechanisms for specified process-parts, to be executed in case given thresholds are violated. The generation is based on the original process definition, the timed graph, and intervention information, and consists of the following basic steps:

```

[works on copy of original process definition]
add process-level extensions
for each activity x in bottom-up order (deepest nestings first)
    if exists intervention for x
        if x is non-blocking
            add activity-level extensions for non-blocking activity x
        elseif x is blocking
            add activity-level extensions for blocking activity x
        end-if
    end-if
end-for

```

### 6.1 Process-Level Extension

First the original process definition must be extended on the top-level, as visualized in the BPMN-diagram [2] on the left-hand side of Figure 4 (elements of the original process are displayed grey-shaded). We decided to use BPMN as graphical representation instead of BPEL-code due to space limitations.

1. Nest the top-level activity of the process within a new sequence-activity *top\_seq*
2. Insert assignment for the current time *t*, which is the start time of the process.
3. Initialize the temporal *state* by assigning it the value 'green'.

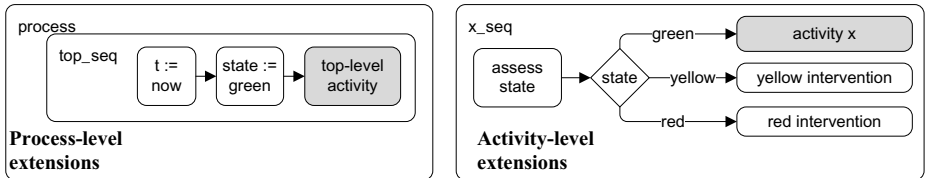


Fig. 4. Process-level Extensions and Extensions for non-blocking Activities

We used XQuery and XPath in our BPEL-prototype for accessing, comparing and manipulating variables, as they offer a rich function-base for diverse purposes, along with datatypes for the structured representation of dates, times and durations.

## 6.2 Generation of Intervention Logic

The intervention logic for an activity consists of a state assessment mechanism to determine the current temporal state, conditional structures to select the corresponding intervention, and timed triggers for blocking elements.

**State Assessment Mechanism.** An important part of intervention logic is the assessment of the current temporal state. The following shows a simplified pseudo-code representation of the necessary state assessment extensions for an activity  $x$  and its thresholds:

```
rel_time := currentTime() - t;
if (rel_time <= x.greenToYellow) then state := green
  elseif (rel_time <= x.yellowToRed) then state := yellow
  else state := red
```

State assessment is based on a comparison of the relative time (duration since start of the process) and the corresponding node-dependent threshold value. For non-blocking activities it takes place before the invocation, and for blocking activities during their execution.

**Basic Intervention Extensions for Non-blocking Activities.** The BPMN-diagram on the right-hand side of Figure 4 shows the extensions for a non-blocking activity  $x$ .

1. Replace activity  $x$  with a new if-activity  $x\_if$  (omitted in the diagram).
2. Add three branches and conditions for states green (if), yellow and red (elseif).
3. Insert activity  $x$  into the if-branch.
4. Generate intervention handling code for the elseif-branches (see details below).
5. Nest  $x\_if$  within a new sequence-activity  $a\_seq$ .
6. Insert state assessment elements before  $x\_if$  into  $a\_seq$ .

**Intervention Extensions for Blocking Activities.** For blocking activities such a passive intervention mechanism is not sufficient. The prediction component must additionally check threshold-violations during the execution of this element. Therefore it is necessary to add time-triggered logic as visualized in Figure 5. We used several mechanisms: timed triggers (circle with clock) which invoke corresponding event-handlers, throwing of faults (fat circle with flash-symbol), and catching of faults within a fault-handler (double circle with flash-symbol). Within a BPEL-scope we use fault-handlers, which catch named faults (exceptions) that are thrown within this scope. Furthermore, it is possible to define time triggered event-handlers based on the onAlarm-element, which periodically executes specified code (basically a concurrent sub-process). The diagram is to be interpreted as follows: the control flow enters the scope and starts the (blocking) activity. An onAlarm event-handler periodically calls state assessment, which checks if the temporal state has changed. If this is the case, it will immediately

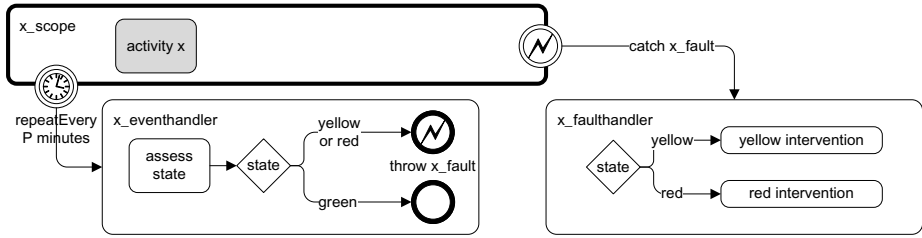


Fig. 5. Activity-level Extensions for blocking Activities

throw  $x\_fault$ , which is caught by  $x\_faulthandler$ . If the state is still green the event handler will return control to the regular control flow (the execution of  $x$ ). The generation of extensions for blocking activities consists of the following steps:

1. Nest activity  $x$  within a new scope  $x\_scope$ .
2. Add  $x\_eventhandler$  to the scope, containing an onAlarm-element with a specified repeatEvery-period, including state assessment logic including a throw-element.
3. Add  $x\_faulthandler$  to the scope, which contains state-dependent intervention handling code.

**Generating Intervention Handler Code.** Intervention handler code is executed when the temporal state changes to yellow or red, and must be integrated in the corresponding if-branches. Intervention code for an activity  $x$  and a certain temporal state is determined by the specified *intervene*-element within the intervention information (see Section 5) and basically generated as follows.

1. Insert a sequence  $x\_int\_seq$  within the corresponding if-branch.
2. Add type-dependent intervention code to  $x\_int\_seq$  (for details see below).
3. If the optional *bpel*-element exists within the intervene element:
  - (a) add the *bpel*-activity before the intervention code that was generated in step 2
  - (b) add the related *bpel*-declarations (variables, partnerLinks, etc.) on process level

Step 2 generates type-dependent intervention code: for an intervention of type *optional* we add the BPEL-activity *empty* and for *terminate* we simply add the BPEL-activity *exit*. Adding BPEL-code for an *alternative* path is equal to steps 3.a) and 3.b). To *parallelize* a sequence we generate a flow element in the corresponding if-branch of the intervention logic, which contains a duplicate of every activity within the sequence-activity (including already generated intervention logic of nested activities).

The last intervention mechanism, *dynamic* service replacement, has to select one out of several replacement alternatives. Selection is based on the current delay of the process, the position and duration of replacement variants, and the objective (desired goal state). Code generation is best explained by an example. In Section 5 we defined a yellow-intervention of type *dynamic* for activity a-r, with three replacement variants ordered by maximum duration: one with a duration of [6,9], one with [4,7] and one with [3,5]. The objective is green, which means that the execution of the selected alternative

should be finished until the green end-threshold of a-r. The calculation of green and yellow thresholds for activity a-r yielded:  $c_{greenToYellow}=11$  and  $c_{yellowToRed}=15$  (hours after the process start). With this information we generate the following code:

```
if (state = yellow)
  rel_time := currentTime() - t;
  timeframe := a-r.greenToYellow - rel_time;
  if (timeframe >= 9) bpel-code of variant
    elseif (timeframe >= 7) bpel-code of variant
    else bpel-code of fastest variant
```

Again we applied a worst-case approach: first we calculate the relative time (duration since process start), followed by the calculation of the time frame. The time frame is the difference between the threshold of the goal state (greenToYellow) and the relative time. The duration of the selected variant must fit into this time frame. Therefore we compare the time frame with the upper bound values (worst case) of the duration intervals specified for each alternative (in decreasing order) and add corresponding BPEL-code. The fastest variant will be selected, even if it does not fit in the time frame. This approach assumes that the faster a service the more expensive it will be, and shall therefore only be selected if absolutely necessary. For our prototype we implemented an additional version, which allows to specify variants ordered by preference – a preferred service will be selected when it fits into the time frame, even if a slower service exists, that also fits into the time frame.

Furthermore, we designed and implemented an improved version for alternative and dynamic interventions on blocking activities, which exploits the following finding: if the (blocking) activity is almost finished, it is not a wise decision to interrupt it and execute one of the variants. Therefore we added a special treatment on state-assessment-level for these types, which checks (on state change) if the remaining execution duration of the activity is less than the duration of the variant that fits into the time frame. If this is the case no fault will be thrown, and the original activity is allowed to finish.

## 7 Prototypical Implementation and Complexity Considerations

We implemented a proof-of-concept Java-based transformation prototype and tested several processes with the open source engine ActiveBPEL. The advantage of this approach is, that it adds pro-active time-awareness to the process, which results in less deadline violation. The process designer is not addressed with complicated calculations and programming of tedious intervention logic. Furthermore, if the effect of current intervention strategies is insufficient, then temporal information, intervention strategies or temporal assessment can be changed easily, and used to generate new time-aware process definition. On the downside we have to state, that the generated process will contain considerably more elements than the original process definition. The number of additional elements varies heavily, depending on various parameters, the nesting-depth, the number of specified interventions, and the complexity of intervention logic. Still, it can be predicted by using the tables in Figure 6. For specific interventions both tables must be combined, e.g., when defining an intervention for non-blocking sequence-activity (with  $n = 3$  nested basic activities), where the yellow-intervention is parallelize and the

Extension Types: Number of additional Elements

	var decl.	var assign	seq	if		scope	fault handler	throw	catch	event handler	on Alarm	Sum
				#	br*							
<b>process**</b>	3***	2	1	0	0	0	0	0	0	0	0	<b>6</b>
<b>non-blocking</b>	0	3	1	1	3	0	0	0	0	0	0	<b>8</b>
<b>blocking</b>	0	3	1	3	7	1	1	1	1	1	1	<b>20</b>

\*number of branches incl. conditions      \*\* always included      \*\*\* includes rel\_time for state assessment

Intervention Types: Number of additional Elements (w/o optional BPEL code)

	act & decl.	flow	seq	if		var assign	Sum
				#	br		
<b>skip</b>	1	0	0	0	0	0	<b>1</b>
<b>terminate</b>	1	0	0	0	0	0	<b>1</b>
<b>parallelize</b>	n	1	0	0	0	0	<b>n + 1</b>
<b>alternative</b>	b + d	0	0	0	0	0	<b>b + d</b>
<b>dynamic</b>	(v*b) + (v*d)	0	1	1	v	2	<b>(v*b) + (v*d) + 4</b>

n ... #activities nested in activity for which intervention has been declared  
 v ... #variants (activities) for dynamic replacement  
 b ... avg. #activities nested in activities, which describe one variant  
 d ... avg. #declarative elements (partnerLinks, etc.) for one variant

Fig. 6. Tables for Prediction of Number of Additional Elements

red-intervention is skip, we calculate the number of additional elements as: sum process + sum non-blocking + sum skip + sum parallelize = 6 + 8 + 1 + 4 = 19 additional elements. In nearly all cases the complexity will be linear, with one exception: nested parallel structures, where increase is exponential in the number of parallelizations in a nesting path, as all nested elements must be duplicated on each level with parallelization during the bottom up generation of intervention logic. Therefore, we propose to specify interventions only for selected mission-critical parts and for parts which have the potential to significantly speed up the process. A related problem is, that a user who monitors the progress of the process will see a rather complicated transformed process.

## 8 Conclusions

The prediction and proactive avoidance of deadline violations decreases costs of processes and increases their quality of service. Existing approaches describe how to model and calculate temporal information for these purposes, but do not show how to apply corresponding interventions on running processes. Therefore we enhanced the original process definition with additional interval-based temporal and intervention information, and showed how to transform it into a time-aware process definition, which pro-actively avoids looming deadlines and that can be executed on any engine that supports BPEL. To achieve this we utilized inherent control-flow features of the process definition language to integrate time-triggered predictive and proactive intervention mechanisms, with a focus on blocking activities that wait for messages of delayed external services. Current and future research comprises handling of non-blocked structures (flows with links), exception handlers, and how to compensate already finished activities.

## References

1. Cardoso, J., Sheth, A., Miller, J.: Workflow Quality of Service. In: Proc. of the Int. Conf. on Integration and Modeling Technology (IEIMT/IEMC). Kluwer Publishers, Dordrecht (2002)
2. OMG: Business Process Modelling Notation (BPMN) 1.1. OMG Specification (2008)

3. W3C: OWL-S: Semantic Markup for Web Services. W3C Member Submission (2004)
4. Eder, J., Gruber, W.: A Meta Model for Structured Workflows Supporting Workflow Transformations. In: Manolopoulos, Y., Pokorný, J., Sellis, T.K. (eds.) ADBIS 2006. LNCS, vol. 4152. Springer, Heidelberg (2006)
5. Gillmann, M., Weikum, G., Wonner, W.: Workflow Management with Service Quality Guarantees. In: Proc. of ACM SIGMOD Int. Conf. on Management of Data. ACM Press, New York (2002)
6. Panagos, E., Rabinovich, M.: Predictive Workflow Management. In: Proc. of the 3rd Int. Workshop on Next Generation Information Technologies and Systems, Neve Ilan, Israel (1997)
7. Kao, B., Garcia-Molina, H.: Deadline Assignment in a Distributed Soft Real-Time System. IEEE Transactions on Par. Dist. Systems 8(12) (1997)
8. van der Aalst, W.M.P., Rosemann, M., Dumas, M.: Deadline-based Escalation in Process-Aware Information Systems. BPM Center Report, BPM-05-05, BPMcenter.org (2005)
9. Eder, J., Pichler, H., Vielgut, S.: An Architecture for Proactive Timed Web Service Compositions. In: Eder, J., Dustdar, S. (eds.) BPM Workshops 2006. LNCS, vol. 4103, pp. 323–335. Springer, Heidelberg (2006)
10. Eder, J., Pichler, H., Vielgut, S.: Avoidance of Deadline-violations for Inter-org. Business Processes. In: Proc. of the 7th Int. Baltic Conf. on DBs and Inf. Systems. IEEE Press, Los Alamitos (2006)
11. Eder, J., Panagos, E., Rabinovich, M.: Time Constraints in Workflow Systems. In: Jarke, M., Oberweis, A. (eds.) CAiSE 1999. LNCS, vol. 1626, p. 286. Springer, Heidelberg (1999)
12. Haimowitz, I.J., et al.: Temporal Reasoning for Automated Workflow in Health Care Enterprises. In: Adam, N.R., Yesha, Y. (eds.) Electronic Commerce 1994. LNCS, vol. 1028. Springer, Heidelberg (1996)
13. Marjanovic, O., Orłowska, M.: On Modeling and Verification of Temporal Constraints in Production Workflows. Knowledge and Information Systems 1(2) (1999)
14. Marjanovic, O., Orłowska, M.: Workflow Temporal Manager. In: Proc. of the Australian Workshop on Intelligent Decision Support and Knowledge Management, Sydney, Australia (1998)
15. Marjanovic, O., Orłowska, M.: Dynamic Verification of Temporal Constraints in Production Workflows. In: Proc. of the Australasian Database Conf. IEEE Computer Society, Los Alamitos (2000)
16. Baggio, G., et al.: Applying Scheduling Techniques to Minimize the Number of Late Jobs in Workflow Systems. In: Proc. of ACM 2004 Symp. on Applied Computing. ACM Press, New York (2004)
17. Eder, J., Panagos, E.: Managing Time in Workflow Systems. In: Workflow Handbook 2001, Future Strategies Inc. (2001) ISBN 0-970-35090-2
18. Newcomer, E.: Understanding Web Services. Addison-Wesley, Reading (2002)
19. Bettini, C., et al.: Free Schedules for Free Agents in Workflow Systems. In: Proc. of 7th Int. Workshop on Temporal Representation and Reasoning. IEEE Computer Society Press, Los Alamitos (2000)
20. Benatallah, B., Casati, F., Ponge, J., Toumani, F.: On temporal abstractions of web service protocols. In: CAiSE 2005 Forum Short Paper Proceedings, CEUR-WS.org (2005)
21. Kazhamiakin, R., et al.: Representation, verification, and computation of timed properties in web service compositions. In: Proc. of Int. Conf. on Web Services. IEEE Comp. Society, Los Alamitos (2006)
22. Pozewaunig, H., et al.: ePERT – Extending PERT for Workflow Management Systems. In: Proc. of Symp. on Adv. in Databases and Information Systems, Nevsky Dialect (1997)
23. Ardagna, D., et al.: PAWS: A Framework for Executing Adaptive Web-Service Processes. IEEE Software 24(6) (2007)



# Asynchronous Timed Web Service-Aware Choreography Analysis

Nawal Guermouche and Claude Godart

LORIA-INRIA-UMR 7503

F-54506 Vandoeuvre-les-Nancy, France

{Nawal.Guermouche,Claude.Godart}@loria.fr

**Abstract.** Web services are the main pillar of the Service Oriented Computing (SOC) paradigm which enables applications integration within and across business organizations. One of the important features of the Web services is the *choreography* aspect which allows to capture collaborative processes involving multiple services. In this context, one of the important investigations is the *choreography compatibility analysis*. We mean by the choreography compatibility the capability of a set of Web services of actually interacting by exchanging messages in a proper manner. Whether a set of services are compatible depends not only on their sequences of messages but also on *quantitative properties* such as *timed properties*. In this paper, we investigate an approach that deals with checking the *timed compatibility of a choreography* in which the Web services support *asynchronous timed communications*.

**Keywords:** Web service, Timed properties, Asynchronous timed Compatibility analysis.

## 1 Introduction

The evolution of computer science technologies gives life to many paradigms such as the Service Oriented Computing (SOC) paradigm. In this latter, Web services are the main pillar. Based on standard interfaces, Web services facilitate application-to-application interaction. This advantageous property gives rise to several important concepts such as the choreography concept. This feature offers the possibility to capture collaborative processes involving multiple services where the interactions between these services are seen from a global perspective. In this context, one of the important elements is the compatibility analysis. By compatibility we mean the capability of a set of services of actually fulfilling successful interactions by exchanging messages.

In the last few years, few works have investigated the compatibility problem of a client and a provider service [4,2,13,9,7]. In all these works, the authors deal with services that support *synchronous communications*. In that case, to characterize the compatibility class of two services, the authors check if each input (resp. output) message of a service corresponds to an output (resp. input)

message of the other service in the same order (i.e., the services are synchronized over messages). However, the nature of distributed systems and particularly the Web services could be asynchronous, hence the problem of the applicability of these approaches in real application scenarios is still open. To overcome such limitations, in this paper, we tackle the problem of analyzing the compatibility of a choreography in which the Web services support *asynchronous* communications. In an asynchronous communication, when a message is sent, it is inserted to a message queue, and the receiver can consume it later from the queue.

It is commonly agreed that in general the interaction of Web services and in particular the compatibility of Web services depend not only on the supported sequences of messages but also on crucial *quantitative properties* such as *timed properties* [2,11,10,13,9,7,8]. We mean by *timed properties* the necessary delays to exchange messages (e.g., in an e-government system to manage handicapped pension requests, a prefecture must send its final decision to grant an handicapped pension to a requester after 168 hours and within 336 hours). There are some works that tried to consider timed properties when analyzing the compatibility of two synchronous services [2,13,9,7]. However, dealing only with synchronous services decreases considerably the feasibility and the applicability of these approaches.

In this paper, we propose a framework for analyzing the choreography compatibility. This framework supports asynchronous communicating services. In our framework we take into account data flow that can be involved when exchanging messages. Furthermore, we consider timed properties that specify the necessary delays to exchange messages. By studying the possible impacts of timed properties on a choreography, we remark that when the Web services are interacting together, *implicit timed dependencies* could be built between the different timed properties of the different services. Such dependencies could give rise to *implicit timed conflicts*. To discover such timed conflicts, we first study the possibility to apply the proposed compatibility approaches of synchronous services [2,13,9,7], and we have remarked that the existing approaches are inadequate to discover all the eventual timed conflicts since the authors rely on synchronizing the services over messages. In order to catch all the possible timed conflicts, in this paper we rely on the *clock ordering process* we have proposed in some earlier work on Web service composition [8]. The clock ordering process aims at making explicit the eventual implicit timed conflicts when services are interacting together.

To summarize, in this paper we make the following contributions: (1) we propose an asynchronous model of Web services that takes into account messages, data types and timed requirements. (2) unlike existing compatibility frameworks, we propose primitives for analyzing and characterizing the compatibility class of a choreography in which the services support *asynchronous timed communications*.

The reminder of the paper is organized as follows. Section 2 presents the e-government case study that we use to show the related issues of the proposed approach. In Section 3 we present how we model the timed behavior of

Web services. For better understanding, in section 4, we discuss informally and intuitively the timed compatibility problem of a choreography. Section 5 presents the formal choreography compatibility investigation we propose. An illustrative example using the e-government scenario is given in Section 6. In Section 7, we discuss related work. Finally section 8 concludes.

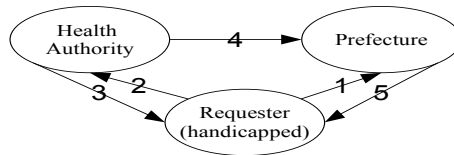
## 2 Case Study: e-Government Application

Let us present a part of an e-government application that we use at the end to illustrate our approach. The goal of the e-government application we consider is to manage handicapped pension request. Such a request involves three Web services: (1) *requester* service (*RS*), (2) *health authority* (*HS*) service, and (3) *prefecture* service (*PS*). The high level choreography model of the process is depicted on Fig. 1. To grant an handicapped pension to a requester, the process can be briefly summarized as follows. (1) via a requester service, a citizen deposits a file in the prefecture, (2) the citizen requests a medical file from the health authority service. (3) The health authority negotiates a date for an appointment to examine the citizen. (4) after the examination, the health authority service sends a medical report to the prefecture. (5) after studying the received file and the medical report, the prefecture sends the notification of the final decision to the citizen.

The interaction between these partners is constrained by timed properties. Below, we give some timed requirements.

- Once the health authority service proposes meeting dates to the citizen, the health authority service must receive the filled form within 24 hours.
- The prefecture requires at least 168 hours and at most 336 hours since receiving the file from the requester to notify the citizen with the final decision.
- Via the requester service, once the citizen obtains the medical form, he must send the filled form within 36 hours.

The Web services we consider could support asynchronous communications. The first issue we deal with is how to analyze the compatibility of a choreography in which the Web services are asynchronous? Moreover, the behavior of the Web services might be constrained by timed requirements. In order to manage



**Fig. 1.** Global view of the e-government application

the global interaction between the Web services (i.e., to ensure that the choreography is deadlock free), we need primitives that consider timed properties when analyzing the compatibility of Web services. Thus, the second issue we handle is how to consider timed properties when analyzing the compatibility of asynchronous services in a choreography?

### 3 Modeling Timed Behavior of Web Services

One of the important ingredient in a compatibility framework is the *timed conversational protocol* of Web services. In our framework, the timed conversational protocol specifies the sequences of messages a service supports, the involved data flow and the associated timed properties to exchange messages. We adopt a finite state machine based formalism to model the timed behavior of Web services (i.e., the timed conversational protocol). Intuitively, the states represent the different phases a service may go through during its interaction. Transitions enable sending or receiving a message. An output message is denoted by  $!m$ , whilst an input one is denoted by  $?m$ . A message involving a list of data types is denoted by  $m(d_1, \dots, d_n)$ , or  $m(\bar{d})$  for short. To capture the timed properties when modeling Web services, we propose to use the standard timed automata clocks [11]. The automata are equipped with a set of clocks. The values of these clocks increase with the passing of time. Transitions are labeled by timed constraints, called *guards*, and resets of clocks. The former represent simple conditions over clocks, and the latter are used to reset values of certain clocks to zero. The guards specify that a transition can be fired if the corresponding guards are satisfiable.

Let  $X$  be a set of clocks. The set of *constraints* over  $X$ , denoted  $\Psi(X)$ , is defined as follows:

$\text{true} \mid x \bowtie c \mid \psi_1 \wedge \psi_2$ , where  $\bowtie \in \{\leq, <, =, \neq, >, \geq\}$ ,  $x \in X$ ,  $\psi_1, \psi_2 \in \Psi(X)$ , and  $c$  is a constant.

**Definition 1.** (*Timed conversational protocol*)

A *timed conversational protocol* of a Web service  $Q$  is a tuple  $(S, s_0, F, M, X, T)$  such that:

$S$  is a set of states,  $s_0$  is the initial state ( $s_0 \in S$ ),  $F$  is the set of final states ( $F \subseteq S$ ),  $M$  is a set of messages,  $X$  is the set of clocks,  $T$  is a set of transitions such that  $T \subseteq S \times M \times \Psi(X) \times 2^X \times S$ , with an exchanged message that involves data types ( $?m(\bar{d})$ : input message,  $!m(\bar{d})$ : output message), a guard over clocks, and the clocks to be reset.

**A transition**  $(s, a, \psi, Y, s')$  is denoted by  $s \xrightarrow{a}_{\psi, Y} s'$ .

**A trace** is a sequence of transitions leading to a final state, denoted as follows:  $s_0 \xrightarrow{\alpha_0}_{\psi_0, Y_0} s_1 \xrightarrow{\alpha_1}_{\psi_1, Y_1} \dots \xrightarrow{\alpha_{n-1}}_{\psi_{n-1}, Y_{n-1}} s_n$  where  $s_n$  is a final state.

The semantic of the former is defined using a transition relation over configurations made of a state and a clock valuation. The clock valuation is a mapping  $u : X \rightarrow \mathbb{T}$  from a set of clocks to the domain of timed values. The mapping  $u_0$

denotes the (initial) clock valuation, such that  $\forall x \in X, u_0(x) = 0$ . Initially, the queue of the services are empty.

A service remains in the same state  $s$  without triggering a transition, when the time increments, if there is no transition  $(s, \alpha, \psi_X, Y, s')$  such that the timed constraints  $\psi_X$  are satisfied, where  $\psi_X \subseteq \Psi(X)$  and  $\alpha$  is either an output message  $!m(\bar{d})$  or an input message  $?m(\bar{d})$  which is available in the queue. In an asynchronous communication, when a message is sent, it is inserted to a message queue, and the receiver consumes (i.e. receives) the message while it is available in the queue.

**Definition 2.** (Semantic of timed conversational protocol)

Let  $P = (S, s_0, F, M, X, T)$  be a conversational protocol. The semantic is defined as a labeled transition  $(\Gamma, \gamma_0, \rightarrow)$ , where  $\Gamma \subseteq S \times V_T$  is the set of configurations, such that  $V_T$  is a set of timed valuations,  $\gamma_0 = (s_0, u_0)$  is the initial configuration, and  $\rightarrow$  is defined as follows:

- Elapse of time:  $(s, u) \xrightarrow{tick} (s, u + \delta)$
- Location switch:  $(s, u) \xrightarrow{\alpha} (s', u')$ , if  $\exists t = (s, \alpha, \psi_X, Y, s')$  such that  $u \wedge \psi_X$  are satisfiable and  $\forall y \in Y, u'(y) = 0, \forall x \in X \setminus Y, u'(x) = u(x)$ , where  $Y \subseteq X$ , and
  - If  $\alpha = !m(\bar{d})$ ,  $Queue := Queue + m(\bar{d})$
  - If  $\alpha = ?m(\bar{d})$ , and  $m(\bar{d}) \in Queue$ ,  $Queue := Queue - m(\bar{d})$

The timed conversational protocols of the three services introduced in Section 2 are depicted on Fig. 2. Next, we will present the intuition behind the choreography compatibility problem.

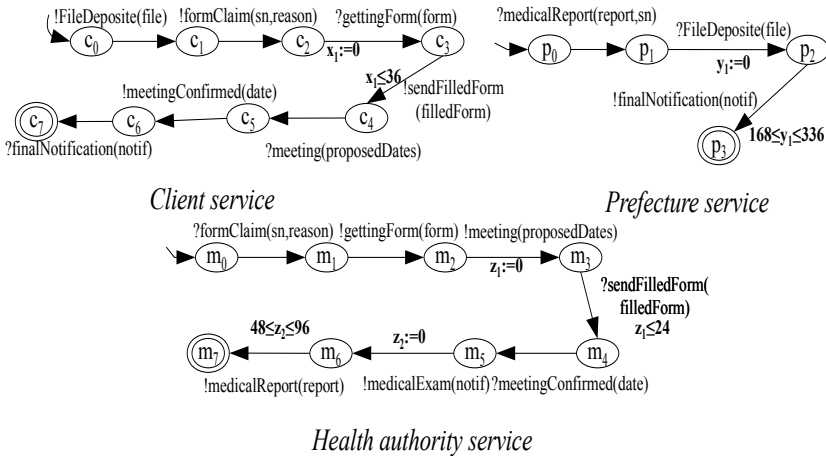


Fig. 2. Web services

## 4 Timed Compatibility Problem

In this section, by using examples, we discuss informally and intuitively the timed choreography compatibility problem and the related issues. Then, in the next section, we present the formal investigation we propose.

*Example 1.* Let us first consider the two *untimed* conversational protocols of the two services  $Q$  and  $Q'$  depicted on Fig. 3. In spite that both services cannot produce and consume their messages in the same order, the two asynchronous services are *fully compatible*. The service  $Q$  starts by sending the message  $m_0(d_0, d_1)$  which becomes available in the queue of  $Q'$ . On the other side,  $Q'$  sends the message  $m_2(d_3)$ . After that,  $Q'$  consumes the message  $m_0(d_0, d_1)$  then it sends the message  $m_1(d_2)$ . Once the message  $m_1(d_2)$  is sent, it is added to the queue of  $Q$ . Therefore,  $Q$  can consume the message  $m_1(d_2)$  and then the message  $m_2(d_3)$ . By using the existing work, these two services are considered as incompatible although they can succeed an execution. In fact, the proposed frameworks (e.g., see [4,21,3,9,7]) deal only with synchronous communicating services.

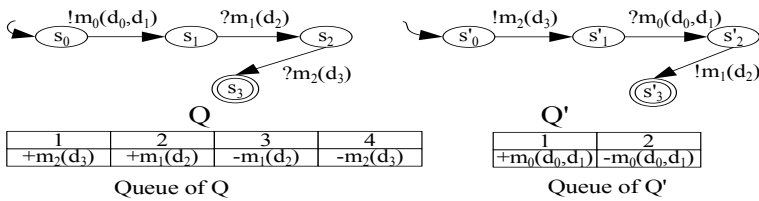


Fig. 3. Untimed asynchronous Web services

Augmenting the conversational protocol of asynchronous services by timed properties lays important challenges. Particularly, *the clocks used to define timed properties are local and mutually independent*. At the same time, in our work, we do not assume that the timed properties are synchronized over messages. Consequently, when services interact together, implicit timed conflicts can arise. To illustrate this issue, in the following we present an illustrative example.

*Example 2.* Let us consider the two timed conversational protocols of the two services  $Q$  and  $Q'$  depicted on Fig. 4. The service  $Q$  starts by sending the message  $m_0(d_0, d_1)$ . So this latter becomes available in the queue of  $Q'$ . On the other hand,  $Q'$  can send the message  $m_2(d_3)$  that can be stored in the queue of  $Q$ . The service  $Q$  remains blocked, since the message  $m_1(d_2)$  is not yet available. But  $Q'$  can consume the message  $m_0(d_0, d_1)$  which has been already sent by  $Q$ . Once consumed,  $Q'$  sends the message  $m_1(d_2)$  after 20 and within 40 units of time from consuming the message  $m_0(d_0, d_1)$  (i.e.,  $20 \leq x \leq 40$ ). Consequently, the message  $m_1(d_2)$  becomes available in the queue of  $Q$  after 20 units of time from consuming the message  $m_0(d_0, d_1)$ . In that case,  $Q$  will be able to consume

the message  $m_1(d_2)$  after 20 units of time. Finally,  $Q$  must consume the message  $m_2(d_3)$  within 10 units of time. However, this message can be consumed only after consuming the message  $m_1(d_2)$ , i.e., after 20 units of time. In fact, the message  $m_1(d_2)$  can be sent (becomes available) by  $Q'$  after 20 units of time. So, the message  $m_2(d_2)$  must be consumed within 10 units of time and at the same time it is possible to consume it after 20 units of time which represents a timed conflict.

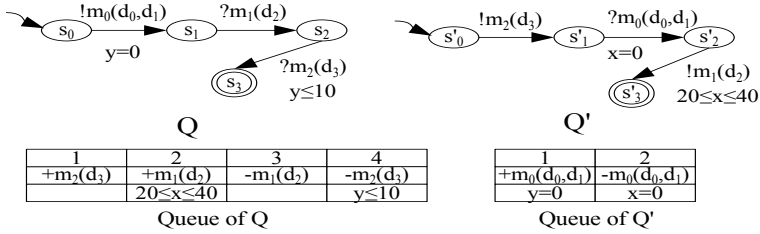


Fig. 4. Incompatible timed asynchronous services

In order to catch the eventual timed conflicts in a choreography, we propose the formal approach described in the following section.

## 5 Formal Compatibility Analysis

In the previous section, we have shown the need of formal primitives of analyzing the timed choreography compatibility of asynchronous services. In general a compatibility framework should be able to characterize the compatibility class of Web services. But in addition, in case of compatibility, it is quite important to characterize the deadlock free interaction schema of a choreography. When the different services are interacting together, timed dependencies could be created between their different timed properties. Therefore, we need mechanisms that allow to catch the eventual implicit timed conflicts. In the following sections we present respectively how we compute the interaction schema of a set of Web services and how to discover the eventual timed conflicts when computing this schema.

### 5.1 Building the Timed Choreography Interaction Schema

By having the set of conversational protocols of the involved services, our aim is to build a global timed conversational protocol that specifies an executable Timed Choreography Interaction Schema (TCIS). In order to build this TCIS, we introduce the concept of configuration that represents the states of the TCIS at a given time. A configuration defines the evolution of the services states when they are interacting together. In the initial configuration, all the services are

in their initial states. Given a source configuration, the TCIS reaches a new configuration when there exists one service that changes its state by exchanging a message so that no timed conflicts arise. The process of discovering the eventual implicit timed conflicts is presented in Section 5.2.

**Definition 3.** (*Timed Choreography Interaction Schema (TCIS)*)

Let  $Q_i(S_i, s_{0_i}, F_i, M_i, X_i, T_i)$  to be a set of Web services for  $i = \{1, \dots, n\}$ . The Timed Choreography Interaction Schema TCIS of  $Q_i$  is defined as a tuple  $(S, s_0, F, M, X, T)$  such that

$S = S_1 \times \dots \times S_n$ ,  $s_0 = s_{0_1} \times \dots \times s_{0_n}$ ,  $F = F_0 \times \dots \times F_n$ ,  $M = M_0 \cup \dots \cup M_n$ ,  $X = X_0 \cup \dots \cup X_n$ ,  $T \subseteq S \times M \times \Psi(X) \times 2^X \times S$  is defined as follows:

- $(s_1 \dots s_i \dots s_n, m(\bar{d}), \psi'_X, Y, s_1 \dots s'_i \dots s_n) \in T$  if  $(s_i, m(\bar{d}), \psi_X, Y, s'_i) \in T_i$

When we build a TCIS, we simulate the transactions of the queues of the services by using one queue. By using the built TCIS, we can characterize each queue transaction of each service. In order to build a TCIS, we propose the algorithm 1.

---

**Algorithm 1:** TCIS\_Computing

---

**Input:** A set of Web services  $Q_i = (S_i, s_{0_i}, F_i, M_i, X_i, T_i)$ , for  $i = \{1, \dots, n\}$ . Empty queue  $Que$ .

**Output:** TCIS =  $(S, s_0, F, M, X, T)$

**begin**

$computedTr = T_1 \times \dots \times T_n$  for  $i = \{1, \dots, n\}$

**while**  $computedTr \neq \emptyset$  **do**

$incompatibility = \text{false}$

$currentTr = \{t_1, \dots, t_n\}$  where  $t_i \in T_i$  and  $\{t_1, \dots, t_n\} \in computedTr$

$computedTr = computedTr - currentTr$

**for each transition**  $(s_i, m(\bar{d}), \psi_i, Y_i, s'_i)$  **of each trace**  $t_i \in currentTr$  **do**

**if**  $Cycle\_Checked((s_i, m(\bar{d}), \psi_i, Y_i, s'_i))$  **then**

/\*If the message  $m$  is an input message, then  $polarity(m) = ?$  else  $polarity(m) = !*$ \*/

**if**  $(polarity(m) = '!')$  or  $(polarity(m) = '?')$  and  $m(\bar{d}) \in Que$  **then**

$t_{candidate} = (s_1 \dots s_i \dots s_n, m(\bar{d}), \psi, Y, s_1 \dots s'_i \dots s_n)$

**if**  $Clock\_Order(t_{candidate})$  **then**

$T = T \cup t_{candidate}$

**if**  $polarity(m(\bar{d})) = !$  **then**

$Que = Que + m(\bar{d})$

**else**

$Que = Que - m(\bar{d})$

**else**

$\perp$  there is a timed conflict,  $t_{candidate}$  is not accepted,  $incompatibility = \text{true}$

**else**

**if**  $polarity(m) = '?'$  and  $m(\bar{d}) \notin Que$  **then**

$\perp$  The current message is not yet available. We choose another transition of another service.

**else**

$\perp$  The current message is not yet available. We choose another transition of another service.

**if not incompatibility then**

/\*there are good traces\*/

**if**  $Que \neq \emptyset$  **then**

$\perp$  There is an extra message. The current traces combination of the services are not compatible

**else**

$\perp$  The current traces combination of the services are compatible

**else**

$\perp$  The current traces combination of the services are not compatible

**end**

---



In the worse case, the algorithm 1 is exponential in time. In fact, in order to check if a set of services are compatible, the cartesian product of the services traces could be parsed.

The algorithm 1 considers cycles when analyzing the compatibility of a choreography thanks to the algorithm 2.

---

**Algorithm 2** : Cycle\_Checked

---

```

Input: a transition  $(s, m(\bar{d}), \psi, Y, s')$ 
Output: boolean
begin
  if  $s'$  is already visited then
    if  $\text{polarity}(m) \neq !$  then
      /*The message could be sent infinitely*/
      mark the message  $m(\bar{d})$ 
      return true
    else
      if  $m(\bar{d}) \in \text{Queue}$  and  $m(\bar{d})$  is marked then
        | return true
      else
        | return false
    else
      | return true
  end

```

---

### 5.2 Making Explicit the Implicit Timed Constraints Dependencies

In order to make explicit the dependencies between the timed properties when building the TCIS, we use the *clock ordering process* we proposed in our previous work on Web service composition directed by client data [8]. The clock ordering process aims at defining an order between the different clocks of the different services when they are interacting together. The idea behind the *clock ordering process* is to define a total order between the different clocks of the services for each new TCSA transition. To explain the idea behind the clock ordering process, we use the following example.

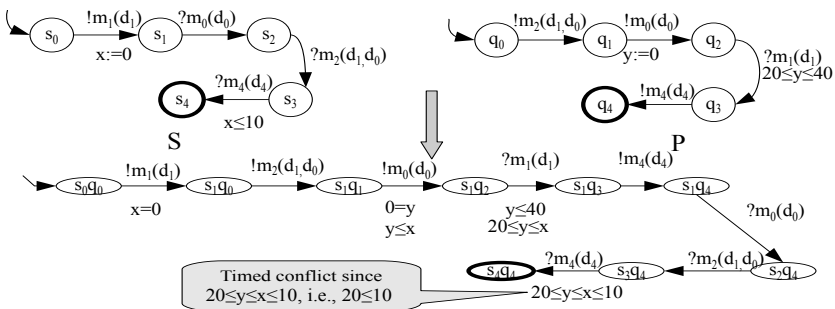


Fig. 5. Making explicit the implicit timed conflicts

*Example 3.* Let us consider the two timed conversational protocols of the services  $S$  and  $P$  depicted on Fig. 5. The service  $S$  can send the message  $m_1(d_1)$  and resets the clock  $x$ . So we build the TCIS transition  $(s_0q_0, !m_1(d_1), x = 0, s_1q_0)$ . Then,  $P$  sends the message  $m_2(d_1, d_0)$ . We build the TCIS transition  $(s_1q_0, !m_2(d_1, d_0), s_1q_1)$ . After that,  $P$  sends the message  $m_0(d_0)$  and resets the clock  $y$ . Since the clock  $x$  is reset before the clock  $y$ , hence we can define the order  $y \leq x$ . We build the corresponding TCIS transition  $(s_1q_1, !m_0(d_0), y = 0, y \leq x, s_1q_2)$ . As the message  $m_1(d_1)$  has been already sent by  $S$ , so  $P$  can consume it so that  $20 \leq y \leq 40$ . By propagating the order  $y \leq x$  defined above, we built the TCIS transition  $(s_1q_2, ?m_1(d_1), 20 \leq y \leq x, y \leq 40, s_1q_3)$ . Once the message  $m_1(d_1)$  is consumed,  $P$  sends the message  $m_4(d_4)$ . On the other side,  $S$  consumes the message  $m_0(d_0)$  that has been already sent by  $P$ . We build the TCIS transition  $(s_1q_4, ?m_0(d_0), s_2q_4)$ . Then  $S$  consumes the message  $m_2(d_1, d_0)$  that has been already sent by  $P$ . We build the TCIS transition  $(s_2q_4, ?m_2(d_1, d_0), s_3q_4)$ . Finally,  $S$  must consume the message  $m_4(d_4)$  within 10 units of time from sending the message  $m_1(d_1)$ . By propagating the order  $20 \leq y \leq x$  we defined above, we build the TCIS transition  $(s_3q_4, ?m_4(d_4), 20 \leq y \leq x \leq 10, s_4q_4)$ . The order  $20 \leq y \leq x \leq 10$  presents a timed conflict, i.e.,  $20 \leq 10$  and it is not possible to fire the transition  $(s_3q_4, ?m_4(d_4), 20 \leq y \leq x \leq 10, s_4q_4)$  (i.e., the message  $m_2(d_3)$  cannot be consumed). As remarked, without the timed propagation process, the timed conflict could not be detected.

In order to define the clock ordering process, we are using the algorithm 3.

---

**Algorithm 3:** Clock\_Order
 

---

**Input:** a transition  $(s_i, m_i(\bar{d}), \psi_i, Y_i, s'_i)$

**Output:** boolean

**begin**

**if**  $s_i$  is the initial state **then**

    | return true

**else**

    /\*propagation of the constraints of the form  $x \geq v$  (resp.  $x > v$ ) of a predecessor transition over the current transition\*/

**for each**  $q_{i-1} \in \psi_{i-1}$ , such that  $q_{i-1} = x \geq v$  or  $q_{i-1} = x > v$  of  $(s_{i-1}, m_{i-1}(\bar{d}), \psi_{i-1}, Y_{i-1}, s'_{i-1})$  **do**

$\psi_i = \psi_i \cup q_{i-1}$

      /\*The value of the clocks reset in the current transition is smaller than the value of a clock reset in the predecessor transition \*/

**for each**  $y = 0 \in Y_i$  and  $z = 0 \in Y_{i-1}$  **do**

        |  $\psi_i = \psi_i \cup y \leq z$

      /\*We propagate the order defined in the predecessor transition over the current transition\*/

**for each**  $z_1 \leq z_2 \in \psi_{i-1}$  **do**

        |  $\psi_i = \psi_i \cup z_1 \leq z_2$

**if**  $\exists v \leq y_0 \leq \dots \leq y_n \leq v' \in \psi_i$  where  $v' < v$  **then**

      | return false

**else**

      | return true

**end**

---

### 5.3 Characterization of Compatibility Classes

Previously, we have shown how we can build a deadlock free TCIS. In this section, we present how we can characterize the compatibility class of a set of asynchronous Web services. Before that, let us present the *subsumption* and *crossing* relations of protocols.

We say that a protocol  $Q_i$  is subsumed by a given TCIS if each transition of each trace of the protocol  $Q_i$  belongs to the given TCIS. In the context of our work, this means that for each transition  $(s_i, \alpha_i, \psi_i, Y_i, s'_i)$  of  $Q_i$ , there exists a transition  $(s_1 \dots s_i \dots s_n, \alpha'_i, \psi'_i, Y_i, s_1 \dots s'_i \dots s_n)$  of TCIS which can be preceded by a sequence of messages.

**Definition 4.** (*Protocol subsumption  $\subseteq_{tcis}$* )

Let  $TCIS = (S, s_0, F, M, X, T)$  be a computed TCIS and  $Q_i = (S_i, s_{0_i}, F_i, M_i, X_i, T_i)$  be a protocol of a Web service. We say that the TCIS subsumes  $Q_i$ , denoted  $Q_i \subseteq_{tcis} TCIS$  if for each trace  $s_{0_i} \xrightarrow{\alpha_0}_{\psi_0, Y_0} s_{1_i} \xrightarrow{\alpha_1}_{\psi_1, Y_1} \dots s_{n-1_i} \xrightarrow{\alpha_{n-1}}_{\psi_{n-1}, Y_{n-1}} s_{n_i}$  in  $Q_i$ , there exists a trace  $t: (s_0 \dots s_n) \xrightarrow{\vartheta_0} (s_0, \dots s_{0_i} \dots s_n) \xrightarrow{\alpha_0}_{\psi'_0, Y_0} (s_0, \dots s_{1_i} \dots s_n) \dots \xrightarrow{\vartheta_{n-1}} (s_0, \dots s_{n-1_i} \dots s_n) \xrightarrow{\alpha_{n-1}}_{\psi'_{n-1}, Y_{n-1}} (s_0, \dots s_{n_i} \dots s_n) \xrightarrow{\vartheta_n} (s_0, \dots s_n)$  of TCIS such that for  $i = 0, \dots, n$ ,  $(s_0 \dots s_n) \xrightarrow{\vartheta_i} (s_0 \dots s_n)$  is a (possibly empty) message sequence of TCIS.

We say that a protocol  $Q_i$  *crosses* a given TCIS if there is at least one trace of  $Q_i$  which is subsumed by this TCIS. Next, we present the formal definition of the *crossing* relation.

**Definition 5.** (*Protocol crossing  $\cap_{tcis}$* )

Let  $TCIS = (S, s_0, F, M, X, T)$  be a computed TCIS and  $Q_i = (S_i, s_{0_i}, F_i, M_i, X_i, T_i)$  be a protocol of a Web service. We say that  $Q_i$  **crosses** the TCIS, denoted  $Q_i \cap_{tcis} TCIS$  if  $\exists s_{0_i} \xrightarrow{\alpha_0}_{\psi_0, Y_0} s_{1_i} \xrightarrow{\alpha_1}_{\psi_1, Y_1} \dots s_{n-1_i} \xrightarrow{\alpha_{n-1}}_{\psi_{n-1}, Y_{n-1}} s_{n_i}$  in  $Q_i$ , such that there exists a trace  $t: (s_0 \dots s_n) \xrightarrow{\vartheta_0} (s_0, \dots s_{0_i} \dots s_n) \xrightarrow{\alpha_0}_{\psi'_0, Y_0} (s_0, \dots s_{1_i} \dots s_n) \dots \xrightarrow{\vartheta_{n-1}} (s_0, \dots s_{n-1_i} \dots s_n) \xrightarrow{\alpha_{n-1}}_{\psi'_{n-1}, Y_{n-1}} (s_0, \dots s_{n_i} \dots s_n) \xrightarrow{\vartheta_n} (s_0, \dots s_n)$  of TCIS such that for  $i = 0, \dots, n$ ,  $(s_0 \dots s_n) \xrightarrow{\vartheta_i} (s_0 \dots s_n)$  is a (possibly empty) message sequence of TCIS.

We say that a set of Web services constitutes a *full compatible choreography* if each protocol of each service is subsumed by the TCIS. However, when there are some traces that are subsumed by the TCIS and there are some traces that are not subsumed by the TCIS, we say that the set of Web services constitutes a *partial compatible choreography*. But, when the TCIS is an empty protocol, thus we say that the set of Web services constitutes a *full incompatible choreography*.

**Definition 6.** (*Choreography compatibility classes*)

Let  $TCIS = (S, s_0, F, M, X, T)$  be a computed TCIS of a set of Web services  $Q_i = (S_i, s_{0_i}, F_i, M_i, X_i, T_i)$  for  $i \in \{1, \dots, n\}$

- A set of Web services  $Q_i$  for  $i = \{1, \dots, n\}$  are said to be fully compatible if  $\forall i \in \{1, \dots, n\}, Q_i \subseteq_{tcis} TCIS$
- A set of Web services  $Q_i$  are said to be partially compatible if  $\exists i \in \{1, \dots, n\}, Q_i \not\subseteq_{tcis} Q$  and  $Q_i \cap_{tcis} TCIS$
- A set of Web services  $Q_i$  are said to be fully incompatible if  $TCIS = \emptyset$ .

## 6 Illustrative Example

By using the e-pension application we introduced in Section 2, let us now present an illustrative example of how analyzing the compatibility of the corresponding choreography. Initially, the three services client service (*CS*), prefecture service (*PS*) and health authority service (*HS*) are in their initial states. That means, the first *TCIS* configuration is  $(c_0p_0m_0)$ . From this configuration, *CS* enables the transition  $(c_0, !FileDeposit(file), c_1)$ . We build the *TCIS* transition  $(c_0p_0m_0, !FileDeposit(file), c_1p_0m_0)$ . From the configuration  $c_1p_0m_0$ , *CS* enables the transition  $(c_1, !formClaim(sn, reason), c_2)$ . We build the *TCIS* transition  $(c_1p_0m_0, !formClaim(sn, reason), c_2p_0m_0)$ . *HS* can consume the message *formClaim*(*sn, reason*) which has been already sent by *CS*. Thus, we build the *TCIS* transition  $(c_2p_0m_0, ?formClaim(sn, reason), c_2p_0m_1)$ . The new configuration becomes  $c_2p_0m_1$ . From this latter, *HS* enables the transition  $(m_1, !gettingForm(form), m_2)$ . We can build the *TCIS* transition  $(c_2p_0m_1, !gettingForm(form), c_2p_0m_2)$ . After sending the message *gettingForm*(*form*), *HS* sends the message *meeting*(*proposeDates*) and resets a clock  $z_1$ . In that case, we build the *TCIS* transition  $(c_2p_0m_2, !meeting(proposeDates), z_1 = 0, c_2p_0m_3)$ . From the new configuration  $c_2p_0m_3$ , *CS* can consume the available message *gettingForm*(*form*) which is already sent by *HS*. Once consumed, the clock  $x$  is reset. Since the clock  $z_1$  is reset before the clock  $x$ , hence we define the clock order  $x \leq z_1$ . Then, we build the *TCIS* transition  $(c_2p_0m_3, ?gettingForm$

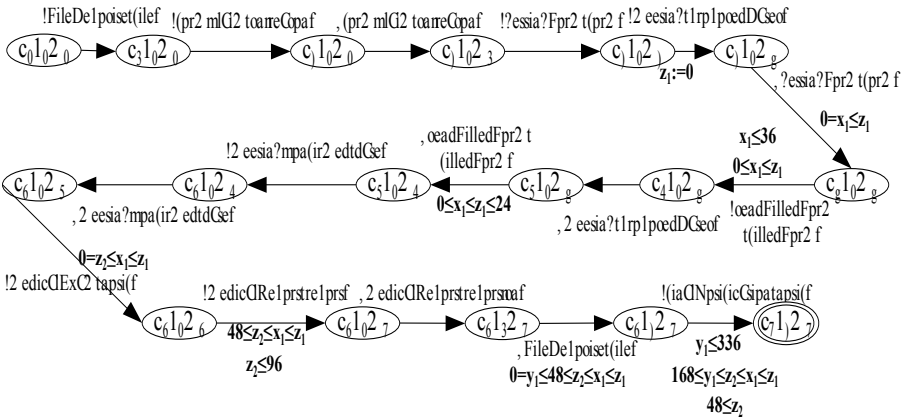


Fig. 6. TCIS of the e-pension application

(*for* - *m*),  $x = 0, x \leq z_1, c_3p_0m_3$ ). After that, *CS* can send the message *sendFilledForm*(*filledForm*) within 36 units of time ( $x \leq 36$ ). Regarding the clock order  $x \leq z_1$  we have defined above, we build the *TCIS* transition ( $c_3p_0m_3, !\text{sendFilledForm}(\text{filledForm}), 0 \leq x \leq z_1, x \leq 36, c_4p_0m_3$ ).

When the configuration  $c_5p_0m_3$  is reached, *HS* can consume the message *sendFilledForm*(*filledForm*) which is already sent (regarding the built *TCIS*) by *CS*. *HS* can consume the message *sendFilledForm*(*filledForm*) within 24 units of time ( $z_1 \leq 24$ ). Regarding the order we have defined above, we define the order  $0 \leq x \leq z_1 \leq 24$  that we associate to the *TCIS* transition ( $c_5p_0m_3, ?\text{sendFilledForm}(\text{filledForm}), 0 \leq x \leq z_1 \leq 24, c_5p_0m_4$ ). By applying the same steps, we build the deadlock free TCIS depicted on Fig. 6.

According to this TCIS, the three services *CS*, *PS*, and *HS* are fully compatible, since each protocol of each service is subsumed by the built TCIS. For example, according to the trace  $p_0 \xrightarrow{?medicalReport(report,sn)} p_1 \xrightarrow{?FileDeposit(file)} p_2 \xrightarrow{!finalNotification(notif)} p_3$  of *PS*, we can remark that each transition belongs to the trace of the *TCIS*. For example, if we consider the transition ( $p_0, ?medicalReport(report, sn), p_1$ ), we can see that from the TCIS initial configuration  $c_0p_0m_0$ , we can reach the configuration  $c_6p_0m_7$  that allows to fire the transition ( $c_6p_0m_7, ?medicalReport(report, sn), c_6p_1m_7$ ).

## 7 Related Work

Checking and analyzing in general the Web services features is an important investigation [4,3,2,5,12,11,10]. Particularly, in this paper we are interested in the compatibility analysis of a choreography in which the services support asynchronous communicating services. In general, the compatibility problem is based on analyzing message exchange sequences (conversations). In practice, other metrics affect the Web services compatibility, such as the kind of communication (synchronous or asynchronous) the services support. Besides, quantitative properties such as timed constraints plays a crucial role in Web services interaction.

In [4,3], the authors consider the sequence of messages that can be exchanged between two synchronous Web services. But, considering only message exchange sequences is not sufficient. To succeed a conversation, other metrics can have an impact such as timed properties which are not considered in [4,3]. Another important remark is that in [4,3], the authors consider synchronous Web services. Such assumption is very restrictive since the nature of Web services can be asynchronous. To overcome this limitation, we propose a compatibility checking approach for timed asynchronous services.

The compatibility framework presented in [12,13], that is an extension of the framework presented in [2], considers a more expressive timed constraints model. Although powerful, in some cases, the compatibility framework cannot detect some timed conflicts due to non-cancellation<sup>1</sup> constraints. In fact, the authors

<sup>1</sup> In [13] the non-cancellation constraints are called *C-Invoke*. They specify a time window within which a given message can be fired. Outside the window, the transition is disabled (exchanging the message results in an error).

deal only with synchronous communicating services. Thus, to discover timed conflicts, the authors are based on synchronizing the corresponding timed properties over messages. Therefore, this framework cannot be applied to discover the eventual timed conflicts in case of asynchronous Web services.

In [6], the authors handle the timed conformance problem which consists in checking if a given timed orchestration satisfies a global timed choreography. In this framework, the authors propose to deal with timed cost (i.e., the delay) of operations. According to our work, our aim is to detect conflicts that can arise when a set of Web services are interacting altogether. Whilst, [6] is not interested in analyzing the compatibility of a choreography but in checking if a given orchestration conforms to a choreography. So, one of the assumption is that the choreography does not hold timed conflicts.

We would like to mention that we are not using the techniques that have been proposed in the context of timed automata such as building region automata since the protocols of the services could be huge, consequently, building the structures such as region automata could be very complex and very huge. Moreover, in the context of our work such structure that gives rich information is not required. Whilst, by using the clock ordering process we are just defining an order between the different clocks in order to make explicit the eventual implicit timed conflicts.

## 8 Conclusion

In this paper, we presented a formal framework for analyzing the compatibility of a choreography. Unlike the proposed approaches, this framework caters for timed properties of asynchronous Web services. We presented how to model the timed behavior of Web services. To model timed properties, we propose to use the standard clocks of standard timed automata. In a choreography, when the services are interacting together, implicit timed dependencies can arise which could give rise to timed conflicts. We used the clock ordering process to discover such conflicts.

In a compatibility framework, it is important to characterize the executable interaction schema. To do so, we proposed an algorithm that allows to compute the timed choreography interaction schema of a set of Web services that can support asynchronous communications. We presented the *clock ordering* process that aims at discovering implicit timed conflict in a choreography. By using the mechanisms we proposed, we presented classes of timed choreography compatibility.

In our future work, we are interested in analyzing the compatibility of a choreography in which the instances of the involved services is not known in advance. Our aim is to provide primitives for defining dynamically the required instances for a successful choreography. Moreover, we plan to extend the proposed approach to support more complex timed properties when analyzing the compatibility of a set of Web services.

## References

1. Alur, R., Dill, D.L.: A theory of timed automata. *Theoretical Computer Science* 126(2), 183–235 (1994)
2. Benatallah, B., Casati, F., Ponge, J., Toumani, F.: On temporal abstractions of web service protocols. In: Pastor, Ó., Falcão e Cunha, J. (eds.) *CAiSE 2005*. LNCS, vol. 3520. Springer, Heidelberg (2005)
3. Benatallah, B., Casati, F., Toumani, F.: Analysis and management of web service protocols. In: *23rd International Conference on Conceptual Modeling* (November 2004)
4. Bordeaux, L., Salaün, G., Berardi, D., Mecella, M.: When are two web services compatible? In: Shan, M.-C., Dayal, U., Hsu, M. (eds.) *TES 2004*. LNCS, vol. 3324, pp. 15–28. Springer, Heidelberg (2005)
5. Diaz, G., Pardo, J.-J., Cambrotero, M.-E., Valero, V., Cuartero, F.: Verification of web services with timed automata. In: *Proceedings of the International Workshop on Automated Specification and Verification of Web Sites (WWV 2005)*. ENTCS, vol. 157, pp. 19–34 (2005)
6. Eder, J., Tahamtan, A.: Temporal conformance of federated choreographies. In: Bhowmick, S.S., Küng, J., Wagner, R. (eds.) *DEXA 2008*. LNCS, vol. 5181, pp. 668–675. Springer, Heidelberg (2008)
7. Guermouche, N., Godart, C.: Timed model checking based approach for compatibility analysis of synchronous web services. *Research report* (2008)
8. Guermouche, N., Godart, C.: Timed properties-aware asynchronous web service composition. In: *Proceedings of the 16th International Conference on Cooperative Information Systems (CoopIS 2008)*, Monterrey, Mexico, November 9–14, 2008, pp. 44–61 (2008)
9. Guermouche, N., Perrin, O., Ringeissen, C.: Timed specification for web services compatibility analysis. In: *International Workshop on Automated Specification and Verification of Web Systems (WWV 2007)*, San Servolo island, Venice, Italy, December 14, 2007, pp. 155–170 (2007)
10. Kazhamiakin, R., Pandya, P.K., Pistore, M.: Representation, verification, and computation of timed properties in web service compositions. In: *Proceedings of the IEEE International Conference on Web Services (ICWS)*, pp. 497–504 (2006)
11. Kazhamiakin, R., Pandya, P.K., Pistore, M.: Timed modelling and analysis in web service compositions. In: *Proceedings of the The First International Conference on Availability, Reliability and Security, ARES*, pp. 840–846. IEEE Computer Society Press, Los Alamitos (2006)
12. Ponge, J.: A new model for web services timed business protocols. In: *Atelier (Conception des systèmes d’information et services Web) SIWS-Inforsid* (2006)
13. Ponge, J., Benatallah, B., Casati, F., Toumani, F.: Fine-grained compatibility and replaceability analysis of timed web service protocols. In: Parent, C., Schewe, K.-D., Storey, V.C., Thalheim, B. (eds.) *ER 2007*. LNCS, vol. 4801, pp. 599–614. Springer, Heidelberg (2007)

# Evaluation Patterns for Analyzing the Costs of Enterprise Information Systems

Bela Mutschler<sup>1</sup> and Manfred Reichert<sup>2</sup>

<sup>1</sup> Business Informatics, University of Applied Sciences Ravensburg-Weingarten, Germany  
bela.mutschler@hs-weingarten.de

<sup>2</sup> Institute of Databases and Information Systems, University of Ulm, Germany  
manfred.reichert@uni-ulm.de

**Abstract.** Introducing *enterprise information systems* (EIS) is usually associated with high costs. It is therefore crucial to understand those factors that determine or influence these costs. Existing cost analysis methods are difficult to apply. Particularly, these methods are unable to cope with the dynamic interactions of the many technological, organizational and project-driven cost factors, which specifically arise in the context of EIS. Picking up this problem, in previous work we introduced the EcoPOST framework to investigate the complex cost structures of EIS engineering projects through qualitative cost evaluation models. This paper extends this framework and introduces a pattern-based approach enabling the reuse of EcoPOST evaluation models. Our patterns do not only simplify the design of EcoPOST evaluation models, but also improve the quality and comparability of cost evaluations. Therewith, we further strengthen our EcoPOST framework as an important tool supporting EIS engineers in gaining a better understanding of those factors that determine the costs of EIS engineering projects.

**Keywords:** Information Systems Engineering, Cost Analysis, Evaluation Models, Patterns.

## 1 Introduction

While the benefits of *enterprise information systems* (EIS) are usually justified by improved process performance [1], there exist no approaches for systematically analyzing related cost factors and their dependencies. Though software cost estimation has received considerable attention during the last decades [2] and has become an essential task in software engineering, it is difficult to apply existing approaches to EIS, particularly if the considered EIS shall support business processes. This difficulty stems from the inability of these approaches to cope with the numerous technological, organizational and project-driven cost factors which have to be considered for process-aware EIS (and which do only partly exist in data- or function-centered information systems). As example consider the costs which emerge when redesigning business processes. Another challenge deals with the many dependencies existing between different cost factors. Activities for *business process redesign*, for example, can be influenced by intangible impact factors like available *process knowledge* or *end user fears*. These dependencies, in turn, result in dynamic effects which influence the overall costs of EIS engineering projects. Existing evaluation techniques [3] are typically unable to deal



with such dynamic effects as they rely on too static models based upon snapshots of the considered software system.

What is needed is an approach that enables project managers and EIS engineers to model and investigate the complex interplay between the many cost and impact factors that arise in the context of EIS. This paper is related to the EcoPOST methodology, a sophisticated and practically validated, model-based methodology to better understand and systematically investigate the complex cost structures of EIS engineering projects [4,5]. Specifically, this paper extends previously described concepts [6,7] and introduces a pattern-based approach to enable the reuse of EcoPOST evaluation models. Using the presented evaluation patterns does not only simplify the design of EcoPOST evaluation models, but also improves the quality of EcoPOST cost evaluations.

Section 2 summarizes the EcoPOST methodology. This background information is needed for understanding this work. Section 3 introduces evaluation patterns for designing evaluation models. Section 4 deals with the use of our evaluation patterns. Section 5 discusses related work. Section 6 concludes with a summary.

## 2 The EcoPOST Cost Analysis Methodology - A Brief Summary

We designed the EcoPOST methodology [3,4,5,6,7] to ease the realization of process-aware EIS. The EcoPOST methodology comprises seven steps (cf. Fig. 1). *Step 1* concerns the comprehension of an evaluation scenario. This is crucial for developing problem-specific evaluation models. *Steps 2 and 3* deal with the identification of two different kinds of *Cost Factors* representing costs that can be quantified in terms of money (cf. Table 1): *Static Cost Factors* (SCFs) and *Dynamic Cost Factors* (DCFs).

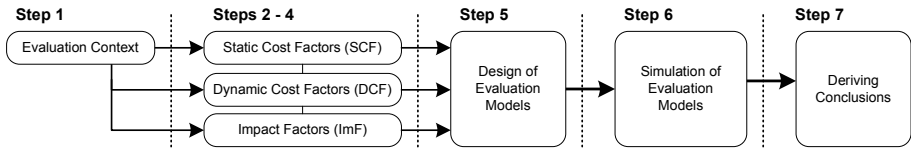
**Table 1.** Cost Factors

<i>SCF</i>	<i>Static Cost Factors</i> (SCFs) represent costs whose values do not change during an EIS engineering project (except for their time value, which is not further considered in the following). Typical examples: software license costs, hardware costs and costs for external consultants.
<i>DCF</i>	<i>Dynamic Cost Factors</i> (DCFs), in turn, represent costs that are determined by activities related to an EIS engineering project, e.g. process modelling, requirements elicitation and definition, process implementation and adaptation. These activities cause measurable efforts which, in turn, vary due to the influence of intangible <i>impact factors</i> .

*Step 4* deals with the identification of *Impact Factors* (ImFs), i.e., intangible factors that influence DCFs and other ImFs. We distinguish between organizational, project-specific, and technological ImFs. ImFs cause the value of DCFs (and other ImFs) to change, making their evaluation a difficult task to accomplish. As examples consider factors such as "End User Fears", "Availability of Process Knowledge", or "Ability to (re)design Business Processes". Also, ImFs can be static or dynamic (cf. Table 2).

**Table 2.** Impact Factors

<i>Static ImF</i>	Static ImFs do not change, i.e., they are assumed to be constant during an EIS engineering project; e.g., when there is a fixed degree of user fears, process complexity, or work profile change.
<i>Dynamic ImF</i>	Dynamic ImFs may change during an EIS engineering project, e.g., due to interference with other ImFs. As examples consider process and domain knowledge which is typically varying during an EIS engineering project (or a subsidiary activity).



**Fig. 1.** Basic EcoPOST Methodology (without Evaluation Patterns)

Unlike SCFs and DCFs the values of ImFs are not quantified in monetary terms. Instead, they are “quantified” by experts using qualitative scales describing the degree of an ImF. As known from software cost estimation models, such as COCOMO [2], qualitative scales we use comprise different “values” (ranging from “very low” to “very high”) expressing the strength of an ImF on a given cost factor.

Generally, dynamic evaluation factors (i.e., DCFs and dynamic ImFs) are difficult to comprehend. In particular, intangible ImFs (i.e., their appearance and impact in EIS engineering projects) are not easy to follow. When evaluating the costs of EIS engineering projects, therefore, DCFs and dynamic ImFs constitute a major source of misinterpretation and ambiguity. To better understand and to investigate the dynamic behavior of DCFs and dynamic ImFs, we introduce the notion of *evaluation models* as basic pillar of the EcoPOST methodology (Step 5; cf. Section 2.2). These evaluation models can be simulated (Step 6) to gain insights into the dynamic behavior (i.e., evolution) of DCFs and dynamic ImFs (Step 7). This is important to effectively control the design and implementation of EIS as well as the costs of respective projects. Note that EcoPOST evaluation models can be designed and simulations can be performed using any System Dynamics modeling and simulation tool. In our case, we used the tool “Vensim”.

## 2.1 Evaluation Models

In EcoPOST, dynamic cost/impact factors are captured and analyzed by evaluation models which are specified using the System Dynamics [8] notation (cf. Fig. 2). An evaluation model comprises SCFs, DCFs, and ImFs corresponding to model variables. Different types of variables exist. *State variables* can be used to represent dynamic factors, i.e., to capture changing values of DCFs (e.g., the “Business Process Redesign Costs”; cf. Fig. 2A) and dynamic ImFs (e.g., “Process Knowledge”). A state variable is graphically denoted as rectangle (cf. Fig. 2A), and its value at time  $t$  is determined by the accumulated changes of this variable from starting point  $t_0$  to present moment  $t$  ( $t > t_0$ ) – similar to a bathtub which accumulates at a defined moment  $t$  the amount of water poured into it in the past. Typically, state variables are connected to at least one *source* or *sink* which are graphically represented as cloud-like symbols (except for state variables connected to other ones) (cf. Fig. 2A). Values of state variables change through inflows and outflows. Graphically, both flow types are depicted by twin-arrows which either point to (in the case of an *inflow*) or out of (in the case of an *outflow*) the state variable (cf. Fig. 2A). Picking up again the bathtub image, an *inflow* is a pipe that adds water to the bathtub, i.e., inflows increase the value of state variables. An *outflow*, by contrast, is a pipe that purges water from the bathtub, i.e., outflows decrease the

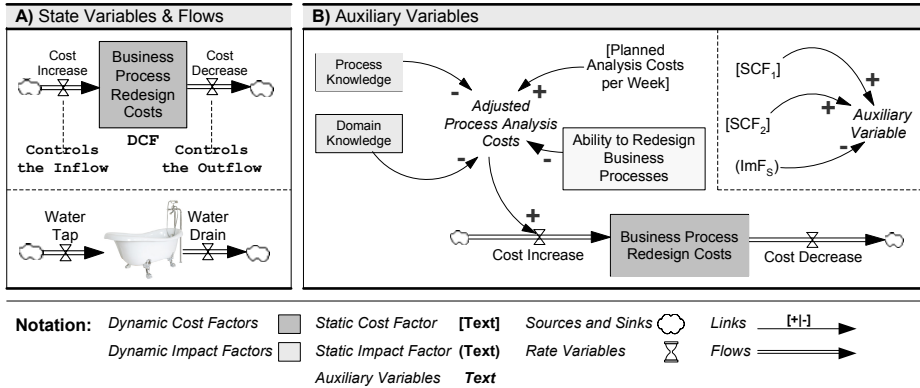


Fig. 2. Evaluation Model Notation and Initial Examples

value of state variables. The DCF "Business Process Redesign Costs" shown in Fig. 2A, for example, increases through its inflow ("Cost Increase") and decreases through its outflow ("Cost Decrease"). Returning to the bathtub image, we further need "water taps" to control the amount of water flowing into the bathtub, and "drains" to specify the amount of water flowing out. For this purpose, a *rate variable* is assigned to each flow (graphically depicted by a valve; cf. Fig. 2A). In particular, a rate variable controls the inflow/outflow it is assigned to based on those SCFs, DCFs, and ImFs which influence it. It can be considered as an interface which is able to merge SCFs, DCFs, and ImFs.

Besides state variables, evaluation models may comprise *constants* and *auxiliary variables*. Constants are used to represent static evaluation factors, i.e., SCFs and static ImFs. Auxiliary variables, in turn, represent intermediate variables and typically bring together – like rate variables – cost and impact factors, i.e., they merge SCFs, DCFs, and ImFs. As example consider the auxiliary variable "Adjusted Process Analysis Costs" in Fig. 2B. It merges the three dynamic ImFs "Process Knowledge", "Domain Knowledge" and "Ability to Redesign Business Processes", and the SCF "Planned Analysis Costs per Week". Both constants and auxiliary variables are integrated into an evaluation model with labeled arrows denoted as *links* (not flows). A *positive link* (labeled with "+") between x and y (with y as dependent variable) indicates that y will tend in the same direction if a change occurs in x. A *negative link* (labeled with "-") expresses that the dependent variable y will tend in the opposite direction if x changes.

EcoPOST evaluation models are useful for EIS engineers and project managers. However, the evolution of DCFs and dynamic ImFs is still difficult to comprehend. Thus, we added a simulation component to our evaluation framework (cf. Fig. 1).

## 2.2 Understanding Model Dynamics through Simulation

To enable simulation of an evaluation model we need to formally specify its behavior by means of a *simulation model*. We use *mathematical equations* for this purpose. Thereby, the behavior of each model variable is specified by one equation (cf. Fig. 3), which

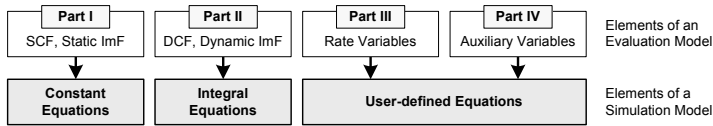


Fig. 3. Elements of a Simulation Model

describes how a variable is changing over time from simulation start. Details on the specification of simulation models can be found in [39].

Generally, results of a simulation enable EIS engineers to gain insights into causal dependencies between organizational, technological, and project-specific factors. This helps them to better understand resulting effects and to develop a concrete “feeling” for the dynamic implications of EcoPOST evaluation models. To investigate how a given evaluation model “works” and what might change its behavior, we simulate the dynamic implications described by it – a task which is typically too complex for human mind. In particular, we conduct “behavioral experiments” based on a series of simulation runs. During these simulation runs selected parameters are changed in a controlled manner to systematically investigate their effects within an evaluation model, i.e., to investigate how the output of a simulation will vary if its initial condition is changed. This procedure is also known as *sensitivity analysis*. Simulation outcomes can be further analyzed using graphical charts (generated by the used simulation tool).

### 2.3 Applying EcoPOST in Practice: Experiences and Lessons Learned

We applied the EcoPOST framework in several case studies in the automotive domain. This has made us aware of a number of *critical success factors* which foster the transfer of the EcoPOST framework into practice.

*First*, it is important that EcoPOST users get enough time to become familiar with the provided evaluation concepts. Note that EcoPOST exhibits a comparatively large number of different concepts and tools, such that it will need some time to effectively apply them. In practice, this can be a barrier for potential users. However, this complexity quickly decreases through gathered experiences.

*Second*, it is crucial that results of EcoPOST evaluations are carefully documented. This does not only enable their later reuse, it also allows to reflect on past evaluations and lessons learned as well as to reuse evaluation data. For that purpose, the *EcoPOST Cost Benefit Analyzer* can be used, which is a tool we developed to support the use of EcoPOST [3]. For example, it enables storage of complete evaluation scenarios, i.e., evaluation models and their related simulation models.

*Third*, evaluation models should be validated in an open forum where stakeholders such as policy makers, project managers, EIS architects, software developers, and consultants have the opportunity to contribute to the model evolution process.

*Finally*, the use of EcoPOST has shown that designing evaluation models can be a complicated and time-consuming task. Evaluation models can become complex due to the high number of potential cost and impact factors as well as the many causal dependencies that exist between them. Evaluation models we developed to analyze a

large EIS engineering project in the automotive domain, for example, comprise more than ten DCFs and ImFs and more than 25 causal dependencies [3]. Taking the approach described so far (cf. Section 2), each evaluation and each simulation model would have to be designed from scratch. Besides additional efforts, this results in an exclusion of existing modeling experience, and prevents the reuse of both evaluation and simulation models. In response to this problem, we introduce a set of reusable *evaluation patterns*.

### 3 EcoPOST Evaluation Patterns

EIS engineering projects often exhibit similarities, e.g., regarding the appearance of certain cost and impact factors. We pick up these similarities by introducing customizable patterns. This shall increase model reuse and facilitate practical use of our EcoPOST framework. *Evaluation patterns* (EPs) do not only ease the design and simulation of evaluation models, but also enable reuse of evaluation information. This is crucial to foster practical applicability of the EcoPOST framework.

Specifically, we introduce an *evaluation pattern* (EP) as a predefined, but customizable EcoPOST model, i.e., EPs can be built based on same elements as introduced in Section 2. An EP consists of an *evaluation model* and an associated *simulation model*. More precisely, each EP constitutes a template for a specific DCF or ImF as it typically exists in many EIS engineering projects. Moreover, we distinguish between *primary* EPs (cf. Section 3.2) and *secondary* ones (cf. Section 3.3).

A primary EP describes a DCF whereas a secondary EP represents an ImF. We denote an EP representing an ImF as secondary as it has a supporting role regarding the design of EcoPOST cost models based on primary EPs.

The decision whether to represent cost/impact factors as static or dynamic factors in EPs also depends on the model designer. Many cost and impact factors can be modeled both as static or dynamic factors. Consequently, EPs can be modeled in alternative ways. This is valid for all EPs discussed in the following.

#### 3.1 Research Methodology and Pattern Identification

As sources of our patterns (cf. Tables 3 and 4) we consider results from surveys [5], case studies [3,10], software experiments [4], and profound experiences we gathered in EIS engineering projects in the automotive domain. These projects addressed a variety of typical settings in enterprise computing which allows us to generalize our experiences.

To ground our patterns on a solid basis we first create a list of candidate patterns. For generating this initial list we conduct a detailed literature review and rely on our experience with EIS-enabling technologies, mainly in the automotive industry. Next we

**Table 3.** Overview of primary Evaluation Patterns and their Data Sources

Pattern Name	Discussed in Paper	Survey	Case Study	Literature	Experiment	Experiences
Business Process Redesign Costs	yes	x	x	x	-	x
Process Modeling Costs	yes	-	-	x	x	x
Requirements Definition Costs	yes	-	x	x	-	x
Process Implementation Costs	yes	x	x	x	x	x
Process Adaptation Costs	no	x	x	x	x	x

**Table 4.** Overview of secondary Evaluation Patterns and their Data Sources

Pattern Name	Discussed in Paper	Survey	Case Study	Literature	Experiment	Experiences
Process Knowledge	yes	x	-	x	x	x
Domain Knowledge	yes	x	-	x	x	x
Process Evolution	yes	x	-	x	-	x
Process Complexity	yes	-	-	x	-	-
Process Maturity	no	-	-	x	-	x
Work Profile Change	no	x	-	x	x	x
End User Fears	no	x	x	x	-	x

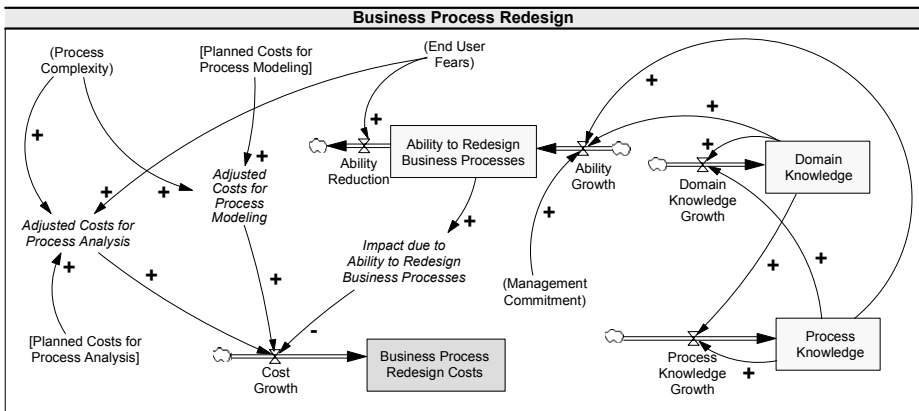
thoroughly analyze the above mentioned material to find empirical evidence for our candidate patterns. We then map the identified evaluation data to our candidate patterns and - if necessary - extend the list of candidate patterns.

A pattern is defined as a reusable solution to a commonly occurring problem. We require each of our evaluation patterns to be observed at least three times in different settings of literature and our empirical research. Only those patterns, for which enough empirical evidence exists, are included in the final list of patterns, which is presented in the following. Also note that these patterns represent a first baseline which clearly needs to be extended in future. This includes a deeper analysis of additional cost areas such as data modelling or system configuration efforts.

### 3.2 Primary Evaluation Patterns

**Business Process Redesign Costs.** The EP shown in Fig. 4 deals with the costs of business process redesign activities. Prior to EIS development such activities become necessary for several reasons. As examples consider the need to optimize business process performance or the goal of realizing a higher degree of process automation.

This EP is based on our experiences (from several process redesign projects) that business process redesign costs are primarily determined by two SCFs: "Planned Costs



**Fig. 4.** Primary Evaluation Pattern: Business Process Redesign Costs

for Process Analysis” and ”Planned Costs for Process Modeling”. While the former SCF represents planned costs for accomplishing interviews with process participants and costs for evaluating existing process documentation, the latter SCF concerns costs for transforming gathered process information into a new process design. Process redesign costs are thereby assumed to be varying, i.e., they are represented as DCF.

**Process Modeling Costs.** The EP shown in Fig. 5 deals with the costs of process modeling activities in EIS engineering projects. Such activities are typically accomplished to prepare the information gathered during process analysis, to assist software developers in implementing the EIS, and to serve as guideline for implementing the new process design (in the organization). Generally, there exist many notations that can be used to specify process models. Our EP, for example, assumes that process models are expressed as *event-driven process chains* (EPC).

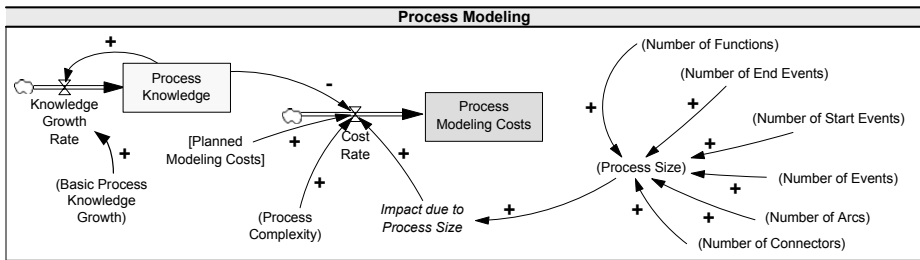


Fig. 5. Primary Evaluation Pattern: Process Modeling Costs

Basically, this EP (cf. Fig. 5) reflects our experiences that ”Process Modeling Costs” are influenced by three ImFs: the two static ImFs ”Process Complexity” and ”Process Size” (whereas the impact of process size is specified based on a table function transforming a given process size into an EcoPOST impact rating [3]) and the dynamic ImF ”Process Knowledge” (which has been also confirmed by our survey described in [3]). The ImF ”Process Complexity” is not further discussed here. Instead, we refer to [3] where this ImF has been introduced in detail. The ImF ”Process Size”, in turn, is characterized based on (estimated) attributes of the process model to be developed. These attributes depend on the used modeling formalism. As aforementioned, the EP from Fig. 5 builds on the assumption that the EPC formalism is used for process modeling. Taking this formalism, we specify process size based on the ”Number of Functions”, ”Number of Events”, ”Number of Arcs”, ”Number of Connectors”, ”Number of Start Events”, and ”Number of ”End Events”. Finally, the DCF ”Process Modeling Costs” is also influenced by the dynamic ImF ”Process Knowledge” (assuming that increasing process knowledge results in decreasing modeling costs). Level of process knowledge increases with costs (the comprehensiveness of the modeled process increases over time).

**Requirements Definition Costs.** The EP from Fig. 6 deals with costs for defining and eliciting requirements [3]. It is based on the two DCFs ”Requirement Definition Costs” and ”Requirement Test Costs” as well as on the ImF ”Requirements to be Documented”.

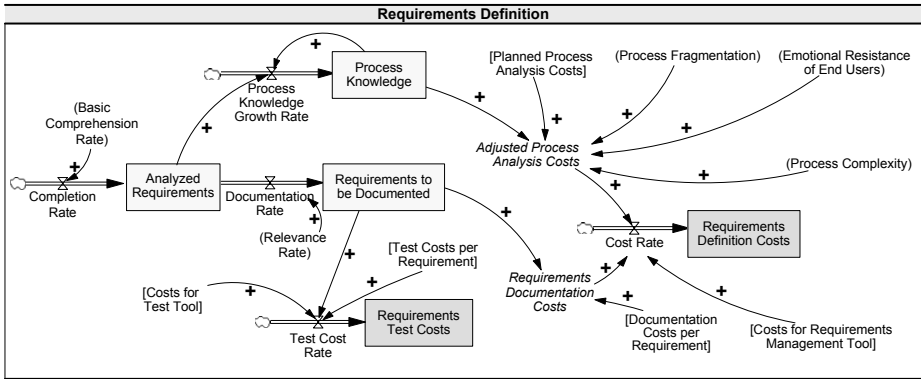


Fig. 6. Primary Evaluation Pattern: Requirements Definition Costs

This EP reflects our observation from practice that the DCF "Requirements Definition Costs" is determined by three main cost factors: costs for a requirements management tool, process analysis costs, and requirements documentation costs. Costs for a requirements management tool are constant and are therefore represented as SCF. The auxiliary variable "Adjusted Process Analysis Costs", in turn, merges the SCF "Planned Process Analysis Costs" with four process-related ImFs: "Process Complexity", "Process Fragmentation", "Process Knowledge", and "Emotional Resistance of End Users" (whereas only process knowledge is represented as dynamic ImF).

Costs for documenting requirements (represented by the auxiliary variable "Requirements Documentation Costs") are determined by the SCF "Documentation Costs per Requirement" and by the dynamic ImF "Requirements to be Documented". The latter ImF also influences the dynamic ImF "Process Knowledge" (resulting in a positive link from "Analyzed Requirements" to the rate variable "Process Knowledge Growth Rate"). "Requirements Test Costs" are determined by two SCFs ("Costs for Test Tool" and "Test Costs per Requirement") and the dynamic ImF "Requirements to be documented" (as only documented requirements need to be tested). Costs for a test tool and test costs per requirement are assumed to be constant (and are represented as SCFs).

**Process Implementation Costs.** The EP shown in Fig. 7 deals with costs for implementing a process and the interference of these costs through impact factors [3]. An additional EP (not shown here) deals with the costs caused by adapting the process(es) supported by an EIS. This additional EP is identical to the previous EP "Process Implementation Costs" – except for the additional ImF "Process Evolution".

### 3.3 Secondary Evaluation Patterns

**Process Knowledge.** Fig. 8 shows an EP which specifies the ImF "Process Knowledge", i.e., causal dependencies on knowledge about the process(es) to be supported.

**Domain Knowledge.** The EP from Fig. 9 deals with the evolution of domain knowledge along the course of an EIS engineering project. Our practical experiences allow for the



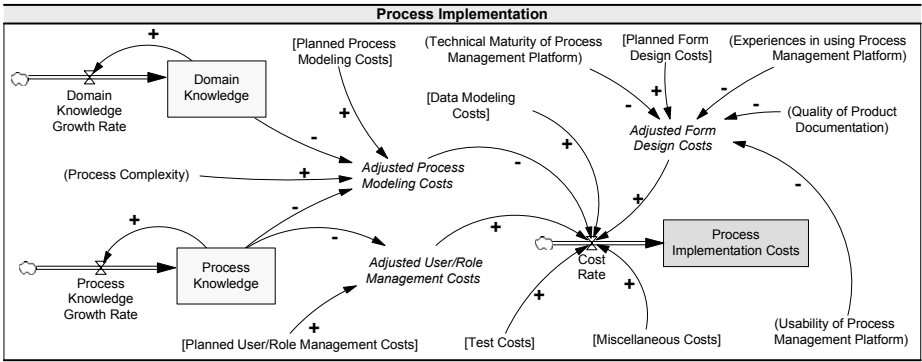


Fig. 7. Primary Evaluation Pattern: Process Implementation Costs

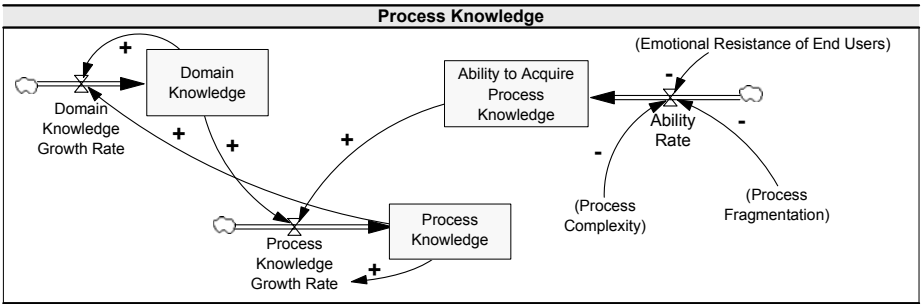


Fig. 8. Secondary Evaluation Pattern: Process Knowledge

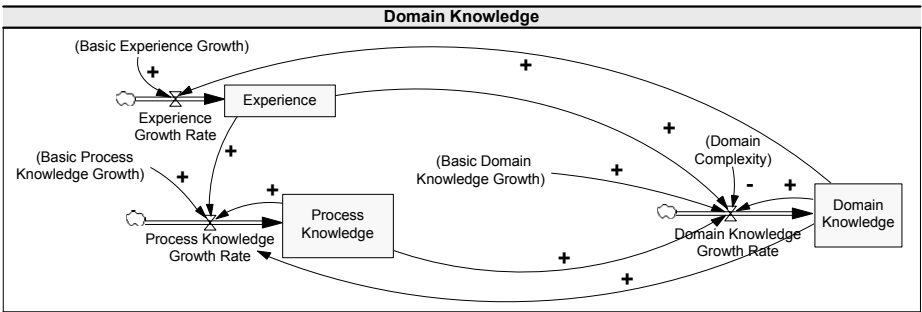


Fig. 9. Secondary Evaluation Pattern: Domain Knowledge

conclusion that "Domain Knowledge" is a dynamic ImF influenced by three other ImFs: the period an EIS engineer is working in a specific domain (captured by the dynamic ImF "Experience"), the dynamic ImF "Process Knowledge", and the complexity of the considered domain (represented by the static ImF "Domain Complexity").

**Process Evolution.** The EP shown in Fig. 10 covers the static ImF "Process Evolution". Specifically, it describes origins of process evolution. Basically, this EP reflects the assumption that business process evolution is caused by various drivers. Note that arbitrary drivers of evolution can be included in the EP.

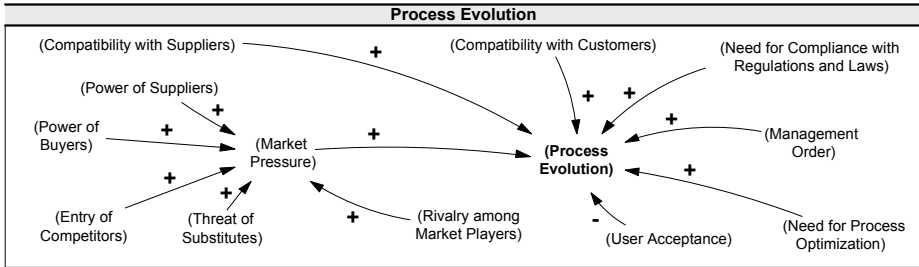


Fig. 10. Secondary Evaluation Pattern: Business Process Evolution

**Process Complexity.** The EP from Fig. 11 deals with the ImF "Process Complexity". Note that this EP does not specify process complexity itself, but defines it based on an easier manageable replacement factor. In our context, this replacement factor corresponds to the complexity of the process model describing the business process to be supported [11]. Thus, we extend process complexity to "Process Complexity / Process Model Complexity". The EP from Fig. 11 further aligns with the assumption that respective process models are formulated using EPC notation. According to the depicted EP, the static ImF "Process Complexity/Process Model Complexity" is determined by four other static ImFs: "Cycle Complexity", "Join Complexity" (JC), "Control-Flow Complexity" (CFC), and "Split-Join-Ratio" (SJR) (whereas the latter ImF is derived from the SCFs "Join Complexity" and "Control-Flow Complexity").

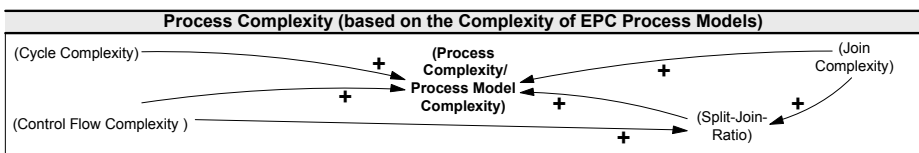


Fig. 11. Secondary Evaluation Pattern: Process Complexity

The complexity driver "Cycle Complexity" is confirmed in [12][13]. Arbitrary cycles, for example, can lead to EPC models without clear semantics (cf. [14] for examples). The ImF "Control-Flow Complexity" is characterized by [11]. It is based on the observation that the three split connector types in EPC models introduce a different degree of complexity. According to the number of potential post-states an AND-split is weighted with 1, an XOR-split is weighted with the number of successors  $n$ , and an OR-split is weighted with  $2n - 1$ . The sum of all connector weights of an EPC model is then denoted as "Control-Flow Complexity" [15]. The ImF "Join Complexity" can be defined

as the sum of weighted join connectors based on the number of potential pre-states in EPC models [16,17]. Finally, the mismatch between potential post-states of splits and pre-states of joins in EPC models is included as another driver of complexity. This mismatch is expressed by the static ImF "Split-Join-Ratio" (= JC/CFC) [16,17]. Based on these four static ImFs (or drivers of complexity), we derive the EP from Fig. 11. Thereby, an increasing cycle complexity results in higher process complexity. Also, both increasing CFC and increasing JC result in increasing process complexity. A JSR value different from 1 increases error probability and thus process complexity. It is important to mention that – if desired – other drivers of process complexity can be considered as well. Examples can be found in [13,17].

**Work Profile Change.** This EP (not shown here, but discussed in [3]) deals with change of end user work profiles (and the effects of work profile changes). More specifically, it relates the perceived work profile change to changes emerging in the five job dimensions of Hackman's *job characteristics model* [18,19]: (1) *skill variety*, (2) *task identity*, (3) *task significance*, (4) *autonomy*, and (5) *feedback from the job*. For each of these five core job dimensions, the emerging change is designated based on the level before and after EIS introduction.

**End User Fears.** This EP (not shown here, but discussed in [3] and [6]) is based on experiences which allow to conclude that the introduction of an EIS may cause end user fears, e.g., due to work profile change (i.e., job redesign) or changed social clues. Such fears can lead, for example, to emotional resistance of end users. This, in turn, can make it difficult to get needed support from end users, e.g., during process analysis.

## 4 Working with Patterns: Customization and Composition

Using EcoPOST evaluation patterns starts with the identification of those patterns which are relevant in a given context. After selecting a pattern, it might have to be customized. Note that EPs are applied in different evaluation context. Thereby, we have to distinguish between customization of an *evaluation model* (Step I) and of its corresponding *simulation model* (Step II). The former always requires the subsequent adaptation of the underlying simulation model, while the latter is also possible without customizing the associated evaluation model. Adapting an evaluation model can be achieved by adding or removing model variables, flows, or links. An example can be found in [3]. Correctness of customized EPs is ensured through EcoPOST-specific design rules [7].

Customizing a simulation model, by contrast, means to adapt functions of the simulation model, e.g. changes of SCF values. Customizing an EP can be quickly realized as a single EP does not require complex adaptations.

Another important feature with respect to the practical applicability of the EcoPOST framework concerns pattern composition (cf. Fig. 12). In particular, EcoPOST enables EIS engineers to compose new evaluation models by merging EPs. Unlike pattern customization, composing patterns is typically more complex and costly. Note that the number of composition variants might be quite large. Indeed, composition can be partly

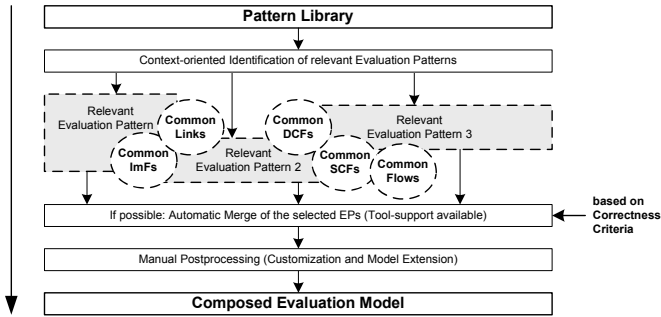


Fig. 12. Composition of Patterns

automated, but usually manual postprocessing becomes necessary. Respective concepts and merge algorithms are introduced in [3].

In a large case study [3] in the automotive domain, we have successfully applied EPs when designing complex evaluation and simulation models (see [3] for details).

## 5 Related Work

Boehm et. al [20] distinguish six categories of cost estimation techniques. They distinguish between *model-based approaches* (e.g., COCOMO, SLIM), *expertise-based approaches* (e.g., the Delphi method), *learning-oriented approaches* (using neural networks or case based reasoning), *regression-based approaches* (e.g., the ordinary least squares method), *composite approaches* (e.g., the Bayesian approach), and *dynamic-based approaches* (explicitly acknowledging that cost factors change over project duration). Picking up this classification, EcoPOST can be considered as an example of a dynamic-based approach (the other categories rely on static analysis models).

There are other formalisms that can be applied to unfold the dynamic effects caused by causal dependencies in EIS engineering projects. *Causal Bayesian Networks* (BN) [21], for example, promise to be a useful approach. BN deal with (un)certainly and focus on determining probabilities of events. A BN is a directed acyclic graph which represents interdependencies embodied in a given joint probability distribution over a set of variables. In our context, we are interested in the interplay of the components of a system and the effects resulting from this. BN do not allow to model feedback loops as cycles in BN would allow infinite feedbacks and oscillations that prevent stable parameters of the probability distribution. *Agent-based modeling* provides another promising approach. Resulting models comprise a set of reactive, intentional, or social agents encapsulating the behavior of the various variables that make up a system [22]. During simulation, the behavior of these agents is emulated according to defined rules [23]. System-level information (e.g., about intangible factors being effective in a EIS engineering project) is thereby not further considered. However, as system-level information is an important aspect in our approach, we have not further considered the use of agent-based modeling.

Patterns were first used to describe best practices in architecture [24]. However, they have also a long tradition in computer science, e.g., in the fields of software architecture (*conceptual patterns*), design (*design patterns*), and programming (*XML schema patterns*, *J2EE patterns*, etc.). Recently, the idea of using patterns has been also applied to more specific domains like workflow management [25][26] or inter-organizational control [27]. Generally, patterns describe solutions to recurring problems. They aim at supporting others in learning from available solutions and allow for the application of these solutions to similar situations. Often, patterns have a generative character. Generative patterns (like the ones we introduce) tell us how to create something and can be observed in the environments they helped to shape. Non-generative patterns, in turn, describe recurring phenomena without saying how to reproduce them.

Reusing System Dynamics models has been discussed before as well. On the one hand, authors like Senge [28], Eberlein and Hines [29], Liehr [30], and Myrtevit [31] introduce generic structures (with slightly different semantics) satisfying the capability of defining "components". On the other hand, Winch [32] proposes a more restrictive approach based on the parameterization of generic structures (without providing standardized modeling components). Our approach picks up ideas from both directions, i.e. we address both the definition of generic components as well as customization.

## 6 Summary and Future Work

This paper extends our EcoPOST framework, a model-based methodology to systematically investigate the complex cost structures of EIS engineering projects, by introducing the notion of evaluation pattern (EP). Each EP constitutes a template for specific cost or impact factors we encounter in typical EIS engineering projects. All EPs have been derived based on different pillars: results from two surveys [5], case studies [3][10], a controlled software experiment [4], and practical experiences gathered in EIS engineering projects.

In future work we will extend available EPs and apply them in a broader context in order to gather detailed experiences in applying EcoPOST. This includes the performance of additional experiments to analyze different use cases (e.g., customization and composition) for our patterns.

## References

1. Reijers, H.A., van der Aalst, W.M.P.: The Effectiveness of Workflow Management Systems - Predictions and Lessons Learned. *Int'l. J. of Inf. Mgmt.* 25(5), 457–471 (2005)
2. Boehm, B., Abts, C., Brown, A.W., Chulani, S., Clark, B.K., Horowitz, E., Madachy, R., Reifer, D., Steece, B.: *Software Cost Estimation with Cocomo 2*. Prentice-Hall, Englewood Cliffs (2000)
3. Mutschler, B.: *Analyzing Causal Dependencies on Process-aware Information Systems from a Cost Perspective*. PhD Thesis, University of Twente (2008)
4. Mutschler, B., Weber, B., Reichert, M.: Workflow Management versus Case Handling: Results from a Controlled Software Experiment. In: *Proc. ACM SAC 2008*, pp. 82–89 (2008)
5. Mutschler, B., Reichert, M., Bumiller, J.: Unleashing the Effectiveness of Process-oriented Information Systems: Problem Analysis, Critical Success Factors and Implications. *IEEE Transactions on Systems, Man, and Cybernetics*, 38(3), pp.280-291 (2008)

6. Mutschler, B., Reichert, M., Rinderle, S.: Analyzing the Dynamic Cost Factors of Process-aware Information Systems: A Model-based Approach. In: Krogstie, J., Opdahl, A.L., Sindre, G. (eds.) CAiSE 2007 and WES 2007. LNCS, vol. 4495, pp. 589–603. Springer, Heidelberg (2007)
7. Mutschler, B., Reichert, M.: On Modeling and Analyzing Cost Factors in Information Systems Engineering. In: Bellahsène, Z., Léonard, M. (eds.) CAiSE 2008. LNCS, vol. 5074, pp. 510–524. Springer, Heidelberg (2008)
8. Richardson, G.P., Pugh, A.L.: System Dynamics - Modeling with DYNAMO (1981)
9. Mutschler, B., Reichert, M.: Exploring the Dynamic Costs of Process-aware IS through Simulation. In: Krogstie, J., Opdahl, A.L., Sindre, G. (eds.) CAiSE 2007 and WES 2007. LNCS, vol. 4495, pp. 173–182. Springer, Heidelberg (2007)
10. Mutschler, B., Rijkpema, M., Reichert, M.: Investigating Implemented Process Design: A Case Study on the Impact of Process-aware Information Systems on Core Job Dimensions. In: Krogstie, J., Opdahl, A.L., Sindre, G. (eds.) CAiSE 2007 and WES 2007. LNCS, vol. 4495, pp. 379–384. Springer, Heidelberg (2007)
11. Cardoso, J.: Control-flow Complexity Measurement of Processes and Weyuker's Properties. In: Proc. Int'l. Enformatika Conference, vol. 8, pp. 213–218 (2005)
12. Cardoso, J., Mendling, J., Neumann, G., Reijers, H.: A Discourse on Complexity of Process Models. In: Proc. Int'l. Workshop on Business Process Design (BPI 2006), pp. 115–126 (2006)
13. Latva-Koivisto, A.: Finding a Complexity Measure for Business Process Models. Research Report, Helsinki University of Technology (2001)
14. Kindler, E.: On the Semantics of EPCs: Resolving the Vicious Circle. *Data Knowledge Engineering* 56(1), 23–40 (2006)
15. Gruhn, V., Laue, R.: Complexity Metrics for Business Process Models. In: Proc. 9th Int'l. Conf. on Business Information Systems, BIS 2006 (2006)
16. Mendling, J., Moser, M., Neumann, G., Verbeek, H.M.W., van Dongen, B.F., van der Aalst, W.M.P.: Faulty EPCs in the SAP Reference Model. In: Dustdar, S., Fiadeiro, J.L., Sheth, A.P. (eds.) BPM 2006. LNCS, vol. 4102, pp. 451–457. Springer, Heidelberg (2006)
17. Mendling, J., Moser, M., Neumann, G., Verbeek, H.M.W., van Dongen, B.F., van der Aalst, W.M.P.: A Quantitative Analysis of Faulty EPCs in the SAP Reference Model. BPM Center Report, BPM-06-08, BPMcenter.org (2006)
18. Hackman, R.J., Oldham, G.R.: Development of the Job Diagnostic Survey. *Journal of Applied Psychology* 60(2), 159–170 (1975)
19. Hackman, R.J., Oldham, G.R.: Motivation through the Design of Work: Test of a Theory. *Organizational Behavior & Human Performance* 16(2), 250–279 (1976)
20. Boehm, B., Abts, C., Chulani, S.: Software Development Cost Estimation Approaches - A Survey. Technical Report, USC-CSE-2000-505 (2000)
21. Jensen, F.V.: Bayesian Networks and Decision Graphs. Springer, Heidelberg (2002)
22. Brassel, K.H., Möhring, M., Schumacher, E., Troitzsch, K.G.: Can Agents Cover All the World? In: Simulating Social Phenomena. LNEMS, vol. 456, pp. 55–72 (1997)
23. Scholl, H.J.: Agent-based and System Dynamics Modeling: A Call for Cross Study and Joint Research. In: Proc. 34th Hawaii Int'l. Conf. on System Sciences (HICSS 2001) (2001)
24. Alexander, C., Ishikawa, S., Silverstein, M.: A Pattern Language. Oxford Press, Oxford (1979)
25. van der Aalst, W.M.P., ter Hofstede, A.H.M., Kiepuszewski, B., Barros, A.P.: Advanced Workflow Patterns. In: Scheuermann, P., Etzion, O. (eds.) CoopIS 2000. LNCS, vol. 1901, pp. 18–29. Springer, Heidelberg (2000)
26. Weber, B., Reichert, M., Rinderle-Ma, S.: Change Patterns and Change Support Features - Enhancing Flexibility in Process-Aware Information Systems. *Data and Knowledge Engineering* 66(3), 438–466 (2008)

27. Kartseva, V., Hulstijn, J., Tan, Y.H., Gordijn, J.: Towards Value-based Design Patterns for Inter-Organizational Control. In: Proc. 19th Bled E-Commerce Conference (2006)
28. Senge, P.M.: *The 5th Discipline - The Art and Practice of the Learning Organization*, 1st edn. Currency Publications (1990)
29. Eberlein, R.J., Hines, J.H.: *Molecules for Modelers*. In: Proc. 14th SD Conference (1996)
30. Liehr, M.: A Platform for System Dynamics Modeling - Methodologies for the Use of Pre-defined Model Components. In: Proc. 20th Int'l. System Dynamics Conference (2002)
31. Myrtveit, M.: Object-oriented Extensions to System Dynamics. In: Proc. 18th Int'l. System Dynamics Conference (2000)
32. Winch, G., Arthur, D.J.W.: User-Parameterised Generic Models: A Solution to the Conundrum of Modelling Access for SMEs? *System Dynamics Review* 18(3), 339–357 (2003)

# Using the REA Ontology to Create Interoperability between E-Collaboration Modeling Standards

Frederik Gailly and Geert Poels

Faculty of Economics and Business Administration, Ghent University  
{Frederik.Gailly, Geert.Poels}@ugent.be

**Abstract.** E-collaboration modeling standards like ISO/IEC 15944 and the UN/CEFACT Modeling Methodology (UMM) provide techniques, terms and reference models for modeling collaborative business processes. They offer a standardized approach for business partners to codify the business conventions, agreements and rules that govern business collaborations and to share business process information. Although effective in creating interoperability between organizations at the business process level, prospective business partners are required to commit to the same modeling standard. In this paper we show how the REA enterprise ontology can be used to semantically relate the ISO/IEC 15944 and UMM e-collaboration standards. Using the REA ontology as a shared business collaboration ontology, business partners can create interoperability between their respective business process models without having to use the same modeling standard.

**Keywords:** interoperability, business model, e-collaboration, business ontology.

## 1 Introduction

Today's fast paced global landscape calls for agile organizations that can swiftly integrate their business processes with that of other organizations. As many business processes are administrative processes that process information (as opposed to physical processes that handle and transform material goods), integrating such processes becomes a matter of creating interoperability, which is the ability of two different systems or components to exchange information and use information that has been exchanged [1].

Possible collaboration between companies gives rise to interoperability problems at different business levels: data, service, process and business level [2]. Over the years different kind of technologies (XML, schema standards and mapping, web services) have been proposed and used by different types of enterprise information integration tools (e.g. data warehouses, message mappings tools, virtual data integration) [3], which support the creation of interoperability at the data and service level (Functional Services View (FSV) of the open-EDI reference model [4]). Recently more attention is paid to solving the interoperability barriers at the process and enterprise level (Business Operational View (BOV)). Prospective business partners might use different languages to document and enact business processes (e.g. BPMN, BPEL, UML activities, EPC), which creates syntactic and semantic barriers to the integration



of their respective business processes. A possible solution is the use of e-collaboration modeling standards like the UN/CEFACT's Modeling Methodology (UMM) [5] and the ISO/IEC 15944 standard [6], to describe the global choreography between business partners involved in collaborative business processes. E-collaboration standards address the Business Operation View and provide a set of terms (e.g. business conventions, agreements and rules), reference models (e.g. a standard business process lifecycle) and techniques (e.g. UML activity diagrams) that can be used to model business processes in a way that they can be integrated easily with other business processes that also use these standards.

When two organizations wish to establish a B2B e-collaboration relationship and merge their respective business processes into one collaborative business process (e.g. integrating the sales process of the supplier with the acquisition process of the customer), they must agree on a common BOV e-collaboration standard. As currently none of the available standards is predominant in any industry, enterprises might end up with multiple representations of business processes articulated in many different languages and committing to different conceptualizations, a situation which is clearly suboptimal from the point of view of development and maintenance costs.

An alternative for imposing a modeling standard is the use of a shared ontology onto which the local ontologies of the different actors are mapped. Several proposals have been made to compare the abstract syntax and semantics of enterprise and business process modeling languages via a common meta-model (e.g. UEMML [7] for enterprise modeling languages and BPDM [8] for business process modeling languages). These meta-models are based on domain-independent ontologies (also called core or upper-level ontologies) or meta-meta-models.

Mappings between BOV e-collaboration standards like UMM [5] and the ISO/IEC 15944 standard [6] and these common meta-models may remove the syntactic barriers to the creation of collaborative business processes, but they do not guarantee the removal of domain-specific semantic barriers because they make abstraction of specific e-collaboration semantics like commitments, contracts and requiring money flows. Therefore, in this paper, we present an approach to the creation of interoperability between business processes of different partners wishing to engage in B2B e-collaboration by means of a shared domain-specific ontology which accounts for e-collaboration-specific semantics. In particular, we show how the Resource-Event-Agent (REA) enterprise ontology [9-11] can be used as a shared ontology for the UMM and ISO/IEC 15944 standards such that business process models articulated using one standard can easily be transformed into models articulated using the other standard, without losing domain-specific semantics.

This paper is structured as follows: Section 2 provides an overview of our approach. Section 3 briefly describes the REA ontology. Section 4 presents the UMM and ISO/IEC 15944 standards by means of an example based on the enhanced Telecommunications Operations Map (eTOM), which describes in detail the business processes required by a service provider in the telecommunications industry [12]. Section 5 then describes how the REA ontology is used to create interoperability between eTOM business process models developed using these two standards. Section 6 concludes the paper.

## 2 Ontology-Based Model Interoperability in the E-Collaboration Domain

According to Guarino [13], a core ontology is a formal representation of a conceptualization of the world (i.e. everything that exists or can exist), a domain ontology is a formal representation of the conceptualization of a particular part (i.e. a domain) of this world (e.g. business, medicine, sports) and a task ontology is formal representation of the conceptualization of a task executed within the world, possibly across different domains (e.g. planning, diagnosing, measuring). Domain and task ontologies should be defined as specializations of a core ontology meaning that their concepts add domain/task-specific meaning to the core ontology concepts.

Based on the ideas of Guarino, Guizzardi [14] defined the relationships between ontologies, conceptual modeling languages and models (Fig. 1). A model is articulated using some modeling language meaning that it instantiates the meta-model defining this language. The model is meant to represent an abstraction of a situation in the world (e.g. an order-to-cash process seen as an ordered collection of tasks). The abstraction of a given situation is constructed in terms of some domain or task conceptualization (e.g. workflows). This conceptualization is formally represented by a domain or task ontology and if this ontology is used to define the meta-model of the modeling language, then the language 'ontologically commits' to the conceptualization, meaning that the modeling language constructs derive their meaning from the concepts defined in the ontology.

Fig. 1 further shows that domain and task conceptualizations are generalized by real-world conceptualizations which are represented by core ontologies that generalize the domain and task ontologies used to represent the domain and task conceptualizations. These core ontologies define meta-meta-models (defining

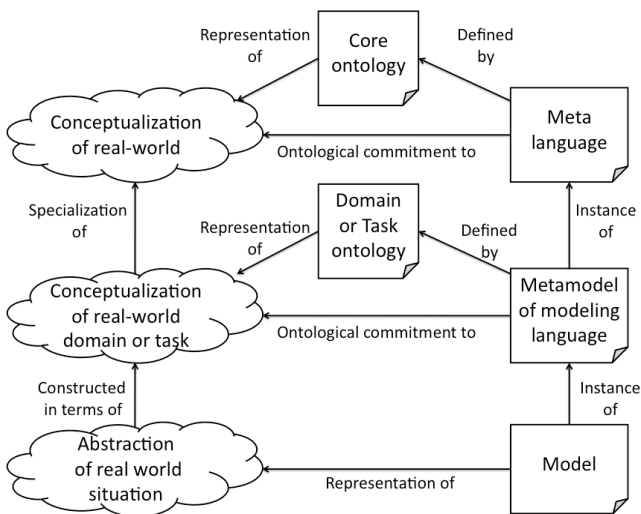


Fig. 1. Relation between conceptualization, ontology and modeling language

meta-languages) of which the meta-models of domain or task-specific modeling languages are instances. Fig. 1 also clarifies how interoperability between models articulated using different modeling languages can be established using ontologies. If the conceptualizations that the different languages commit to overlap and this shared conceptualization (i.e. intersection of conceptualizations) is represented by an ontology, then the meta-models of the languages need to be mapped onto the common ontology to explain how a phenomenon represented in one language would be represented in the other language.

Different researchers have recognized the potential of ontologies to provide precise semantics for conceptual modeling languages and have used core ontologies like BWW [15] or the Unified Foundational Ontology (UFO) [14] to evaluate the semantics of existing general purpose modeling languages like UML [16] and Petri-nets [17]. The mappings between the meta-models of these languages and the chosen reference ontology are then used to evaluate the ontological adequacy of the modeling languages. The ontological mappings identified are, however, not used to create interoperability between models articulated using different languages.

The Unified Enterprise Modeling language version 2 (UEML2), which was developed by the INTEROP Network of Excellence, follows the same approach but recognizes that the ontological mappings can also be used create interoperability between models. UEML2 acts as an intermediate language between existing enterprise modeling languages and facilitates interoperability between a wide variety of enterprise modeling languages and models [7]. In terms of Fig. 1, UEML2 is a meta-language defined by the BWW ontology, which is an upper-level ontology, and as such, UEML2 abstracts from specific domain or task semantics. Consequently, domain or task specific semantics attached to special-purpose modeling languages may get lost when mapping the meta-models of these languages to UEML2.

To preserve the domain or task-specific semantics of special-purpose modeling languages a mapping onto a domain or task ontology instead of a core ontology is required. For instance, for creating interoperability between models created using different BOV e-collaboration standards, a mapping onto a domain ontology for e-collaboration will preserve more of the domain semantics than a mapping onto a core ontology. The goal of this paper is to demonstrate such a mapping for the e-collaboration domain.

Ciocoiu and Nau described in general how ontologies can be used to formally define translations between models developed in different languages [18]. Their approach consists of three steps:

1. Define a function that can be used to convert a model articulated in a specific language into a model articulated in a logic-based language. This function is called a logical rendering function and results in a logical rendering of the model.
2. Define an interpretation for the concepts of the language using a shared ontology. Put differently, during this phase the semantics of the modeling language constructs are defined using an ontology. Together the logical rendering function and the ontology-based interpretation make it possible to convert a model in a specific language into an ontology-based model.
3. The results of the previous steps can now be used to create an ontology-based translation between models such that every model can be explained in terms of the conceptualization specified by the shared ontology.

We use the approach of Ciocoiu and Nau to create interoperability between models articulated in the UMM and ISO/IEC 15944 modeling standards (Fig. 2). Instead of using a formal first order predicate logic language, the description logic language OWL was used to specify the ontologies and the logical renderings of the models. We decided to use OWL because OWL is considered as the standard ontology language. Moreover, OWL provides mapping constructs that are used to relate terms in different ontologies. There also exist different easy to use OWL description logic reasoners which can be easily integrated into ontology engineering tools. Reasoners such as Pellet<sup>1</sup> can be used for consistency checking (identify contrary facts), concept satisfiability (verify that all classes can be populated with instances), classification (create a complete subclass hierarchy by identifying subclass relations) and realization (compute the direct types for individuals).

Fig. 2 shows how model interoperability is created by transforming two collaborative business process models (model 1 and model 2) that are developed following two different BOV e-collaboration standards (ISO/IEC 15944 and UMM) into OWL renderings of the models (model 1' and model 2') that refer to their own local ontology (OeBTO<sup>2</sup> and UMM ontology<sup>3</sup>). By mapping these local ontologies onto a global, shared OWL-formalized ontology (i.e. the REA ontology), both models can be interpreted in terms of the same global ontology.

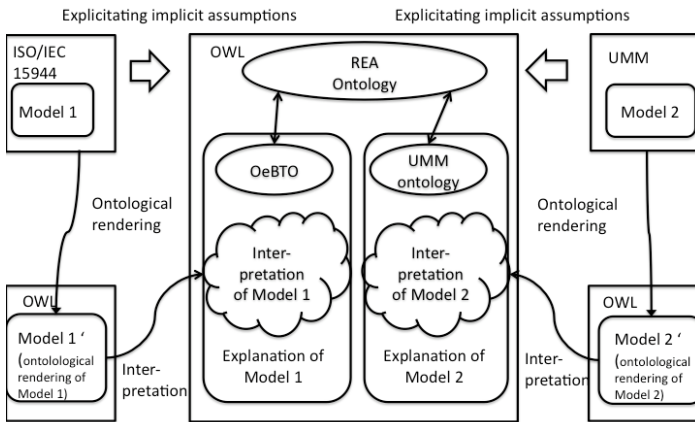


Fig. 2. Ontology-based e-collaboration model translation

### 3 The Resource Event Agent Enterprise Ontology

The Resource-Event-Agent ontology (REA-ontology) [10, 11] originates in a semantic data model for accounting proposed in [9]. The subject domain of REA can be described as 'the enterprise'. Hence REA is an *enterprise ontology* and as such it

<sup>1</sup> <http://clarkparsia.com/pellet/>

<sup>2</sup> The Open-EDI Business Transaction Ontology (OeBTO) is the UML class diagram specification of the collaborative business process conceptualization that underlies the ISO/IEC 15944 standard.

<sup>3</sup> The (unnamed) ontology of the UMM method is also specified using UML class diagrams.

provides a description of explicit knowledge about enterprises that is structured in terms of concepts, a concept classification based on ‘is-a’ relations, relations between concepts other than ‘is-a’ relations, and a set of axioms that hold for these relations. Other well-known business ontologies are the e<sup>3</sup>-value ontology [19] and e-BMO [20], which focus on different, though largely overlapping aspects of business and as a result can be unified with REA to create a more encompassing business ontology, as demonstrated in [21].[21]

The particular conceptualization of enterprises specified by REA is heavily influenced by REA’s accounting background. Primary attention is paid to what changes the value of the enterprise (i.e. recording these value-affecting events and the value composition of the enterprise is what we call ‘accounting’) and who can be held accountable for this (i.e. accounting enables control of the organization and its members). So enterprise reality is described in terms of **R**esources (having value), **E**vents (affecting this value) and **A**gents (having control over the resources and being responsible for the events); hence the name of the ontology. The conceptualization includes additional business concepts to predict future value changes (e.g. contracts, terms and commitments) or to specify policies for value creation, transfer and consumption (e.g. business policies).

Recently the REA-ontology has also been extended with a procedural component that states that all REA concepts can be considered as *business objects* which all have a defined lifecycle determining their states and state transitions. REA events may be decomposed into *business events* which each may trigger state transitions for multiple *business objects*. A *business process* is then defined as an aggregate of REA events. In our previous work [22, 23], we already formalized the REA-ontology in OWL<sup>4</sup> starting from an UML representation of the ontology. This OWL formalization is used in this paper as a reference ontology to create interoperability between models developed using the ISO/IEC 15944 and UMM BOV e-collaboration standards.

## 4 E-Collaboration Modeling Standards

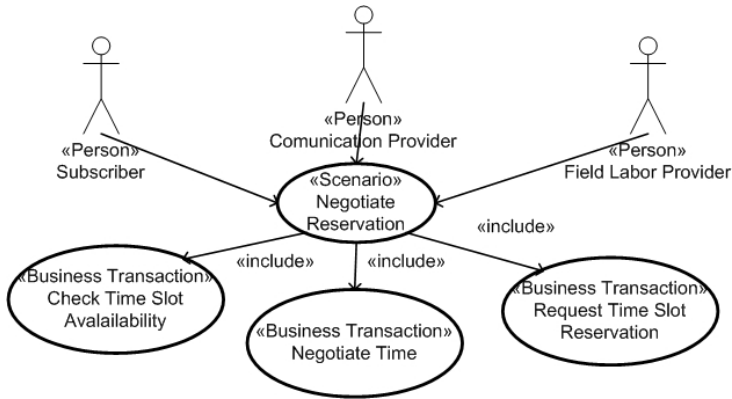
In the next subsections the ISO/IEC 15944 standard and the UMM method, and their ontologies, are introduced by means of the enhanced Telecommunications Operations Map (eTOM) example [12]. In this paper we focus on the eTOM process used by a communication provider to reserve and schedule field technicians for the installation and configuration of goods and services.

### 4.1 ISO/IEC 15944 Standard

The ISO/IEC 15944 standard provides a methodology and tool for specifying shared business practices (as part of shared business transactions) in the form of scenarios, scenario attributes, roles, information bundles and semantic components. This is achieved by developing standard specifications of generally accepted business transaction conventions and practices as scenarios and scenario components. Fig. 3 and Fig. 4 show representations related to the ‘reserve and schedule field technicians’ process following the ISO/IEC 15944 standard. UML is the modeling language of choice in the standard though it is extended with domain-specific stereotypes.

---

<sup>4</sup> See <http://purl.org/REA/REAontology.owl>



**Fig. 3.** Use case diagram for Negotiate Reservation following ISO/IEC 15944

The modeling of ‘the reserve and schedule field technicians’ process starts with a use case scenario Negotiate Reservation that interacts with 3 different *persons*<sup>5</sup>: the Subscriber, the Communication Provider and the Field Labor Provider (see the use case diagram shown in Fig. 3). This use case scenario<sup>6</sup> includes three separate *business transactions*: Check Time Slot Availability, Negotiate Time and Request Time Slot Reservation. Check Time Slot Availability queries the Field Labor Provider for available time slots which results in a list of available time slots. Negotiate Time negotiates an actual time slot using the available slots that correspond to the wishes of the Subscriber. Finally this time slot is reserved by the Communication Provider by means of Request Time Slot Reservation.

The Request Time Slot Reservation *business transaction* is described in more detail in Fig. 4. This UML activity diagram shows the *business events* that transition the state of the *business transaction entities*. The OeBTO defines a *business transaction entity* as a computable representation of any real-world entity that participates, occurs or is materialized during a *business transaction*. In Fig. 4 two *business transactions entities* are distinguished: Field Labor and Labor Contract. The *business event* RequestTimeSlotReservation brings the *business transaction entities* Field Labor and Labor Contract into a ‘proposed’ state (called a *business transaction entity state*). If the Field Labor Provider accepts this request (AcceptTimeSlot Reservation *business event*) then the state of the two *business transaction entities* shown will change into ‘specified’ and the Communication Provider will receive the contract (Receive Contract *business event*), which ends the *business transaction*. The *business transaction entity states* are said to reside in the collaboration space between the Communication Provider and the Field Labor Provider and allow each partner to determine simultaneously what the exact status of the overall business collaboration is.

<sup>5</sup> Concepts in *italic* refer to concepts that are defined in the ontologies of the e-collaboration standards (here OeBTO; in the next sub-section the UMM-ontology).

<sup>6</sup> Scenario is not an OeBTO concept, but is defined in the open-EDI reference model [3]. According to this reference model, business processes may consist of one or more use case scenarios.

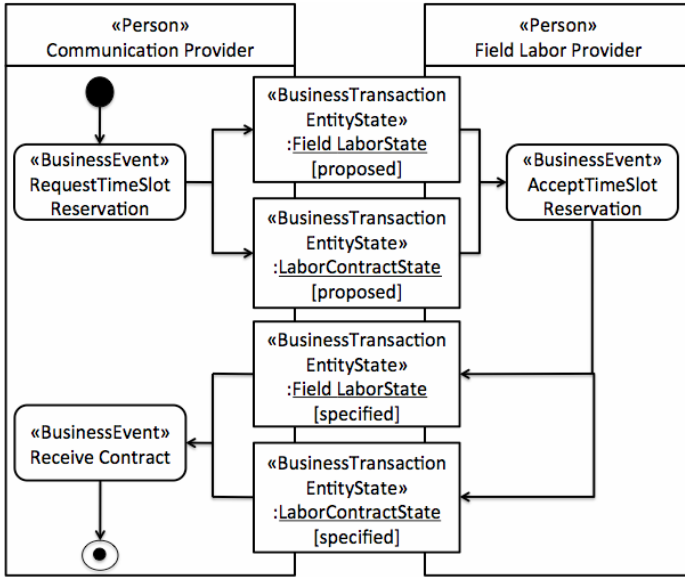


Fig. 4. Activity diagram Request Time Slot Reservation following ISO/IEC 15944

Note that in Fig. 3 and Fig. 4 the UML stereotype extension mechanism was employed for specifying BOV e-collaboration concepts like *person*, *business event*, *business transaction*, *business transaction entity* and *business transaction entity state*. Following the framework shown in Fig. 1, these domain-specific stereotypes represent the ontological commitment of the modeling standard to the conceptualization of the business e-collaboration domain that is specified by the Open-edi Business Transaction Ontology (OeBTO).

The model interoperability approach that we follow (see Fig. 2) requires that the OeBTO UML class diagrams must be translated into OWL. Specifying an OWL formalization of the OeBTO is straightforward because the ISO/IEC 15944 standard contains clear definitions of the concepts and the relationships between the concepts. As UML class diagrams are used to visualize the concepts and relationships, the Ontology Definition Metamodel (ODM) [24] was used to transform these UML class diagrams into OWL specifications. Additionally we followed the guidelines provided by W3C’s Semantic Web Best Practices and Deployment Working Group [25] to formalize the part-whole relations of the OeBTO because OWL does not contain specific primitives for part-whole relations and as a result the ODM lacks this type of mapping.

#### 4.2 UN/CEFACT’s Modeling Methodology

The UN/CEFACT’s Modeling Methodology (UMM) is an UML modeling approach to design a global choreography for the business processes between business partners [5]. UMM distinguishes three views (i.e. Business Domain View, Business Requirements View and Business Transaction View) each covering a set of well

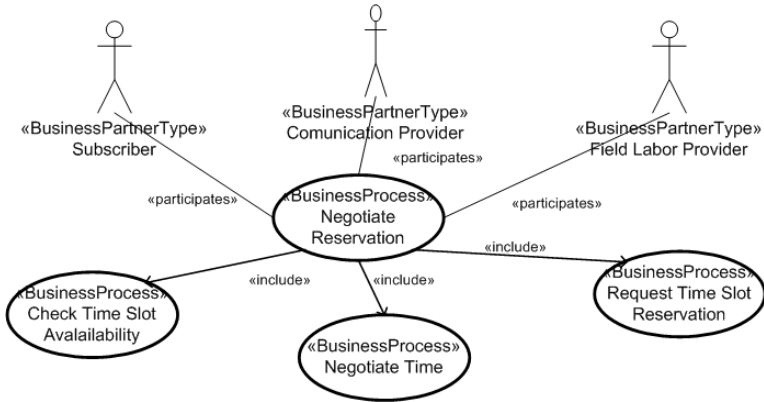


Fig. 5. Use case diagram Negotiate Reservation following UMM

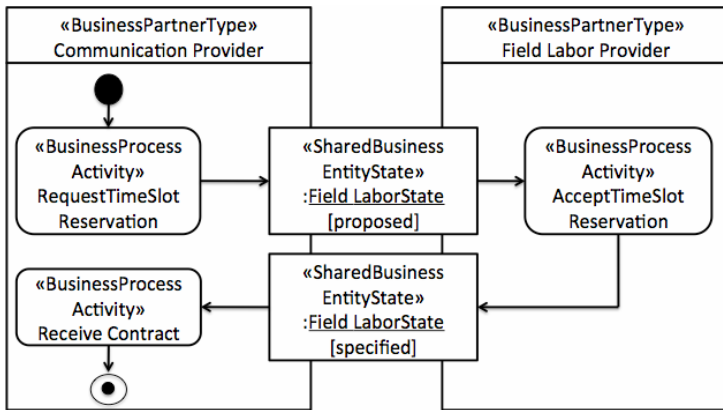


Fig. 6. Activity diagram Request Time Slot Reservation following UMM

defined artifacts of the open-EDI BOV and which can be used to model a complete business collaboration. In this paper only the Business Domain View and the Business Requirements View are considered because these views correspond to the abstraction level employed in ISO/IEC 15944.

In Fig. 5 and Fig. 6 the eTOM ‘negotiate reservation’ process is modeled using UMM. Although for humans the mappings between these models and the models developed using the ISO/IEC 15944 standard (Fig. 3 and 4) are obvious because we used the same names for the modeled concepts, in reality these mappings must be determined based on the semantic annotations in the models. Because both standards focus on the same semantic domain, there exists for different modeling concepts one-to-one mappings (e.g. an OeBTO *person* is an UMM *business partner type*). However, the semantic domains of both standards are not identical, but only overlapping, which means that some of the concepts defined in the ontology of one standard have no counterpart in the other standard. For instance, the ISO/IEC 15944

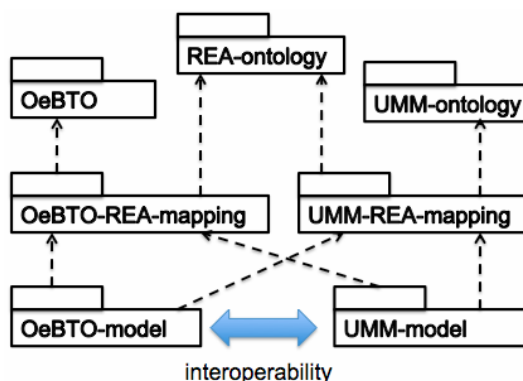


standard prescribes the use of *economic commitments*, a concept that is not used in the UMM approach. An *economic commitment* is a promise by one *person* to transfer *economic resources* to another *person* at some specified point in the future. Consequently, in Fig. 4 (following the ISO/IEC 15944 standard) the *business event* RequestTimeSlotReservation changes the state of two *business transaction entities*: FieldLabor and LaborContract, whereas in Fig. 6 the corresponding UMM *business process activity* only changes the state of the UMM FieldLabor *business entity* as LaborContract (an *economic commitment* in ISO/IEC 15944) cannot be represented using UMM concepts. Another difference between the two standards is that UMM makes an explicit difference between *shared business entities* and *internal business entities*. This difference is only implicitly present in ISO/IEC 15944 models by positioning the *transaction business entity states* in the collaboration space between two *persons*.

Compared to the ISO/IEC 15944 standard, UMM pays less attention to the formal specification of the business collaboration conceptualization that underlies the standard (i.e. the development of an BOV e-collaboration ontology) and more to the development of a domain-specific language (based on UML) that can be used to develop models for the different views. The syntax of the UMM views is defined by extending standard UML meta-models like the use case meta-model and the activity diagram meta-model with stereotypes. The semantics of the defined stereotypes are partly described in text and partly graphically presented by means of UML class diagrams. Based on the available meta-model descriptions we have developed an OWL formalization of the domain-specific concepts (e.g. *business process*, *business process activity*, *business entity state*, etc.) used in UMM.

## 5 E-Collaboration Model Interoperability via the REA Ontology

The federated model interoperability approach outlined in Fig. 2 requires that the concepts of the local ontologies (OeBTO and UMM) are mapped onto the concepts of the global ontology (REA). At this stage the mapping of the ontologies is done manually because the used ontologies are only lightweight ontologies. However in the future the ontologies are extended, we need to further investigate how ontology mapping techniques [26] can be incorporated in our approach. As all ontologies are represented in OWL (see sections 3 and 4), the equivalentClass and equivalentProperty OWL mapping constructs can be used for this purpose. The OWL equivalentClass construct allows one to say that a class description (representing an ontology concept) has exactly the same class extension as another class description. Put differently, given their definitions, both class extensions would always contain the same set of individuals. Similarly, the OWL equivalentProperty construct can be used to state that two properties have the same property extension. OWL properties are binary relations on individuals, i.e. they link two individuals together. For instance, the OWL OeBTO ontology contains a *custody* property that links *person* individuals to *economic resource* individuals (meaning that the person has physical control over the resource). Two properties have the same extension if they link the same individuals together.



**Fig. 7.** OWL import hierarchy for creating interoperability between models developed using the ISO/IEC 15944 and UMM e-collaboration standards

Fig. 7 gives an overview of the OWL ontology files (represented as UML packages) that are needed to create interoperability between e-collaboration models developed using the ISO/IEC 15944 and UMM standards<sup>7</sup>. To define the ontology mappings, separate files (*UMM-REA-mapping.owl* and *OeBTO-REA-mapping.owl*) were created that each import a local ontology (*OeBTO.owl* and *UMM-ontology.owl*) and the REA ontology (*REA-ontology.owl*). Then, based on our understanding of the ontologies, we created ontology mappings. Next, the classification service of the Pellet reasoner was invoked to identify subclass relations between the classes of the local and global ontology and to detect equivalent classes, in order to verify our ontology mappings.

With the help of the reasoner some problems were detected that necessitated changes in the OWL ontologies or mappings. For instance, based on the UML class diagram specification of the UMM ontology and the examples given in [4], we defined *business partner type* as being disjoint with *business entity*. However, by specifying in the UMM-REA mapping that a *business partner type* is equivalent to an REA *economic agent* (which is a logical conclusion based on the definitions of these concepts and the granularity and intended use of the UMM and REA ontologies), Pellet infers that *business partner type* is a subclass of *business entity* (because UMM *business entity* is equivalent to REA *business object* which is the supertype of REA *economic agent*), which contradicts the disjointness constraint in our UMM OWL ontology. Table 1 gives an extract of the final version of the mappings defined in the *UMM-REA-mapping.owl* and *OeBTO-REA-mapping.owl* files<sup>8</sup>.

Creating interoperability between an ISO/IEC 15944 model and an UMM model requires that the models are transformed into ontology-based models that can be interpreted in terms of the business collaboration conceptualizations that are specified by the ontologies referred to. Consequently, UML models like the ones described in rendering transformation in Fig. 2). In doing so, we decided to define the models as

<sup>7</sup> Readers can retrieve these OWL ontology files from the following URL:

<http://www.managementinformatics.ugent.be/CaiseInteropPaper.zip>

<sup>8</sup> The extract contains those mappings that are required to create interoperability between the ISO/IEC 15944 and UMM eTOM models used for the eTOM example in the paper.

**Table 1.** Extract of OWL mappings e-collaboration ontologies and REA-ontology

e-collaboration ontology concept	REA-ontology concept
OeBTO:BusinessTransactionEntity	REA:BusinessObject
UMM:BusinessEntity	REA:BusinessObject
OeBTO:BusinessEvent	REA:BusinessEvent
UMM:BusinessProcessActivity	REA:BusinessEvent
OeBTO:BusinessTransaction	REA:BusinessProcess
UMM:BusinessProcess	REA:BusinessProcess
OeBTO:BusinessTransactionEntityState	REA:BusinessObjectState
UMM:BusinessEntityState	REA:BusinessObjectState
OeBTO:Person	REA:EconomicAgent
UMM:BusinessPartnerType	REA:EconomicAgent
OeBTO:EconomicCommitment	REA:Commitment
OeBTO:EconomicContract	REA:EconomicContract

ontology instantiations (i.e. using OWL individuals that are classified as instances of existing ontology classes) and not by specializing ontology concepts (i.e. by creating new OWL classes as subclasses of existing ontology classes), because this corresponds more to the UML view where the abstract syntax of a language is defined in a meta-model which can then be instantiated to create models.

The OeBTO-model.owl file (see Fig. 7) is the ontological rendering of the ISO/IEC 15944 model for the eTOM example (see sub-section 4.1) and will be used to illustrate the creation of model interoperability.<sup>9</sup> This OeBTO-model.owl file imports indirectly the OeBTO OWL ontology by importing the OeBTO-REA-mapping.owl file (see Fig. 7). In Fig. 8 the UML OWL profile of the ODM [22] is used to visualize the result of the ontological rendering of part of the ISO/IEC 15944 eTOM example, more specifically the activity diagram shown in Fig. 4. This result contains only the classifiers and instance-of relationships drawn with solid lines in Fig. 8.

After importing also the UMM-REA-mapping.owl file, the classification service of the reasoner (Pellet) is invoked to detect equivalent classes in the OeBTO and UMM OWL ontologies. Two classes are equivalent if they map onto the same class in the REA OWL ontology. The realization service of the reasoner (Pellet) can now be invoked to classify the OWL individuals (representing the elements of the ISO/IEC 15944 activity diagram shown in Fig. 4) as instances of the UMM OWL ontology classes. The inferred assertions after reasoning are shown in dashed lines in Fig. 8. For instance, after reasoning, the LaborContractState and the FieldLaborState are classified as UMM *business entity states*. These two OWL individuals are instances of the OeBTO BusinessTransactionEntityState class that is equivalent to the UMM BusinessEntityState class, and therefore they are inferred as instances of this UMM OWL class.

<sup>9</sup> Analogously, the UMM-model.owl file is the ontological rendering of the UMM model for the eTOM example (see sub-section 4.2).

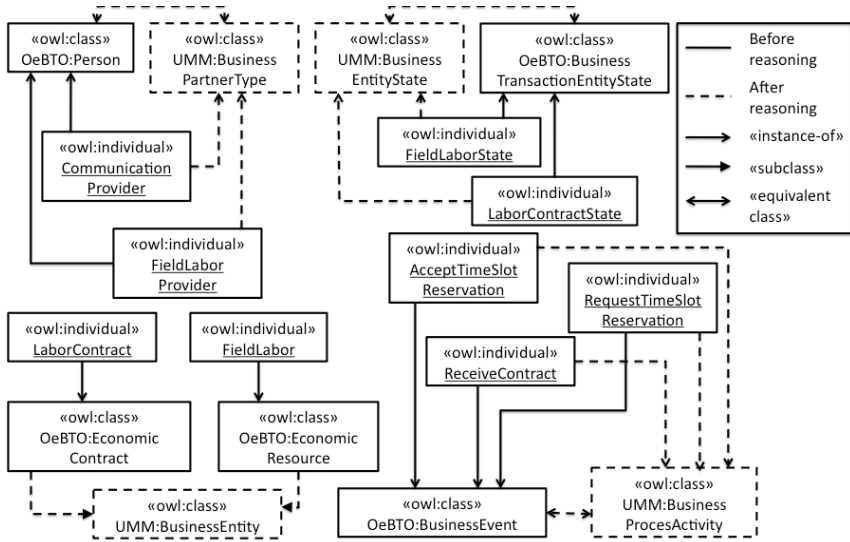


Fig. 8. OeBTO-model before (solid lines) and after reasoning (dashed lines)

Some OeBTO concepts have no direct counterpart in the UMM ontology but because of the classification hierarchy created after reasoning, the reasoner also makes assertions for these concepts. For instance, the LaborContract is an OWL individual of the OeBTO EconomicContract class which has no counterpart in the UMM ontology. However, based on the classification hierarchy, LaborContract is classified as an UMM *business entity* because the UMM BusinessEntity class is a superclass of the OeBTO EconomicContract class and LaborContract is an instance of EconomicContract.

## 6 Conclusions and Future Work

The goal of this paper was to show how ontologies can be used to create interoperability between models developed in different languages that have an overlapping semantic domain. More specifically, the REA-ontology was used to create interoperability between models that were developed following two different e-collaboration modeling standards: ISO/IEC 15944 and the UN/CEFACT Modelling Methodology. Model interoperability was realized by formalizing the ontologies that are part of these e-collaboration standards using OWL and relating these ontologies with the REA ontology using OWL ontology mapping constructs. The mappings between both e-collaboration ontologies and the REA-ontology were then used by an OWL description logic reasoner (Pellet) to identify equivalences between the two modeling standards. Next, we showed by means of an example that a model developed using one standard can be interpreted in terms of the other standard. First, the model was made ontology-based by translating it into an instantiation of the OWL formalized ontology that is part of the standard used to develop the model. Second, a reasoner was invoked that, based on the previously identified equivalences, automatically classifies the model elements as instances of the ontology concepts of the other standard.

A current limitation of our approach is that not all information in an e-collaboration model is preserved when translating it into an ontology-based model. Both e-collaboration standards prescribe the use of general purpose modeling techniques (use case diagrams and activity diagrams) to articulate collaborative business process models. Most of the constructs of these techniques (e.g. actors, use cases, actions, object nodes) are stereotyped so that they derive their meaning from the ontology of the standard. Other constructs (e.g. include relationships between use cases, control flow arrows in activity diagrams) derive their meaning from the meta-model of the modeling technique and are not ‘overloaded’ with domain-specific semantics. As a consequence, these model elements are not classified as instances of the domain ontology classes, so do not appear in the ontology-based models, leading to a loss of information. Therefore an ontology-based model cannot replace the model it is derived from. To transform a model developed using one standard into a model following the other standard, both the original model and the ontology-based model are needed. For the ISO/IEC 15944 and UMM standards, such a translation is straightforward because both standards prescribe the use of the same general purpose modeling techniques. For standards that use different modeling techniques the translation problem becomes more complicated. Further research is required to investigate how solutions for creating mappings between modeling languages (e.g. solutions like UEML [7] and BPDM [8]) can be integrated into our approach.

As future research we plan to build an extension to the REA ontology that integrates the UMM Business Transaction View. This more detailed view on business transactions is needed to extend our approach of creating interoperability between BOV e-collaboration models to Functional Service View (FSV) models. In this context we will also need to evaluate how our approach is related to existing information integration research and industry solutions [3]. Additionally the proposed approach needs to be validated using more complex practical examples.

## References

1. IEEE Computer Society. Standards Coordinating Committee: IEEE standard computer dictionary: a compilation of IEEE standard computer glossaries, 610. Institute of Electrical and Electronics Engineers, New York, NY, USA (1990)
2. Chen, D.: Enterprise Interoperability Framework. In: Petit, M., Latour, T. (eds.) CAISE 2006 EMOI-INTEROP workshop. University Namur Press, Luxembourg (2006)
3. Bernstein, P.A., Haas, L.M.: Information integration in the enterprise. *Communications of the ACM* 51, 72–79 (2008)
4. ISO: Open-EDI reference model. ISO standard 14662. ISO (1997)
5. UN/CEFACT: UN/CEFACT’s Modeling Methodology (UMM): UMM Meta Model – Foundation Module Version 1.0 Technical Specification 2006-10-06 (2006)
6. ISO: Information technology – Business Operational View – Part 4: Business transaction scenarios – Accounting and economic ontology (ISO/IEC 15944-4 ). ISO (2006)
7. Opdahl, A., Berio, G.: Interoperable language and model management using the UEML approach. In: International workshop on Global integrated model management. ACM, Shanghai (2006)
8. OMG: Business Process Definition MetaModel Volume I: Common Infrastructure (BPDM) V1. OMG (2008), <http://www.omg.org/spec/BPDM/20080501>

9. McCarthy, W.E.: The REA Accounting Model: A Generalized Framework for Accounting Systems in A Shared Data Environment. *The Accounting Review* 57, 554–578 (1982)
10. Geerts, G., McCarthy, W.E.: An Ontological Analysis of the Economic Primitives of the Extended-REA Enterprise Information Architecture. *International Journal of Accounting Information Systems* 3, 1–16 (2002)
11. Geerts, G., McCarthy, W.E.: Policy-Level Specification in REA Enterprise Information Systems. *Journal of Information Systems* 20, 37–63 (2006)
12. TeleManagement Forum: enhanced Telecom Operations Map (2008), <http://www.tmforum.org/browse.aspx?catID=1647>
13. Guarino, N.: Formal Ontology and Information Systems. In: *International Conference on Formal ontology in Information Systems (FOIS 1998)*, pp. 3–15. IOS Press, Trento (1998)
14. Guizzardi, G.: On Ontology, ontologies, Conceptualizations, Modeling Languages, and (Meta)Models. In: Vasilecas, O. (ed.) *Frontiers in Artificial Intelligence and Applications, Databases and Information Systems IV*. IOS Press, Amsterdam (2007)
15. Wand, Y., Weber, R.: An Ontological Model of an Information System. *IEEE Transactions on Software Engineering* 16, 1282–1292 (1990)
16. Opdahl, A.L., Henderson-Sellers, B.: Ontological Evaluation of the UML Using the Bunge–Wand–Weber Model. *Software and Systems Modeling* 1, 43–67 (2002)
17. Soffer, P., Wand, Y.: On the notion of soft-goals in business process modeling. *Business Process Management Journal* 11, 663–679 (2005)
18. Ciocoiu, M., Nau, D.S.: Ontology-Based Semantics. In: *Seventh International Conference on Principles of Knowledge Representation and Reasoning (KR 2000)*. Breckenbridge, Colorado (2000)
19. Gordijn, J., Akkermans, J.M.: E3-value: Design and Evaluation of e-Business Models. *IEEE Intelligent Systems* 16, 11–17 (2001)
20. Osterwalder, A.: The Business Model Ontology - a proposition in a design science approach. *Ecole des Hautes Etudes Commerciales*. University of Lausanne, Lausanne (2004)
21. Andersson, B., Bergholtz, M., Edirisuriya, A., Ilayperuma, T., Johannesson, P., Grégoire, B., Schmitt, M., Dubois, E., Abels, S., Hahn, A., Gordijn, J., Weigand, H., Wangler, B.: Towards a Reference Ontology for Business Models. In: Embley, D.W., Olivé, A., Ram, S. (eds.) *ER 2006*. LNCS, vol. 4215, pp. 482–496. Springer, Heidelberg (2006)
22. Gailly, F., Laurier, W., Poels, G.: Positioning and Formalizing the REA enterprise ontology. *Journal of Information Systems* 22, 219–248 (2008)
23. Gailly, F., Poels, G.: Towards Ontology-driven Information Systems: Redesign and Formalization of the REA Ontology. In: Abramowicz, W. (ed.) *BIS 2007*. LNCS, vol. 4439, pp. 245–259. Springer, Heidelberg (2007)
24. OMG: Ontology Definition Metamodel: OMG Adopted Specification (ptc/06-10-11). Object Management Group (2006)
25. Rector, A., Welty, C.: Simple part-whole relations in OWL Ontologies, W3C Semantic Web Best Practices and Deployment Working Group (2005), <http://www.w3.org/2001/sw/BestPractices/OEP/SimplePartWhole/index.html>
26. Kalfoglou, Y., Schorlemmer, M.: Ontology mapping: the state of the art. *Knowledge Engineering Review* 18, 1–31 (2003)

# Value-Based Service Modeling and Design: Toward a Unified View of Services

Hans Weigand<sup>1</sup>, Paul Johannesson<sup>2</sup>, Birger Andersson<sup>2</sup>, and Maria Bergholtz<sup>2</sup>

<sup>1</sup> Tilburg University, P.O. Box 90153,  
5000 LE Tilburg, The Netherlands  
H.Weigand@uvt.nl

<sup>2</sup> Royal Institute of Technology  
Department of Computer and Systems Sciences, Sweden  
{pajo,ba,maria}@dsv.su.se

**Abstract.** Service-oriented architectures are the upcoming business standard for realizing enterprise information systems, thus creating a need for analysis and design methods that are truly service-oriented. Most research on this topic so far takes a software engineering perspective. For a proper alignment between business and IT, a service perspective at the business level is needed as well. In this paper, a unified view of services is introduced by means of a service ontology, service classification and service layer architecture. On the basis of these service models, a service design method is proposed and applied to a case from the literature. The design method capitalizes on existing value modeling approaches.

**Keywords:** service design, business services, value modeling.

## 1 Introduction

Service-Oriented Architectures provide major advantages for today's enterprise information systems by presenting the interfaces that loosely coupled connections require [Pa05]. Web services [WS04] seem to become the preferred implementation technology for realizing the SOA promise of service sharing and interoperability.

In the view of Papazoglou and Van den Heuvel [PH06], (web) service design and development is about identifying the right services, organizing them in a manageable hierarchy of composite services and choreographing them together for supporting a business process. A *business service* (in this context, a service implementing a business process) can be composed of finer-grained services, which in turn are being supported by infrastructure services. Following previous work on SOAD [Zi04], they distinguish between top-down, bottom-up and meet-in-the-middle approaches and discuss major principles of service design such as low coupling and high cohesion. Although the paper provides useful criteria, it considers service design mainly as a software engineering problem, which in our view is not sufficient. As stated in [NL07], “the current trend toward a service-oriented enterprise necessitates a formal characterization of business architecture that reflects service-oriented business thinking”. The starting point for design should be the business level at which services

can be identified that provide value to customers and can be offered in an economically viable way.

IBM's SOAD has evolved into SOMA [AG08] described as "a software development life-cycle method invented and initially developed in IBM for designing and building SOA-based solutions". SOMA advocates a meet-in-the-middle approach. Domain decomposition is a top-down analysis that starts with analysis of the functional areas in the business domain and of the business processes. This is complemented by a bottom-up asset analysis. The two lines are pulled together by Goal-Service Modeling (GSM). SOMA incorporates many more methods and techniques, including conceptual data modeling, and advocates a fractal model for software development. On the basis of the information available, we infer that SOMA is a comprehensive method based on real industrial experience but the same remark applies as above: it does not consider service-oriented business thinking.

In contrast, [Sp08] introduces the notion of service system in an abstract way that can be applied to the business. The service system is defined as an open system capable of improving the state of another system through sharing or applying its resources (providing value), and capable of improving its own state by acquiring external resources (receiving value). The way of thinking is resonating the earlier work of Norman as well as Prahalad on co-creation of value [NR93, PK08], and existing work in the IS field on value modeling [Go00, Mc82].

What is lacking so far is a principled way of linking the two different branches of service science [Al08]. If a business is viewed as a service system, what does this have to do with service-oriented software design?

The objective of this paper is precisely to establish this link by (a) proposing a generic service model in which software services are defined as a service subtype, and (b) providing a method for service identification that starts from a value-based business model. Section 2 sets the stage by reviewing the most relevant business modeling approaches. It also introduces a running example, taken from [AG08] to allow easy comparison. In section 3, we provide a general service model rooted in the REA ontology, and extend it with a fully integrated service layer architecture. Section 4 introduces a value-based service identification method and illustrates it using the running example. Section 5 contains conclusions and directions for future research.

## 2 Business Modeling and Service Systems

There exist a number of approaches, languages, and ontologies for business modeling in literature, e.g., [Go00], [Di05], [Us96] and [Mc82]. In [An06] the e3value [Go00] and the REA ontologies [Mc82] were compared (together with a third business ontology – the BMO [Os04]) in order to establish a common reference business ontology. One result of that comparison was a set of mappings between e3value and REA indicating strong similarities between the concepts of the two.

### 2.1 Business Modeling

The Resource-Event-Agent (REA) ontology was formulated originally in [Mc82] and has been developed further, e.g. in [UM03, Ge06, Hr06]. Its conceptual origins can be



traced back to traditional business accounting. REA was originally intended as a basis for accounting information systems and focused on representing increases and decreases of value in an organization. REA has been extended to form a foundation for enterprise information systems architectures [Hr06], and it has also been applied to e-commerce frameworks [UM03]. The following is a short overview of the core concepts of the REA ontology. In section 3 their relation to the service concept is proposed and motivated.

A *resource* was defined as “any object that is of utility and under the control of some enterprise”. Originally, only resources that could be exchanged were considered, such as goods, services and money. Later on, internal resources were taken into account as well, including intangible ones like knowledge.

Resources are modified or exchanged in processes. A *conversion process* uses some input resources to produce new or modify existing resources. For example, water and flour can be used as input economic resources in a baking conversion process to produce the output economic resource bread. An *exchange process* occurs as two agents exchange (external) resources. To acquire a resource an agent has to give up some other resource. For example, in a goods purchase a buying agent has to give up money in order to receive some goods. The amount of money available to the agent is decreased, while the amount of goods is increased.

The constituents of processes are called *economic events*. An economic event is carried out by an agent and affects a resource. In REA, the notion of stockflow is used to specify in what way an economic event affects a resource. REA identifies five stockflows: produce, use, consume, take and give, where the first three occur in conversion processes and the latter two in exchange processes. The stockflows produce and take are positive stockflows in the sense that they increase the value of some resource for an agent – an economic event with a produce stockflow creates or improves some resource in a conversion process while an economic event with a take stockflow transfers a resource to the agent in an exchange process. Similarly, the stockflows use, consume and give are negative stockflows in the sense that they decrease the value of some resource for an agent – an economic event with a use or consume stockflow uses or consumes some resource in a conversion process while an economic event with a give stockflow transfers a resource from the agent in an exchange process. An *agent* is an individual or organization capable of having control over economic resources, and transferring or receiving the control to or from other agents [Ga07].

The *e3value value ontology* [Go00] aims at identifying exchanges of resources between actors in a business case and it also supports profitability analyses of business cases. e3value includes a graphical notation for business models. The basic concepts in e3value are actors, resources, value ports, value interfaces, value activities and value transfers. An *actor* is an economically independent entity. An actor is often, but not necessarily, a legal entity, such as an enterprise or end-consumer or even a software agent. A set of actors can be grouped into a market segment. A resource (also called *value object*) is something that is of economic value for at least one actor, e.g., a car, Internet access, or a stream of music. A *value port* is used by an actor to provide or receive resources to or from other actors. A value port has a direction: in (e.g., receive goods) or out (e.g., make a payment), indicating whether a resource flows in to or out from the actor. A *value interface* consists of in and out ports that

belong to the same actor. Value interfaces are used to model economic reciprocity and value bundling. A *value exchange* represents one or more potential trades of resources between these value ports. A *value activity* is an operation that can be carried out in an economically profitable way for at least one actor.

Both the e3value and the REA ontologies include concepts on the operational level as well as the knowledge level [Fo97], where the operational level models concrete, tangible individuals in a domain, while the knowledge level models information structures that characterize categories of individuals at the operational level. In REA almost all classes on the operational level have a corresponding class on the knowledge level, which is generally not the case for e3value. The REA ontology distinguishes between event type (abstract transfer of categories of resources) and event (actual transfer of tangible concrete resources), both of which correspond to e3value's value transfer.

## 2.2 Service Systems

The notion of service system as proposed recently by Spohrer and colleagues [Sp08] is based on Vargo's Service-Dominant Logic [VL06]. No explicit ontology or modeling technique has been published yet, but we can identify a number of key concepts. A *system* is a configuration of *resources*. Some resources are *operants* that act on other (operand or operant) resources. A *service* is the application of resources to bring about changes that have *value* for another system. Services are performed in the context of *economic exchanges* – the mutual value creation by two or more interacting systems. So value is created in *interactions* between service systems. As a first rough approximation, we could make the following mapping to REA: resource – resource, system – agent, and exchange – exchange process. For service it is not so clear.

For service system interactions, Spohrer proposes the ISPAR model that follows a kind of Conversation for Action protocol for reaching agreement, but draws particular attention to failed interactions as sources of learning. For [Al08], the service interactions (or service encounters as he calls them) are part of a service value chain with a certain division of responsibilities.

## 2.3 Running Example

XYZ Financial Services (XFS) is a fictitious company introduced in [AG08] developing new services for baby boomers. The analysis of XFS has revealed that as these customers advance retirement age, their investment strategies are becoming more risk-averse and they tend to shift their savings from stocks and securities to saving accounts and certificates of deposits. Realizing also that no-interest checking and saving accounts are becoming an important source of revenue, XFS wants to design services that would attract and retain these customers.

## 3 A Unified Service Model

Based on a survey of the literature on services [UM03, Pr04, WS04, VL06, OA06, Sp08, among others] it is possible to identify five salient characteristics of services that apply both to business services and software services.

- A service is an economic resource, since it is an object that is considered valuable by actors and that can be transferred from one actor to another.
- A service is always provided by one actor for the benefit of another actor.
- A service is existence dependent on the processes in which it is produced and consumed, which means that the service exists only when it is consumed and produced. In other words, a service is consumed and produced simultaneously. In contrast to goods and information, a service cannot be stored for later consumption.
- A service encapsulates a set of resources owned by the provider. More precisely, when an actor uses a service in a process, she actually uses the resources encapsulated by the service. When an actor acquires a service in an exchange process, the customer does not get ownership rights on the encapsulated resources, but only use rights.
- A service is always governed by a policy. This means that when a service is used, the resources encapsulated have to be used in compliance with a number of rules formulated in a policy.

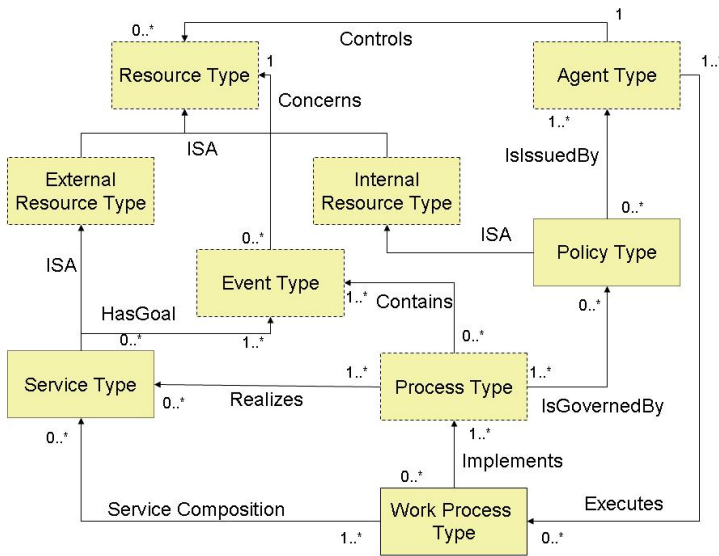


Fig. 1. Basic Service Ontology (core REA concepts in dashed boxes)

### 3.1 Service Ontology

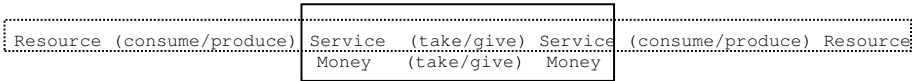
As has been noted REA offers a comprehensive ontology of business concepts, but does not elaborate on the concept of a service. Our ontology (Fig. 1) is based on the core REA constructs Resource, Event, and Agent. All concepts are modeled at the knowledge level [Fo97]. In the following explanations of the concepts the word ‘type’ is sometimes omitted to reduce repetition.

A service is a resource as it is viewed as valuable by some agent and can be exchanged between agents. This is captured in the model by modeling Service Type as a subtype of Resource Type. As such, it automatically inherits all features of external resources, in particular:

- it can be exchanged between agents
- it is realized by a (conversion) process
- it can be used within a (conversion or exchange) process

A distinctive feature of a service is that it has a goal to modify (and hence add value to) other resources which is modelled by the association *hasGoal* from Service to Economic Event. For example, the goal of the hairdressing service is to convert the customer's hair. A service does not specify how it is to be realized, i.e. how its goals are to be achieved. Instead, a service can be realized in many different ways. This is captured in the model by means of the association *Realizes* from Process to Service. To realize a service, the process must achieve at least the goal of the service. To be precise, the event being the goal of the service is contained in the process realizing it.

Services are exchanged like other resources in an exchange process (not included in the Fig. 1) that meets the REA duality principle. This exchange process needs to be distinguished conceptually from the process realizing the service, but they are interwoven in time. The typical sequence of stockflow events is like this:



On the left hand side, we find resources that are consumed or used to produce the service. In the middle, we see the exchange of the service –which is part of an exchange process. On the right hand side, the service is consumed resulting in a change of the customer's resource, which corresponds to the goal event.

A process in the REA sense represents only changes in value of resources as expressed through economic events; it does not address control flow or temporal aspects. In order to represent the latter, a new class Work Process has been introduced. A work process implements a process by executing a number of activities according to some composition (orchestration). So the work process may involve a number of (sub-) services. A work process is executed by an agent.

A policy is a set of assertions intended to govern some behavior. In REA terms, it should be classified as an internal resource that can be created, used and converted but not exchanged. Policies do not apply to a service as such, but to the process in which the service is exchanged or to the process in which the service is produced. An *external* (or public) *policy* constrains the set of events contained in an exchange process. For example, an exchange process involving a loan service should include the exchange of an exemption statement. Note that a contract can be modeled as a special type of external policy. An *internal policy* constrains the set of events in a conversion process that realizes a service (or any other resource, for that matter).

Our service ontology unifies the notion of business service (like hotel rooms, loans and hair dressing) and software service (e.g. web service by means of which hotel reservations can be made). The ontological representation of service is fully in line with SOA as a way of hiding process details to the service customer.

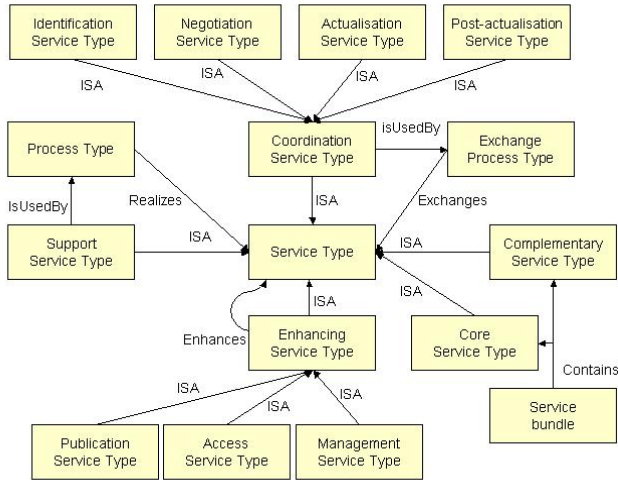


Fig. 2. Service Classification

### 3.2 Service Classification

The starting point of our service classification is the recognition of what we call core services. The reason for viewing these services as core is that they provide the *raison d’être* for an actor in a value network, as they specify what value the actor is able to provide to the network. Core services are easy to identify.

Given a set of core services, there is a need for a number of services that add to or can improve on the core services. We divide these services into four categories: complementary services enhancing services, support services, and coordination services.

#### *Complementary service*

A service complements another service if they are part of the same service bundle and their goals concern the same resource [We07]. For example, a gift-wrapping service complements a book sales service by having as goal to improve the book by packaging it in an attractive way. Thus, both services concern the same resource, the book. A service bundle is defined in our ontology as the services provided to a customer in the same service exchange process.

#### *Enhancing service*

An enhancing service is a service that adds value to another service (rather than to some other kind of resource). The possibility of enhancing services follows from our conceptualization of a service as a resource. The enhancing service has an effect on the quality of another service or some feature like visibility or accessibility. By definition, it is existence-dependent on the other service. On the basis of a review of cases, we have identified the following types of enhancing services:

- **Publication service.** A publication service provides information about another service (or any other resource) e.g. by means of a web page, a TV ad or a public service registry. Hence it produces visibility of the service. At the same time, it increases the knowledge of customers, so it has a dual focus.

- Access service. An access service gives an agent access to another service, i.e. the agent uses the access service to invoke the other service. An advantage of using an access service is that it can act like Facade object in Software Engineering [Ga95] that induces loose coupling by hiding the service details from the consumer. At the same time, it can contain medium-specific logic. Formally, if A is an access service to B, the following must hold: the goal of B is included in the goal of A, and B is a core service.
- Management service. A management service is a service that has as goal to maintain or optimize another service, typically by changing some feature of the work process. The management service can be seen as a service provided by some agent to the owner of the operational service being the customer. The service management includes several tasks that can be delegated to supporting management services, such as monitoring services, controlling services, authorization services, and evaluation services.

#### *Support services*

A service A supports a service B if A has as goal to produce B, or if A has as goal to produce a resource that is used in a process that produces B [Er07].

#### *Coordination services*

A coordination service is any service that supports (ontologically speaking “is used in”) an exchange process. It is used for ensuring that communicating parties in a business relation are coordinated or synchronized. The value object exchanged can be a service but also a good.

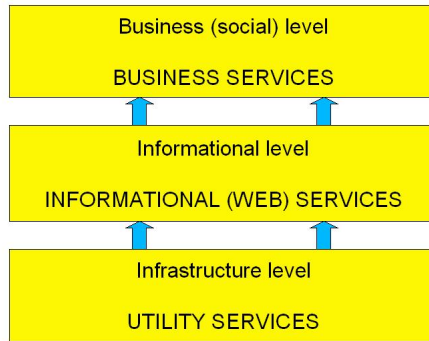
Coordination services can be classified according to the stage in a business relation where the stages are identification, negotiation, actualisation, and post-actualisation [UM03]. For example, a catalogue service is instrumental in the identification stage. In the negotiation stage the terms and conditions of resource deliveries are formed (negotiation service, brokerage service) or reservations are made (reservation service). The actualisation stage is concerned with the actual deliveries of offered resources, including payment (payment service) whereas the post-actualisation stage may include all kinds of in-warranty services.

Coordination services are most elaborate when services are exchanged on a market and usually more simple when the services are exchanged between departments or individuals within an organization.

### **3.3 Service Layer Architecture**

For a Service Layer Architecture that integrates business services and software services, we draw upon the enterprise ontology of Dietz [Di06] that distinguishes a social (performational) level, an informational level and a formational level. To illustrate: an order is a request at the social level, a message at the informational level and a document or file at the formational level.

In the context of IS design, an *informational* service is a software service that has as goal to produce information or enhance communication. A *utility* service is a service that is realized by means of IT hardware and supports informational services by storing, processing or transferring data. A *business service* is an economic service provided by an economic actor to fulfill a customer need. Both informational and utility services can be classified as supporting services (cf. section 3.2).



**Fig. 3.** Service Layer Architecture

Informational services are software services that are characterized by some economic autonomy. They can use other informational services and utility services as resources. Informational services can be supported by management services whose aim is to maintain and improve the quality level of the managed service over time, and other enhancing services.

Utility services [Er07] are software services at the infrastructure level that are characterized by a certain economic autonomy (which makes them suitable candidates for outsourcing) and usually support more than one informational service. The value provided by the utility service is the storage of data or the execution of programs. Also at this level we can have managing and other enhancing services.

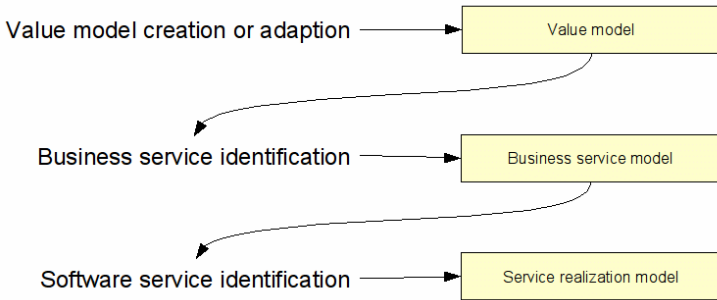
Informational services support the business level in different ways, depending on their focus:

- Decision support. Some business services are knowledge intensive. Take for example a credit-check service at the business level. Assessing the credit level of a customer is a responsible task assigned to some business actor. However, the service can make use of a web service that generates a credit rating on the basis of a database and/or business intelligence function. In this case, the web service at the informational level is a resource used by the credit-check process at the business level. It can even replace the credit-check if the service is completely delegated to the software. A decision service may be seen as the application of a set of business rules (that make up a policy) to generate a statement.
- Process support. Work processes at the business level can be supported by an informational service that takes care of the orchestration. Such services correspond to what [Er07] calls task-oriented services and what [Pa05] calls business process service.
- Information management support (for any kind of business service). Here we talk about what [Er07] calls entity-oriented information services that maintain and provide information about entities (business objects) and typically have a CRUD-style web service interface. For example, a web service that creates hotel reservations on request and stores them in the database is an information management service supporting a coordinating (reservation) service at the business level.

## 4 Service Identification – A Method Proposal

The service models introduced in section 3 enable the designer to start designing or identifying services in the business domain, and use this as input for the design of web services in the information system domain. As argued in [HJ07], this means that the web service designer does not need to start from scratch. The advantage of using value modeling is that it is already supported by established modeling techniques and methodologies. In section 4.1 we introduce a value-based service design method. In 4.2, this method is applied to the running example.

The first step in the method consists of creating a value model, which represents resource exchanges and conversions. However, in order to realize a value network as described by an e3value model, there is a need for a number of additional resources not explicitly visible in the e3value model. These resources are services required for managing the exchanges as well as the conversions of resources. Thus, the main purpose of the proposed method is to assist a designer in identifying these services in a systematic way.



**Fig. 4.** Service design method schematic overview

### 4.1 A Service Design Method

Given the service ontology and architecture of section 3, a service design method can be developed that bridges the business and software level. The following description is not a complete cookbook, but is intended just to show how this bridging can be achieved.

#### Step 1: Value model creation or adaptation

In the value modeling step, we model the business activities from an economic perspective – in terms of value creation. In the current age of global networks, the focus cannot limit itself to a single actor only but should be on modeling value constellations of business parties. When modeling we do, however, take the perspective of one of the actors of the value network, to be called the focal actor, and identify services needed by that actor. We propose to use e3value modeling as introduced in section 2. Services do occur in the value model as a particular kind of value objects exchanged. The value model distinguishes service bundles and can identify complementary services. The value model also identifies supporting services needed to realize the core services.



The result of this step is a value model of the network that does abstract from IT services and from processes. The advantage is two-fold: first, this model is a relatively stable reference point when processes and particular service implementations evolve over time. Secondly, the model allows addressing business evolution at the appropriate level. For example, the reconfiguration of services for technical or logistic reasons is something that should be addressed at the information level. However, the replacement of one partner in the network by another one is a decision that needs to consider the strategic impacts.

The value model does not only contain core services, but can also represent quality features relating to these services, such as “convenience” or “low-budget” (so-called second-order values [We06]). They have an impact on how the service is provided, and hence may also influence the software service design.

### **Step 2: Business Service Identification**

In the second step, the value model is used as a basis for identifying more business services in addition to the core services, complementary services and supporting services identified in step 1, in particular:

- enhancing services (access, information and management)
- coordination services that support the resource exchanges

The identification of business services includes the specification of business rules and policies governing the services (i.e., its exchange and conversion processes). A *table* notation can be used with four columns: the core service, its enhancing services, its coordination services and the applicable policies.

### **Step 3: Software Service Identification**

The goal of this step is to identify services at the informational level and infrastructure level. A top-down approach can start from the (business) services identified in phase 2. Alternatively, a meet-in-the-middle approach can start from the available applications and identify services that embed these applications while at the same time supporting a business service. It is noteworthy that the informational services are exchanged between the IS domain and the business domain, usually represented by different departments each having their own responsibilities and autonomy, and so the service identification itself is very much a negotiation process.

The informational services are not logically derivable from the e3value model, but for various classes of business services (e.g. a reservation service, or a payment service) generic solutions can be applied. Such an approach can be seen as a concrete implementation of the suggestion made by [PH06] that service design should draw on business standards whenever possible.

Process support provides service orchestration. As work processes are governed by business rules/policies, it is recommendable to design decision services that incorporate these rules and can be used by the service orchestration.

Informational services can be further supported by management services whose aim is to maintain and improve the quality level of the managed service over time. Note that these are software services themselves.

Finally, the infrastructural support for the informational services has to be designed, including their sourcing and required SLA level.

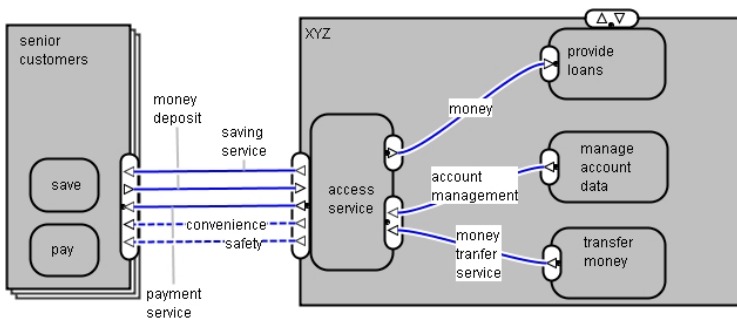
## 4.2 Example: XYZ Financial Services

As described in section 2, XFS decided to target a particular market segment, the more risk-averse conservative senior citizens. A value analysis makes clear that there are opportunities for co-creation of value: for instance, certificates of deposit (CD) provide long-term funds to XFS, while at the same time offering the customer a safe place for their long-term savings. For the purpose of this example and because the data are limited, we have kept the value model very simple. One feature that we want to draw attention to is the second-order values “convenience” and “safety”. To reach the target group, these can be an important competitive feature that should also be taken into account when considering the way the services are provided.

*Step 2. Identification of business services.* Evidently, there are two core services offered by XYZ to its customers: account management and money transfer (we ignore here the loan service that explores the money provided by this customer group but offered to another group). For lack of space, we do not elaborate on the enhancing services and lump them all together in a single access service.

*Step 3. Software services.* The core services identified in step 2 lend themselves to automation support. For transfer of money, XYZ needs to involve an inter-bank payment exchange service, and to access this external service, an access service must be added. The customer access service can also be supported by informational services. Examples are: product information service (that provides information about the product to potential customers), product contracting service (that allows a customer to open an account), a payment access service, and an account management service.

The payment access service is to be distinguished from the payment core service. The latter incorporates the basic functions of money transfer. The payment access service provides an interface to this service. Since convenience is a strategic second-order value, it is important to offer one or more user-friendly interfaces. For example, a pay-by-phone interface and a web interface. The distinction between payment access service and payment core service is apparently not made in SOMA. However, it is very useful way of improving business agility.



**Fig. 5.** Initial value Model for XYZ Case, business level only

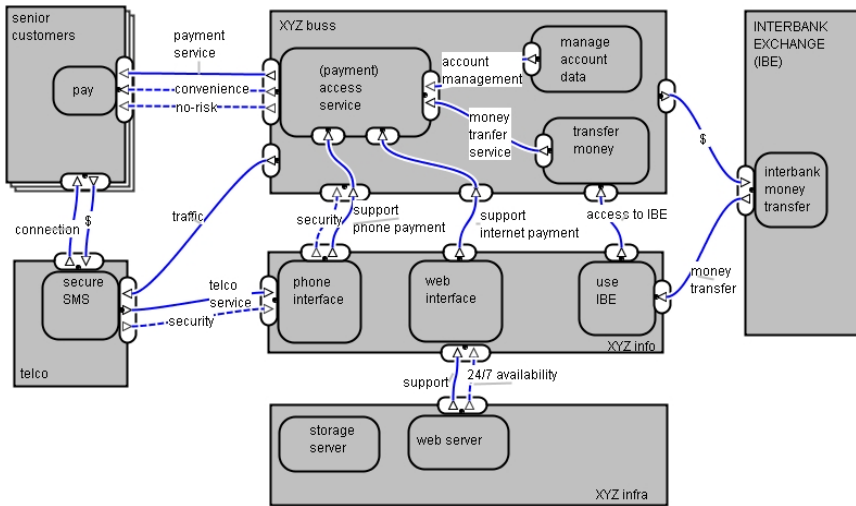


Fig. 6. Three-layer service model (for payment service only)

Information support services can be identified for the business objects involved, such as Customer, with a CRUD-style interface (create customer, update customer, delete customer, update address, etc).

In the case of a meet-in-the-middle approach, the informational services identified so far should be confronted with the existing legacy applications.

**Some remarks**

- when comparing our value-based model results with those from SOMA – however sketchy both are – one difference is that e3value tends to draw attention to value networks. In the XFS case, it is clear that XFS cannot deliver the proposed services on its own. In particular, a pay-by-SMS service is the result of a co-creation of value that benefits all parties involved in some way. In contrast, SOMA seems to restrict itself to the internal software services in the company.
- an advantage of the value-based method is that it not only identifies services, but can also record second-order values, corresponding to extra-functional properties such as security or availability that should be considered in SLA agreements.
- the value-based method clarifies the dual focus of informational services, such as an XFS web interface. It provides support to the business (the coordination services). At the same time, it is a kind of access service that serves the customer in getting to the payment service. The effect of this dual focus is that it has at least two goals: satisfy the customer (who interacts with the service) and satisfy the business (that has set up and maintains the service).

**5 Conclusion**

In this paper, we have developed a unified model of services. On the basis of the service ontology, we have proposed a service design method that starts from a value

model and helps to identify core and enhancing services. Subsequently, it can help identifying possible web services. Whereas most SOA design methods consider service design as a software engineering problem, we consider it as both a business engineering and software engineering problem.

Topics for future research include validation by means of more cases, as well as the development or reuse of graphical notations supporting the service design. The graphical notation of e3value is useful but has problems with rendering larger models.

## References

- [An06] Andersson, B., Bergholtz, M., Edirisuriya, A., Ilayperuma, T., Johannesson, P., Gordijn, J., Grégoire, B., Schmitt, M., Dubois, E., Abels, S., Hahn, A., Wangler, B., Weigand, H.: Towards a Reference Ontology for Business Models. In: Embley, D.W., Olivé, A., Ram, S. (eds.) ER 2006. LNCS, vol. 4215, pp. 482–496. Springer, Heidelberg (2006)
- [Al08] Alter, S.: Service system fundamentals: Work system, value chain, and life cycle. *IBM Systems Journal* 47(1), 71–86 (2008)
- [AG08] Arsanjani, A., et al.: SOMA: A method for developing service-oriented solutions. *IBM Systems Journal* 47(3), 377–396 (2008)
- [BP07] OASIS Web Services Business Process Execution Language Version 2.0 (2007), <http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html>
- [Di06] Dietz, J.: *Enterprise Ontology - Theory and Methodology*. Springer, Berlin (2006)
- [Er07] Erl, T.: *Soa: principles of service design*. Prentice-Hall, Englewood Cliffs (2007)
- [Fo97] Fowler, M.: *Analysis Patterns. Reusable Object Models*. Addison-Wesley, Reading (1997)
- [Ga07] Gailly, F., Poels, G.: Towards Ontology-driven Information Systems: Redesign and Formalization of the REA Ontology. Working paper, Univ. Ghent (2008-03-27), [http://www.FEB.Ugent.be/fac/research/WP/Papers/wp\\_07\\_445.pdf](http://www.FEB.Ugent.be/fac/research/WP/Papers/wp_07_445.pdf)
- [Ga95] Gamma, E., et al.: *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, Reading (1995)
- [Ge99] Geerts, G., McCarthy, W.E.: An Accounting Object Infrastructure For Knowledge-Based Enterprise Models. *IEEE Int. Systems & Their Applications*, 89–94 (1999)
- [Go00] Gordijn, J., Akkermans, J.M., van Vliet, J.C.: Business modeling is not process modeling. In: Mayr, H.C., Liddle, S.W., Thalheim, B. (eds.) ER Workshops 2000. LNCS, vol. 1921. Springer, Heidelberg (2000)
- [HJ07] Henkel, M., Johannesson, P., Perjons, E., Zdravkovic, J.: Value and Goal Driven Design of E-Services. In: Proc. of the IEEE Int. Conference on E-Business Engineering (Icebe 2007). IEEE Computer Society, Washington (2007)
- [Hr06] Hruby, P.: *Model-Driven Design of Software Applications with Business Patterns*. Springer, Heidelberg (2006)
- [Mc82] McCarthy, W.E.: The REA Accounting Model: A Generalized Framework for Accounting Systems in a Shared Data Environment. *The Accounting Review* (1982)
- [NL07] Nayak, N., Linehan, M., et al.: Core business architecture for a service-oriented enterprise. *IBM Systems Journal* 46(4), 723–742 (2007)
- [NR93] Norman, R., Ramirez, R.: From value chain to value constellation: Designing interactive strategy. *Harvard Business Review*, 65–77 (July-August 1993)
- [OA06] OASIS. Reference Model for Service Oriented Architecture 1.0 (2006), <http://www.oasis-open.org/committees/download.php/19679/soa-rm-cs.pdf>

- [Os04] Osterwalder, A.: The Business Model Ontology, Ph.D. thesis, HEC Lausanne (2004), <http://www.hec.unil.ch/aosterwa/> (last accessed, 007-07-01)
- [Pa05] Papazoglou, M.: Web Services Technologies and Standards. ACM Computing Surveys (2005)
- [PH06] Papazoglou, M., van den Heuvel, W.J.: Service-oriented design and development methodology. *Int. Journal of Web Engineering and Technology* 2(4), 412–442 (2006)
- [PK08] Prahalad, C.K., Krishnan, M.S.: The New Age of Innovation: Driving Cocreated Value Through Global Networks (2008)
- [Pr04] Preist, C.: A Conceptual Architecture for Semantic Web Services. In: McIlraith, S.A., Plexousakis, D., van Harmelen, F. (eds.) ISWC 2004. LNCS, vol. 3298, pp. 395–409. Springer, Heidelberg (2004)
- [Sp08] Spohrer, J., et al.: The Service System Is the Basic Abstraction of Service Science. In: Proc. HICSS (2008)
- [UM03] UN/CEFACT Modelling Methodology (UMM) User Guide (2003), [http://www.unece.org/cefact/umm/UMM\\_userguide\\_220606.pdf](http://www.unece.org/cefact/umm/UMM_userguide_220606.pdf) 2008-02-19
- [UN08] United Nations, Dept. of Economic and Social Affairs. Common DataBase (CDB) Data Dictionary (February 19, 2008), <http://unstats.un.org/unsd/cdbmeta/gesform.asp?getitem=398>
- [Us96] Uschold, M., Gruninger, M.: Ontologies: Principles, Methods, and Applications. *Knowledge Engineering Review* 11(2), 93–155 (1996)
- [VL06] Vargo, S.L., Lusch, R.F., Morgan, F.W.: Historical Perspectives on Service-Dominant Logic. In: Lusch, R.F., Vargo, S.L., Sharpe, M.E. (eds.) *The Service-Dominant Logic of Marketing*, Armonk, NY, pp. 29–42 (2006)
- [We06] Weigand, H., Johannesson, P., Andersson, B., Bergholtz, M., Edirisuriya, A., Ilayperuma, T.: On the Notion of Value Object. In: Dubois, E., Pohl, K. (eds.) CAiSE 2006. LNCS, vol. 4001, pp. 321–335. Springer, Heidelberg (2006)
- [We07] Weigand, H., et al.: Strategic Analysis Using Value Modeling—The c3-Value Approach. In: HICSS 2007, p. 175 (2007)
- [WS04] W3C. Web Services Architecture W3C Working Group (2004), <http://www.w3.org>
- [Zi04] Zimmerman, O., Krogdahl, P., Gee, C.: Elements of Service-Oriented Analysis and Design (2004), <http://www-128.ibm.com/developerworks/library/ws-soad1/>

# Data-Flow Anti-patterns: Discovering Data-Flow Errors in Workflows

Nikola Trčka, Wil M.P. van der Aalst, and Natalia Sidorova

Department of Mathematics and Computer Science  
Eindhoven University of Technology  
P.O. Box 513, 5600 MB Eindhoven, The Netherlands  
{n.trcka,w.m.p.v.d.aalst,n.sidorova}@tue.nl

**Abstract.** Despite the abundance of analysis techniques to discover control-flow errors in workflow designs, there is hardly any support for data-flow verification. Most techniques simply abstract from data, while data dependencies can be the source of all kinds of errors. This paper focuses on the discovery of data-flow errors in workflows. We present an analysis approach that uses so-called “anti-patterns” expressed in terms of a temporal logic. Typical errors include accessing a data element that is not yet available or updating a data element while it may be read in a parallel branch. Since the anti-patterns are expressed in terms of temporal logic, the well-known, stable, adaptable, and effective model-checking techniques can be used to discover data-flow errors. Moreover, our approach enables a seamless integration of control flow and data-flow verification.

## 1 Introduction

A *Process-Aware Information System* (PAIS) is a software system that manages and executes operational processes involving people, applications, and/or information sources on the basis of process models [6]. Examples of PAISs are workflow management systems, case-handling systems, enterprise information systems, etc. Many of these systems are driven by explicit process models, i.e., based on a process model a system is configured that supports the modeled process. In this paper, we primarily focus on the analysis of the models used to configure workflow management systems [2,9,11,21]. However, our approach is also applicable to other PAISs.

In the last 15 years, many analysis techniques have been developed to analyse workflow models [2]. Most of these techniques focus on verification, i.e., on the discovery of design errors. Although many process representations have been used or proposed, most researchers are using Petri nets as a basic model [1,20]. The flow-oriented nature of workflow processes makes the Petri net formalism a natural candidate for the modeling and analysis of workflows. Most workflow management systems provide a graphical language that is close to Petri nets, or that has a token-based semantics making a (partial) mapping to Petri nets relatively straightforward. Industrial languages like Business Process Modeling

Notation (BPMN), extended Event-driven Process Chains (eEPCs) and UML activity diagrams, are examples of languages that can be translated to Petri nets.

Unfortunately, lion’s share of attention has only been devoted to control flow while other perspectives such as data flow and resource allocation have been completely ignored. Existing analysis techniques typically check for errors such as deadlocks, livelocks, etc. while abstracting from data and resources. In most cases the abstraction from resource information is unavoidable, as resources are often external and dynamic in nature. The role of data in the workflow is however important: Routing choices in a workflow are typically based on data, which makes the control flow data dependent. Moreover, the data flow can be erroneous itself. Another limitation of the most of the existing workflow verification approaches is the way they communicate to the user: they are not configurable, and it is not always clear what types of errors they capture (the details are typically hidden in the verification algorithms).

To address some of the limitations of existing approaches, we propose a new analysis framework based on (a) workflow nets with data, (b) temporal logic, and (c) “anti-patterns”. A *Workflow net with Data* (WFD-net) is a special type of a Petri net, with a clear start and end point and annotations related to the handling of data (a task can *read*, *write*, or *destroy* a particular data element). Assuming a WFD-net representation, we define several *anti-patterns* related to the data flow. The term “anti-patterns” was coined in 1995 by Andrew Koenig [12]. He stated that “An anti-pattern is just like pattern, except that instead of solution it gives something that looks superficially like a solution, but isn’t one” [12]. The goal of anti-patterns is to formally describe repeated mistakes such that they can be recognized and repaired. In this paper, we use the temporal logic CTL\* (and its subclasses CTL and LTL) to formalize our anti-patterns. This formalization can be used to discover the occurrence of such anti-patterns in WFD-nets by standard model-checking techniques [4]. Although not elaborated on in this paper, the same techniques can be used to define correctness notions related to the control flow and check these in an integral way.

An example of an anti-pattern is *DAP 1: Missing data*. This anti-pattern describes the situation where some data element needs to be accessed, i.e. read or destroyed, but either it has never been created or it has been deleted without having been created again. This property can be expressed in both CTL and LTL. Hence, given a WFD-net it can be easily checked using standard model checkers.

The remainder of this paper is organized as follows. Section 2 presents related work. Section 3 introduces WFD-nets. This representation is used in Section 4 to define a comprehensive set of data-flow anti-patterns. The formalization of these anti-patterns is given in Section 5. Section 6 concludes the paper.

## 2 Related Work

Since the mid-nineties, many researchers have been working on workflow verification techniques [11,16]. It is impossible to give a complete overview of the

related work here (see [3] for references). Therefore, we only mention the work directly relevant to this paper, namely verification approaches in which control and data flow are both taken into account for verification.

The importance of data-flow verification in workflow processes was first mentioned in [15]. There, several possible errors in the data flow are identified, like, e.g., the missing and the redundant data error, but no means for checking these errors is provided. Later, [18] conceptualized the errors from [15] using UML diagrams, and gave supporting verification algorithms. This work was further extended and generalized in [19]. None of these approaches consider data removal. The exact details of the erroneous scenarios are not always clear, being hidden in the verification algorithms, and good diagnostics are missing. Moreover, the methods are not adaptive enough, as new properties cannot be easily added to the checks.

In [8], a model called *dual workflow nets* is proposed, that can describe both the data flow and the control flow. The notion of classical soundness from [1] is extended to support the case when data flow can influence control flow. No explicit data correctness properties are considered.

The ADEPT<sub>flex</sub> tool [14] supports a limited set of checks for data-flow correctness. The focus is entirely on dynamic changes in workflow models.

The work closest to ours is [7]. There, model checking is used to verify business workflows, from both the control- and data-flow perspective. The underlying workflow language is UML diagrams as opposed to the Petri net approach taken in this paper. Only a few data correctness properties are identified and no systematic classification is presented. Data can only be read or written, but not destroyed. Finally, [7] only considers LTL model-checking while several of our anti-patterns are not expressible in LTL.

In the field of software verification model checking have been successfully used to discover program bugs that are caused by, e.g., non-initialized or dead variables [17]. In this, totally different, application domain, concurrency issues are rarely treated and systematic classification of errors is missing.

### 3 Workflow Nets with Data

Workflow nets (WF-nets) are commonly used as a basic representation for workflow processes [1]. A WF-net is a Petri net with one unique source place and one unique sink place such that all nodes are on a path from the source place to the sink place. The transitions in a WF-net represent tasks. A WF-net is instantiated for a particular case by putting a token on the source place. The completion of this instance is denoted by a token on the sink place. WFD-nets extend WF-nets with data elements and define four relationships between tasks and these data elements. First, a task may *read* a particular data element. This data element is thus expected to have a value before the task is executed. Second, a task may *write* (to) a particular data element. This means that this data element gets a new value. If it did not have a value yet, it is created; otherwise it is overwritten. Third, a task may *destroy* a data element, leaving it with no value. Finally, a task may use a particular data element in its *guard* (optional).



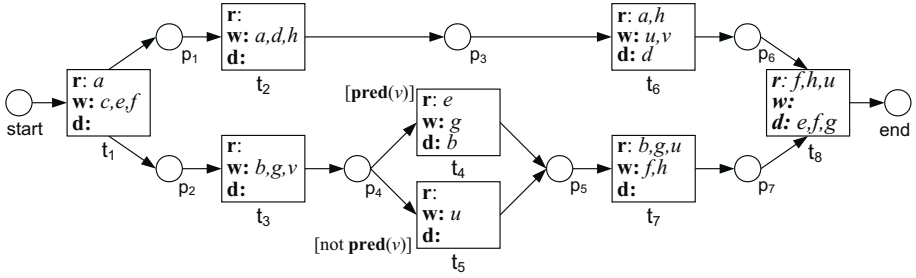


Fig. 1. A WFD-net

We only consider data elements that are case-related, i.e., that belong to an individual process instance and cannot be shared among different cases and/or different processes. The techniques of this paper, however, can be applied to support complete data interplay, if all processes are modeled and combined into one (huge) WFD-net. In addition, we assume workflows to start from an empty data state; starting with some existing data can easily be modeled with an artificial start task.

The following definition introduces *Workflow nets with Data* (WFD-nets).

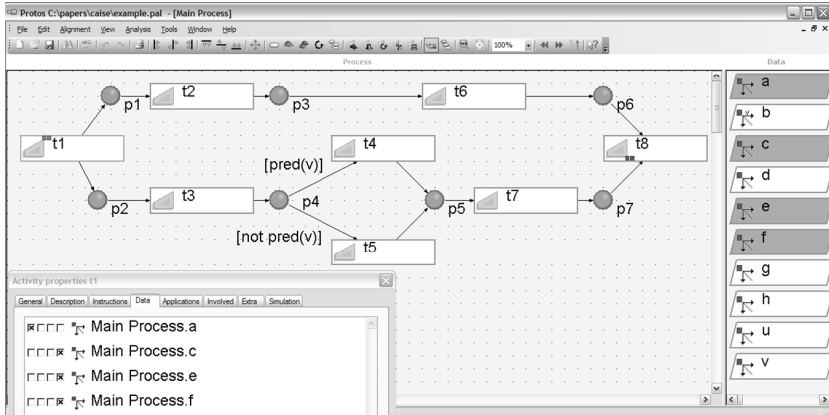
**Definition 1 (WFD-net).** A tuple  $\langle P, T, F, \mathcal{D}, \mathcal{G}_{\mathcal{D}}, \text{Read}, \text{Write}, \text{Destroy}, \text{Guard} \rangle$  is a Workflow net with data (a WFD-net) iff:

- $\langle P, T, F \rangle$  is a WF-net, with places  $P$ , transitions  $T$  and arcs  $F$ ;
- $\mathcal{D}$  is a set of data elements;
- $\mathcal{G}_{\mathcal{D}}$  is a set of guards over  $\mathcal{D}$ ;
- $\text{Read} : T \rightarrow 2^{\mathcal{D}}$  is the reading data labeling function;
- $\text{Write} : T \rightarrow 2^{\mathcal{D}}$  is the writing data labeling function;
- $\text{Destroy} : T \rightarrow 2^{\mathcal{D}}$  is the destroying data labeling function; and
- $\text{Guard} : T \rightarrow \mathcal{G}_{\mathcal{D}}$  is the guarding function, assigning guards to transitions.  $\square$

Note that a WFD-net is just an annotated WF-net; its formal semantics will be given in Section 5 using the concept of unfolding to a WF-net.

Fig. 1 shows an example of a WFD-net. There are 10 data elements ( $a, \dots, h, u$ , and  $v$ ), and these elements are linked to tasks in the process. Task  $t_6$ , e.g., reads from data elements  $a$  and  $h$ , writes to  $u$  and  $v$ , and destroys  $d$ . Thus:  $\text{Read}(t_6) = \{a, h\}$ ,  $\text{Write}(t_6) = \{u, v\}$ ,  $\text{Destroy}(t_6) = \{d\}$ , and  $\text{Guard}(t_6) = \text{true}$  (i.e., no guard). If one ignores the read, write, destroy, and guard annotations, Fig. 1 is a WF-net with source place  $\text{start}$  and sink place  $\text{end}$ . All cases start with task  $t_1$  and end with task  $t_8$ . In-between,  $t_2$  and  $t_6$  are executed in sequence and this is done in parallel with the lower process fragment that starts with  $t_3$  and ends with  $t_7$ . In-between  $t_3$  and  $t_7$  either  $t_4$  or  $t_5$  is executed. This choice depends on the evaluation of  $\text{pred}(v)$ ; if this predicate evaluates to true,  $t_4$  is selected, otherwise  $t_5$ .

WFD-nets can be seen as an abstraction from notations deployed by popular modeling tools. To illustrate this we show in Fig. 2 the Protos model corresponding to the WFD-net from Fig. 1. Protos (Pallas Athena) uses a Petri-net-based



**Fig. 2.** A Protos model showing both the control flow and data flow

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. Like most other tools it allows for the modeling of both control flow and data flow. The left-hand side of Fig. 2 shows the control flow while the right-hand side shows the different data elements. The colors (different shades of grey in this case) show the relationships between  $t_1$  and these data elements, and the bottom window of Fig. 2 shows the nature of these relationships.

As illustrated by Fig. 2, the language used by Protos is close to Definition 1. Other popular notations such as the Business Process Modeling Notation (BPMN), extended Event-driven Process Chains (eEPCs), UML activity diagrams, etc. also allow for the modeling of both control flow and data flow. In fact, the basic idea to link data elements to tasks originates from IBM's Business Systems Planning (BSP) methodology developed in the early eighties. Here a so-called *CRUD matrix* is used showing Create, Read, Update, and Delete relationships between tasks and data elements. The Read relationship in a CRUD matrix is similar to the Read function and the Delete relationship is similar to the Destroy function in Definition 1. The Update relationship is similar to the Write function, but may also refer to a combination of read and write. The Create relationship can be seen as the first write action for a data element. In Protos a variant of the CRUD matrix is used and the basic relations are Mandatory, Created, Deleted, and Modified. Other tools use other variants. However, all of these operations can be translated into the primitives given in Definition 1. Hence, the applicability of the results presented in the remainder extends to other notations (BPMN, eEPCs, etc.) and variants of the CRUD matrix.

Soundness 1 is the mostly used correctness notion for workflows. The basic idea is that the process cannot deadlock or livelock and it is always still possible to terminate properly. However, the classical soundness notions do not consider data. This is serious limitation. For example, the workflow design shown in Fig. 1

is sound but has some serious design flaws when considering the data annotations. For example, data element  $b$  may be destroyed in task  $t_4$  while it is needed in the following task  $t_7$  for reading. To identify such problems we use so-called *data-flow anti-patterns*.

## 4 Data-Flow Anti-Patterns

In this section we introduce data-flow anti-patterns and explain them using the example WFD-net shown in Fig. 1. For the sake of readability, when saying “data element  $d$  is read” in the descriptions of anti-patterns, we actually mean “data element  $d$  is read or used for the evaluation of a guard”. Evaluating predicate  $pred$  on data element  $v$  in Fig. 1 thus will be interpreted as reading  $v$  by transitions  $t_4$  and  $t_5$ .

**DAP 1 (Missing Data).** *This anti-pattern describes the situation where some data element needs to be accessed, i.e. read or destroyed, but either it has never been created or it has been deleted without having been created again.*

In Fig. 1, data elements  $a$  and  $b$  are missing. Note that  $a$  needs to be read immediately by the first task, although it has not been created yet. Data element  $b$  is created by  $t_3$ , but it can be destroyed by  $t_4$  before it reaches  $t_7$  that needs to read it.

Unlike some other anti-patterns we will present later, we do not introduce a strong and a weak variant for missing data depending on the fact whether we will certainly miss a data element, or we miss it only at some execution paths that might be chosen. We require that data should be present independently of the choices made in the workflow—the absence of data necessary for an action indicates a flaw in the workflow.

**DAP 2 (Strongly Redundant Data).** *A data element is strongly redundant if there is a writing activity after which in all possible continuations of the execution this data element is never read before it gets destroyed or the workflow execution is completed.*

In Fig. 1, data elements  $c$  and  $d$  are strongly redundant. Task  $t_1$  creates  $c$  but it is never read in the workflow, while task  $t_2$  creates  $d$  and  $t_6$  destroys  $d$  without reading it.

**DAP 3 (Weakly Redundant Data).** *A data element is weakly redundant if there is some execution scenario in which it is written but never read afterwards, i.e. before it is destroyed or the workflow execution is completed.*

If a data element is strongly redundant (DAP 2), it is also weakly redundant (DAP 3), while the opposite does not hold in general. Consider data element  $e$  in Fig. 1. It is created by  $t_1$  and it is only read by  $t_4$ . In case  $t_5$  and not  $t_4$  is chosen,  $e$  remains unread, and hence it is weakly redundant. On the other hand,

if  $t_5$  is chosen,  $e$  is read between its creation and destruction, and therefore  $e$  is not strongly redundant.

Strongly redundant data indicates in most situations a real flaw in the workflow. Weakly redundant data can in principle refer not to a flaw but to a design decision aimed e.g. at the uniformization/simplification of data requests (asking all clients to provide data  $d_1, \dots, d_k$ , while  $d_k$  is of interest only for the clients with a particular value of  $d_1$ ) or at the improvement of the performance (computing some weakly redundant data element  $d$  in parallel to some other activity whose result will make it clear whether  $d$  is needed afterwards or not; in case  $d$  is needed, it is immediately available, and it is ignored otherwise).

**DAP 4 (Strongly Lost Data).** *A data element is strongly lost if there is a writing activity after which in all possible continuations of the execution this element gets overwritten without having been read first.*

In Fig. 11, element  $f$  is strongly lost, since  $t_1$  writes to  $f$ ,  $t_7$  rewrites it, and  $f$  cannot be read in between.

**DAP 5 (Weakly Lost Data).** *A data element is weakly lost if there is an execution sequence in which it is overwritten without been read first.*

Strongly lost data (DAP 4) implies weakly lost data (DAP 5) but, in general, not the other way around. In Fig. 11,  $g$  and  $h$  are weakly lost. Task  $t_3$  writes to  $g$ , then  $g$  may be overwritten by  $t_4$ , after which  $g$  is read by  $t_7$ . In case  $t_5$  is chosen instead of  $t_4$ ,  $g$  is read by  $t_7$  without having been overwritten. The example of  $h$  shows a concurrency-related instance of this anti-pattern. In case of the execution sequence  $t_1t_2t_6t_3t_4t_7t_8$ ,  $t_2$  writes to  $h$ ,  $t_6$  reads it,  $t_7$  writes again and  $t_8$  reads  $h$ . If  $t_6$  is scheduled to be executed later, and the execution sequence is  $t_1t_2t_3t_4t_7t_6t_8$ ,  $t_2$  writes to  $h$ , then  $t_7$  rewrites it, and only then  $h$  is read. Note that in the latter case both  $t_6$  and  $t_8$  use the value of  $h$  produced by  $t_7$ .

Strongly lost data normally indicates a real flaw in the workflow, while weakly lost data may be a design decision, but may also be a flaw. Examples where weakly lost data may be an instance of a normal behavior are, e.g., reading some client's data (address, telephone number, etc.), which might remain possible along the whole workflow. The fact that they are updated (overwritten) without ever having been read is then a normal scenario.

**DAP 6 (Inconsistent Data).** *Data is inconsistent if a task is using this data while some other task (or another instance of the same task) is writing to this data or is destroying it in parallel.*

In Fig. 11,  $u$  is inconsistent since  $t_5$  and  $t_6$  may write to  $u$  in parallel and it is not clear which version of  $u$  will be used by  $t_7$  and  $t_8$ . Data element  $v$  is also inconsistent, as  $t_6$  might change its value before or after the predicate **pred** is evaluated. Inconsistent data normally indicates a real flaw in the workflow.

The following anti-patterns are related to data removal. They should be seen more as efficiency drawbacks rather than strict correctness problems. These anti-patterns are especially important for scientific workflows, where data is often

large and its unnecessary storage should be avoided, while automatic garbage collection is rare.

**DAP 7 (Never destroyed).** *A data element is never destroyed if there is a scenario in which it is created but not destroyed afterwards.*

For example,  $a$  is never destroyed after its creation by  $t_2$ , which indicates the possibility of leaving garbage by the workflow.

**DAP 8 (Twice Destroyed).** *A data element is twice destroyed if there is a scenario in which it is destroyed twice in a row without having been created in between.*

This anti-pattern is similar to the strongly lost data error but concerns data deletion. It can be seen as a special instance of DAP [II](#).

**DAP 9 (Not Deleted on Time).** *A data element is not deleted on time when there is a task that reads it without destroying it, and after this task the data element is never read again in the workflow, independently of the choices made.*

For example,  $t_7$  is the last (and the only) task reading  $g$ , but  $g$  is deleted later, by  $t_8$ . Thus  $g$  is not deleted on time.

## 5 Formalization and Implementation

After introducing the anti-patterns in an informal manner, we now show that these patterns can be formalized and supported by standard model checking tools. We first introduce CTL\* and its subclasses LTL and CTL. Then we provide a translation of WFD-nets to Kripke structures to facilitate the verification of the desired temporal properties, and we provide formalizations for the anti-patterns formulated in Section [4](#). Finally, we discuss how the approach can be supported by existing tools.

### 5.1 Temporal Logic CTL\*

CTL\* [4](#) is a powerful (state-based) temporal logic combining linear time and branching time modalities. It is typically defined on Kripke structures, so we introduce this model first.

**Definition 2.** *A Kripke structure is a tuple  $(S, A, \mathcal{L}, \rightarrow)$  where  $S$  is a finite set of states,  $A$  is a non-empty set of atomic propositions,  $\mathcal{L} : S \rightarrow 2^A$  is a (state) labeling function, and  $\rightarrow \subseteq S \times S$  is a transition relation.  $\square$*

If  $(s, s') \in \rightarrow$ , then there is a *step* from  $s$  to  $s'$ , then also written as  $s \rightarrow s'$ . For a state  $s$ ,  $\mathcal{L}(s)$  is the set of atomic propositions that *hold* in  $s$ .

A *path* from  $s$  is an infinite sequence of states  $s_0, s_1, s_2, \dots$  such that  $s = s_0$ , and either  $s_k \rightarrow s_{k+1}$  for all  $k \in \mathbb{N}$ , or there exists an  $n \geq 0$ , such that  $s_k \rightarrow s_{k+1}$  for all  $0 \leq k < n$ ,  $s_n \not\rightarrow$ , and  $s_k = s_{k+1}$  for all  $k \geq n$ . For a path  $\pi = s_0, s_1, s_2, \dots$  and some  $k \geq 0$ ,  $\pi^k$  denotes the path  $s_k, s_{k+1}, s_{k+2}, \dots$ .

We now define the syntax of CTL\*.

**Definition 3.** *The classes  $\Phi$  of CTL\* state formulas and  $\Psi$  of CTL\* path formulas are generated by the following grammar:*

$$\begin{aligned} \phi &::= a \mid \neg\phi \mid \phi \wedge \phi \mid E\psi \\ \psi &::= \phi \mid \neg\psi \mid \psi \wedge \psi \mid X\psi \mid \psi \cup \psi \end{aligned}$$

with  $a \in A$ ,  $\phi \in \Phi$ , and  $\psi \in \Psi$ . □

Validity of CTL\* formulas is defined as follows.

**Definition 4.** *We define when a CTL\* state formula  $\phi$  is valid in a state  $s$  (notation:  $s \models \phi$ ) and when a CTL\* path formula  $\psi$  is valid on a path  $\pi$  (notation:  $\pi \models \psi$ ) by simultaneous induction as follows:*

- $s \models a$  iff  $a \in \mathcal{L}(s)$ ;
- $s \models \neg\phi$  iff  $s \not\models \phi$ ;
- $s \models \phi_1 \wedge \phi_2$  iff  $s \models \phi_1$  and  $s \models \phi_2$ ;
- $s \models E\psi$  iff there exists a path  $\pi$  from  $s$  such that  $\pi \models \psi$ ;
- $\pi \models \phi$  iff  $s$  is the first state of  $\pi$  and  $s \models \phi$ ;
- $\pi \models \neg\psi$  iff  $\pi \not\models \psi$ ;
- $\pi \models \psi_1 \wedge \psi_2$  iff  $\pi \models \psi_1$  and  $\pi \models \psi_2$ ;
- $\pi \models X\psi$  iff  $\pi^1 \models \psi$ ; and
- $\pi \models \psi \cup \psi'$  iff there exists a  $j \geq 0$  such that  $\pi^j \models \psi'$ , and  $\pi^k \models \psi$  for all  $0 \leq k < j$ . □

A formula  $X\psi$  says that  $\psi$  holds *next*, i.e. in the second state of a considered path. A formula  $\psi \cup \psi'$  says that, along a given path,  $\psi$  holds (at least) *until*  $\psi'$  holds. We standardly write  $F\psi$  for  $\top \cup \psi$  (“In the future  $\psi$ ” or “ $\psi$  will hold eventually”),  $G\psi$  for  $\neg F\neg\psi$  (“Globally  $\psi$ ” or “ $\psi$  holds *always* along a path”), and  $A\psi$  for  $\neg E\neg\psi$  (“ $\psi$  holds along *all* paths”). The combinators  $AG$  and  $EF$  can then be interpreted as “in all states” and “in some state” respectively.

The complexity of checking CTL\* formulas is linear in the size of the model but exponential in the size of the formula. We define two most popular (syntactic) restrictions of CTL\* that allow for more optimal verification. A CTL\* state formula of the form  $A\psi$ , where  $\psi$  is a path formula containing no state formulas, is a *linear temporal logic* (LTL) formula. A CTL\* state formula in which every sub-formula of the type  $\psi \cup \psi'$  is prefixed by an  $A$  or  $E$  quantifier, is a *computational tree logic* (CTL) formula. The complexity of LTL model checking is the same as of CTL\*, but the advantage is that LTL formulas can be checked on-the-fly [4]. The complexity of CTL model checking is linear in both the size of the model and the size of the formula, and thus lower than for CTL\* [4]. As we will see later, all our correctness properties belong to either the LTL or the CTL subset (or both). The reason we work with CTL\* is to have a common framework, and to be allowed to (temporarily) jump outside of the restricted domain when rewriting one formula to another.

## 5.2 Unfolding of WFD-Net

Since we use a state-based logic, the *states* of a Kripke structure representing the behavior of a WFD-net, should include information necessary for the formalization of our anti-patterns, namely what happens with the data when some transition is executed. This information is lost if we just build the reachability graph of a WFD-net—e.g. we can see that two transitions writing to a data element  $d$  can be enabled at the same time, but we cannot see whether they can be executed at the same time.

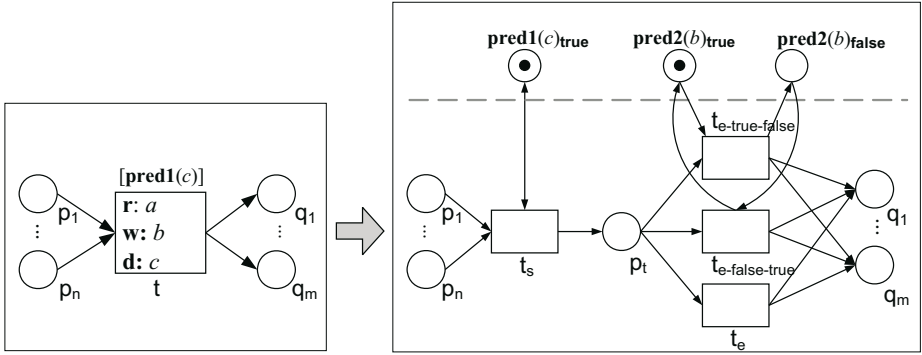
*Preprocessing.* To include the information about the data operations into the states, we use a preprocessing step that converts a WFD-net into a WF-net, while keeping the original structure intact. This step consists of the following smaller steps:

1. We *split* every transition  $t$  into its start  $t_s$  and its end  $t_e$  connected by a place  $p_t$ . A token on  $p_t$  means that transition  $t$  is being executed.
2. To capture the restrictions on the behavior due to guards, we add a “*guard layer*” to our net: For every *predicate*  $\text{pred}$  appearing in some guard we introduce places  $\text{pred}_{\text{true}}$  and  $\text{pred}_{\text{false}}$ . A token on  $\text{pred}_{\text{true}}$  indicates that the predicate is evaluated to true for the current set of data values. A token on  $\text{pred}_{\text{false}}$  means that  $\text{pred}$  evaluates to false.
3. For every transition  $t$  with a *guard*  $\text{pred}$  in the WFD-net, we add an arc from  $\text{pred}_{\text{true}}$  to  $t_s$  and an arc back from  $t_s$  to  $\text{pred}_{\text{true}}$  to our preprocessed net. This self-loop makes sure that  $t$  is executed only when its guard is evaluated to true. For the guard  $\neg\text{pred}$  we add the arcs to the place  $\text{pred}_{\text{false}}$  instead of  $\text{pred}_{\text{true}}$ .
4. A *change* of the value of a data element  $d$  that appears in a predicate  $\text{pred}$  may potentially change the evaluation of  $\text{pred}$ . We reflect that by assuming that every transition  $t$  writing to  $d$  might change the value of  $\text{pred}$  (or not). Therefore, we split  $t_e$  into three transitions: two to represent possible changes of the predicate value (from true to false and from false to true), and one leaving the predicate value unchanged.<sup>1</sup>

Please note that in case the transition changes data items related to  $k$  predicates, it will be in general split into  $3^k$  transitions.

Fig. 3 illustrates the preprocessing for transition  $t$  with a guard  $\text{pred1}(c)$  writing to data element  $b$ . We assume that  $b$  is used in some predicate  $\text{pred2}(b)$ , guarding some other transition(s) of the workflow. Places  $\text{pred1}(c)_{\text{true}}$ ,  $\text{pred1}(c)_{\text{false}}$  (not shown in the figure),  $\text{pred2}(b)_{\text{true}}$  and  $\text{pred2}(b)_{\text{false}}$  are added to represent the values of the predicates. The transition is split into the start transition  $t_s$ , controlled by place  $\text{pred1}(c)_{\text{true}}$ , and transitions  $t_{e\text{-true-false}}$ , changing the value of the predicate  $\text{pred2}$  from true to false,  $t_{e\text{-false-true}}$ , changing the value of the predicate from false to true, and  $t_e$  leaving the value unchanged.

<sup>1</sup> In this paper we assume that predicates do not depend on each other; our method, however, can be easily extended to support dependencies.



**Fig. 3.** Decomposition of a transition in a WFD-net

We make an arbitrary choice assuming that all  $\text{pred}_{true}$  places initially have a token and  $\text{pred}_{false}$  places not. We can afford making an arbitrary choice since the errors related to the use of undefined data for the valuation of guards is captured by DAP  $\square$  and will be signaled as an error, in case it takes place.

*Building the Kripke structure.* The Kripke structure is in fact an extended reachability graph of the preprocessed net. The states of the Kripke structure are states (markings) of the reachability graph and the transition relation is the reachability relation. We define the *set of atomic propositions*  $A = \{p \geq i \mid p \in P, i \in \mathbb{N}\}$  to express properties of markings ( $p \geq i$  means that place  $p$  holds at least  $i$  tokens). The *labels* of states map the markings to the sets of atomic propositions as follows: for some  $p \in P$  and  $i \in \mathbb{N}$ ,  $(p \geq i) \in \mathcal{L}(m)$  iff  $m(p) \geq i$ .

For the sake of readability, we introduce some abbreviations. We write  $p = i$  for  $p \geq i \wedge \neg(p \geq i + 1)$ . To directly formulate that some *transition  $t$  is executing*, we write  $\text{exec}(t)$  instead of  $p_t \geq 1$ . That the workflow is in its *final state* is denoted *term*, defined by  $(\text{end} = 1 \wedge \bigwedge_{p \in P \setminus \{\text{end}\}} (p = 0))$ . To represent the fact that a data element  $d \in \mathcal{D}$  is being *read* by some transition, either as its input or for evaluating a guard, we write  $r(d)$ , abbreviating thus  $\bigvee_{t: d \in \text{Read}(t) \cup \text{data}(\text{Guard}(t))} \text{exec}(t)$ . The constructs  $w(d)$  and  $d(d)$  are defined similarly.

We will use a *convention* that the order of operations within a transition is fixed as first read, then write and after that destroy, which e.g. implies that transition  $t_8$  in Fig.  $\square$  first reads  $f$  and only after that destroys it, i.e. here there is no attempt to read a destroyed data element.

*Example.* We use a simplistic example to show that the addition of the guard layer reduces the number of false positives and false negatives, compared to the analysis on the net without it. Consider the WFD-net from Fig.  $\square$ . If guards are ignored while generating the behavior,  $d'$  will be reported missing in  $t_4$ . This is a false negative, as  $t_4$  can never be enabled— $t_2$  can only be chosen when  $\text{pred}(d)$  is false, and the value of the predicate remains the same when it is evaluated at  $t_4$ . On the other hand, a soundness check on the net with the guard layer will



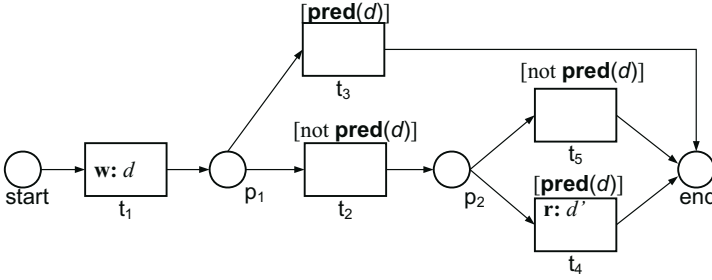


Fig. 4. Data can influence reachability

correctly report that transition  $t_4$  is dead, while the check on the control flow would result in a false positive, saying that the net is sound.

### 5.3 Formalization of Anti-patterns

We explain the formalization process for some of the anti-patterns in detail, and we merely provide the corresponding CTL\* formulas for the rest.

**DAP 7: Missing Data.** A data element  $d$  is missing if there is an execution path along which no writing to  $d$  happens until reading  $d$  or destroying  $d$  takes place. This can be expressed by  $E[-w(d) \cup (r(d) \vee d(d))]$ . A data element  $d$  is also missing if  $d$  first get destroyed and then no writing takes place until  $d$  is read or destroyed, which can be captured by  $EF[d(d) \wedge (\neg w(d) \cup (r(d) \vee d(d)))]$ . The disjunction of these two expressions results in the formalization given in Table 1.

**DAP 2: Strongly Redundant Data.** A data element is strongly redundant if there is a path leading to a writing to  $d$  (i.e.  $EF[w(d) \wedge \dots]$ ) such that in all possible continuations of this path no reading takes place until the workflow terminates or  $d$  get destroyed ( $AX[\neg r(d) \cup (\text{term} \vee (d(d) \wedge \neg r(d)))]$ ). We need  $X$  here because we want to impose  $\neg r(d)$  restriction starting from the next state only, not from the state where the writing in question takes place—reading there would precede the writing, according to our convention. This convention is also the reason for including  $\neg r(d)$  in the conjunction  $d(d) \wedge \neg r(d)$ .

The formalization of **DAP 3: Weakly Redundant Data** differs from its strong counterpart by one letter only: the  $A$  requirement is removed, since it is sufficient to have one path showing the undesired behavior. The principle of formulating **DAP 4** is the same as for **DAP 2**, the principle of formulating **DAP 5**, **DAP 7** and **DAP 8** is the same as for **DAP 3**.

**DAP 6: Inconsistent Data.** A data element  $d$  is inconsistent if some transition  $t$  that changes  $d$  and some transition  $t'$  that uses  $d$  can be executed at the same time, captured by  $\bigvee_{t \in T: d \in \text{change}(t)} EF[(\text{exec}(t) \wedge \bigvee_{t' \neq t: d \in \text{use}(t')} \text{exec}(t'))]$ , or if two or more instances of transition  $t$  changing  $d$  can be executed in parallel, captured by  $\bigvee_{t \in T: d \in \text{change}(t)} EF[p_t \geq 2]$ . Here  $\text{change}(t)$  stands for the set  $\{d \mid$

**Table 1.** Formalization of anti-patterns for a data element  $d$ 

Anti-pattern	Formalization
<b>DAP 1</b> <i>Missing Data</i>	$E[(\neg w(d) \cup (r(d) \vee d(d))) \vee F[d(d) \wedge (\neg w(d) \cup (r(d) \vee d(d)))]]$
<b>DAP 2</b> <i>Strongly Redundant Data</i>	$EF[w(d) \wedge AX[\neg r(d) \cup (\text{term} \vee (d(d) \wedge \neg r(d)))]]$
<b>DAP 3</b> <i>Weakly Redundant Data</i>	$EF[w(d) \wedge X[\neg r(d) \cup (\text{term} \vee (d(d) \wedge \neg r(d)))]]$
<b>DAP 4</b> <i>Strongly Lost Data</i>	$EF[w(d) \wedge AX[\neg(r(d) \vee d(d)) \cup (w(d) \wedge \neg r(d)))]]$
<b>DAP 5</b> <i>Weakly Lost Data</i>	$EF[w(d) \wedge X[\neg(r(d) \vee d(d)) \cup (w(d) \wedge \neg r(d)))]]$
<b>DAP 6</b> <i>Inconsistent Data</i>	$\bigvee_{t \in T: d \in \text{change}(t)} EF[(\text{exec}(t) \wedge \bigvee_{t' \neq t: d \in \text{use}(t')} \text{exec}(t')) \vee p_t \geq 2]$
<b>DAP 7</b> <i>Never destroyed</i>	$EF[w(d) \wedge X[\neg(d(d) \vee w(d)) \cup \text{term}]]]$
<b>DAP 8</b> <i>Twice Destroyed</i>	$EF[d(d) \wedge X[\neg w(d) \cup (d(d) \wedge \neg w(d))]]]$
<b>DAP 9</b> <i>Not Deleted on Time</i>	$\bigvee_{t \in T: d \in (\text{Read}(t) \cup \mathbf{data}(\text{Guard}(t))) \setminus \text{Destroy}(t)} AG[\text{exec}(t) \Rightarrow \text{exec}(t) \cup G(\neg r(d))]$

$d \in \text{Write}(t) \cup \text{Destroy}(t)$ }, and  $\text{use}(t)$  stands for the set  $\{d \mid d \in \text{Read}(t) \cup \mathbf{data}(\text{Guard}(t)) \cup \text{Write}(t) \cup \text{Destroy}(t)\}$ .

**DAP 9: Not Deleted on Time.** To conclude that a data element  $d$  is not deleted on time, we need a transition that reads  $d$  without destroying it (i.e.  $t \in T$  with  $d \in (\text{Read}(t) \cup \mathbf{data}(\text{Guard}(t))) \setminus \text{Destroy}(t)$ ), such that the execution of this transition is never followed by reading  $d$ . This means that for all paths whenever  $t$  is executed ( $AG[\text{exec}(t) \Rightarrow \dots]$ ),  $d$  is never read again after the execution of  $t$  is finished (captured by  $\text{exec}(t) \cup G(\neg r(d))$ ). An additional explanation needed here is that there can be several consecutive states for which  $\text{exec}(t)$  is true, which means that there are events happening in parallel branches while  $t$  continues its execution. The resulting formula is given in Table 1.

All the formulas except for the last one (**DAP 9**) are (or can be rewritten to) CTL formulas. Negations of formulas for DAPs 1, 3, 5, 6, 7 and 8 can be rewritten to LTL. **DAP 9** is a set of LTL formulas itself.

## 5.4 Tool Support

As explained in the previous section, all anti-patterns we identified (or their negations) can be expressed in one of the two most commonly used temporal logics, CTL or LTL. Therefore, to check for data correctness we do not need to build our own tool but can choose from a number of Petri-net model-checkers available (e.g. [10,13,5]). The Model-Checking Kit [10] allows for both CTL and LTL model-checking, and supports a variety of Petri-net modeling languages as

input. Maria [13] is an LTL model-checker, and CPN Tools [5] is a powerful framework for modeling and analysis of Colored Petri nets with the CTL model-checking facility. We used CPN Tools in our verification experiments, and we were able to easily state all the CTL anti-patterns, and to check them within a fraction of a second.

## 6 Conclusion

This paper provides a systematic classification of possible flaws related to the data flow in business workflows. We formulated these flaws as data-flow anti-patterns. To avoid ambiguities inherent to formulations in a natural language, we formalized the anti-patterns in the temporal logic CTL\*. All anti-patterns belong to one of the two (or both) most popular subsets of CTL\*: CTL and LTL. This opens a way to easy tool support for our approach, since there are many model-checkers for both CTL and LTL.

Our approach is a first step towards a unifying framework for the integral analysis of workflows taking into account both control and data flow. As we showed in the example related to Fig. 4 (Subsection 5.2), by including data flow along with control flow into consideration when checking classical properties of workflow like soundness, we can reduce the number of false positives and false negatives caused by (unavoidable) abstraction of data values.

In the future we will try to identify more anti-patterns. We will also build an integrated tool-chain that starts with the check for boundedness, then performs the preprocessing transformations and Kripke structure generation, proceeds in looking for anti-patterns' instances by using an existing model-checker, e.g. [10], and finally generating a verification report for the workflow designer.

## 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 Hee, K.M.: *Workflow Management: Models, Methods, and Systems*. MIT Press, Cambridge (2004)
3. van der Aalst, W.M.P., van Hee, K.M., ter Hofstede, A.H.M., Sidorova, N., Verbeek, H.M.W., Voorhoeve, M., Wynn, M.T.: *Soundness of Workflow Nets: Classification, Decidability, and Analysis*. BPM Center Report BPM-08-02, BPMcenter.org (2008)
4. Clarke, E.M., Grumberg, O., Peled, D.A.: *Model Checking*. The MIT Press, Cambridge (1999)
5. CPN Group, University of Aarhus, Denmark. CPN Tools Home Page, <http://wiki.daimi.au.dk/cpntools/>
6. 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)
7. Eshuis, R.: Symbolic Model Checking of UML Activity Diagrams. *ACM Transactions on Software Engineering Methodology* 15(1), 1–38 (2006)

8. Fan, S., Dou, W.C., Chen, J.: Dual Workflow Nets: Mixed Control/Data-Flow Representation for Workflow Modeling and Verification. In: Chang, K.C.-C., Wang, W., Chen, L., Ellis, C.A., Hsu, C.-H., Tsoi, A.C., Wang, H. (eds.) APWeb/WAIM 2007. LNCS, vol. 4537, pp. 433–444. Springer, Heidelberg (2007)
9. Georgakopoulos, D., Hornick, M., Sheth, A.: An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure. *Distributed and Parallel Databases* 3, 119–153 (1995)
10. Institute of Formal Methods in Computer Science, Software Reliability and Security Group, University of Stuttgart. Model-Checking Kit Home Page, <http://www.informatik.uni-stuttgart.de/fmi/szs/tools/mckit/>
11. Jablonski, S., Bussler, C.: *Workflow Management: Modeling Concepts, Architecture, and Implementation*. International Thomson Computer Press, London (1996)
12. Koenig, A.: Patterns and Antipatterns. *Journal of Object-Oriented Programming* 8(1), 46–48 (1995)
13. Mäkelä, M.: Maria: Modular Reachability Analyser for Algebraic System Nets. In: Esparza, J., Lakos, C.A. (eds.) ICATPN 2002. LNCS, vol. 2360, pp. 434–444. Springer, Heidelberg (2002)
14. Reichert, M., Dadam, P.: ADEPTflex: Supporting Dynamic Changes of Workflow without Loosing Control. *Journal of Intelligent Information Systems* 10(2), 93–129 (1998)
15. Sadiq, S.W., Orlowska, M.E., Sadiq, W., Foulger, C.: Data Flow and Validation in Workflow Modelling. In: Fifteenth Australasian Database Conference (ADC), Dunedin, New Zealand. CRPIT, vol. 27, pp. 207–214. Australian Computer Society (2004)
16. Sadiq, W., Orlowska, M.E.: Analyzing Process Models using Graph Reduction Techniques. *Information Systems* 25(2), 117–134 (2000)
17. Schmidt, D.A.: Data Flow Analysis is Model Checking of Abstract Interpretations. In: Proceedings of the 25th ACM SIGPLAN-SIGACT symposium on Principles of programming languages (POPL 1998), pp. 38–48. ACM, New York (1998)
18. Sun, S.X., Zhao, J.L., Nunamaker, J.F., Liu Sheng, O.R.: Formulating the Data Flow Perspective for Business Process Management. *Information Systems Research* 17(4), 374–391 (2006)
19. Sundari, M.H., Sen, A.K., Bagchi, A.: Detecting Data Flow Errors in Workflows: A Systematic Graph Traversal Approach. In: 17th Workshop on Information Technology & Systems (WITS 2007), Montreal (2007)
20. Verbeek, H.M.W., Basten, T., van der Aalst, W.M.P.: Diagnosing Workflow Processes using Woflan. *The Computer Journal* 44(4), 246–279 (2001)
21. Weske, M.: *Business Process Management: Concepts, Languages, Architectures*. Springer, Berlin (2007)

# Process Algebra-Based Query Workflows

Thomas Hornung<sup>1</sup>, Wolfgang May<sup>2</sup>, and Georg Lausen<sup>1</sup>

<sup>1</sup> Institut für Informatik, Universität Freiburg

[f{hornungt, lausen}@informatik.uni-freiburg.de](mailto:{hornungt, lausen}@informatik.uni-freiburg.de)

<sup>2</sup> Institut für Informatik, Universität Göttingen

[may@informatik.uni-goettingen.de](mailto:may@informatik.uni-goettingen.de)

**Abstract.** In this paper we combine ideas from workflow processing and database query answering. Tailoring process algebras like Milner's *Calculus of Communicating Systems (CCS)* to relational dataflow makes them a natural candidate for specifying data-oriented workflows in a declarative way. In addition to the classical evaluation of relational operator trees, the combination with the CCS control structures provides (guarded) alternatives and test-based iterations using recursive process fragment definitions. For the actual atomic constituents of the process, language concepts from the relational world, like queries, but also the use of abstract datatypes, e.g., graphs, can be embedded.

We illustrate the advantages of the approach by an application scenario with remote, heterogeneous sources and Web Services that return their results asynchronously. The presented approach has been implemented in a prototype.

## 1 Introduction

Most of the information that is needed for daily tasks is available on the Web. The main problem is often not to *get* the information, but to process it efficiently and appropriately in an automatic way. Efficiency does not necessarily mean millions of data items, but often a relatively small number of items, scattered over multiple data sources, and to organize the process of combining, evaluating, making decisions, interacting. Consider for example travel planning: not only the nearest airport to a certain destination has to be found, but depending on the airlines, different airports must be considered, and availability of the flights has to be checked. Then, transportation from/to the airports, possibly provided by local railway companies, has to be arranged. Even employees of travel agencies usually process such enquiries manually, which requires a lot of time and is potentially incomplete and suboptimal. Although the manual process follows a small number of common patterns (e.g., searching for paths in a transitive relationship distributed over several sources, like flight schedules and train schedules, with heuristics for bridging long distances vs. shorter distances, making prereservations, doing backtracking) it is hard to automatize it since the sources are not integrated, and the underlying formalism has to cover both procedural tasks and data manipulation tasks. Often it is easier to design the process *how* to solve

such a problem than stating a single query. Furthermore, most of the data is not immediately available for querying via e.g. query languages like SQL or XQuery, but kept in the *Deep Web*, which consists of dynamically generated result pages, which can only be queried interactively via Web forms.

This technical environment together with the intrinsic complexity of the tasks requires for *flexible* data workflows using a *generic* data model and an extensible set of functional modules, including the ability to interact *actively* with remote services. Important basic functionality includes appropriate mechanisms to deal with *information acquisition* and target-driven *information processing* on a high level, like using design patterns for acting on graph-structured domains.

In the following, we present and discuss an approach that attempts to satisfy the above requirements. The core aspects are the intertwined description of the control flow of the process (by a process algebra, e.g., CCS [14]) and the handling of the dataflow (based on the relational model), and the use of heterogeneous atomic constituents like queries and actions in the workflow: CCS is extended to relational dataflow, called RelCCS, and realized as a language in the *MARS (Modular Active Rules for the Semantic Web)* framework [13] for embedding heterogeneous component languages. RelCCS is complementary to the original rule-based MARS paradigm, and employs the functionality of the MARS framework as an infrastructure.

The focus is not on performance, but on the qualitative ability to express and execute complex workflows and decision processes in a reasonable time – i.e., to replace hours of interactive human Web search by an unsupervised automated process that also may take hours but finally results in one or more proposals, including the optimal one.

The process design/programming in the RelCCS language is not expected to be done by casual users, but by skilled process designers in cooperation with domain experts – analogously to application database design.

*Application Scenario.* Consider the scenario to find either the cheapest or shortest (in terms of total time spent travelling) route to a given location (e.g., for a conference travel) or a combination of both. Human, manual search usually employs some kind of intuitive search strategy. Roughly, the strategy is to try to cover as much distance as possible by plane (assuming the distance is above a certain threshold), and then bridge the remaining distance by train or bus; if this fails, do backtracking. One usually starts with considering a known set of airports near the hometown. This shows that human problem solving, although always considering one possibility (=tuple) at a time, is inherently based on a set-oriented model.

With the means of the presented approach, such tasks can be formulated as data workflows. The backtracking is here replaced by breadth-first search, where the search space is explored stepwise and pruned based on intermediate results.

The actual process can thus be described as (i) determining a set of local airports, e.g., the ten nearest airport to a place, (ii) computing all connections from the starting point to each of these airports, (iii, in parallel) trying to find connections from each of the airports to the destination, and *joining* the results

from (ii) and (iii); always under consideration of arrival and departure times and required time for changing. While for train connections, sources usually are able to return transitive connections, flight portals only return transitive connections over the flights of the same airline. Thus, here an actual graph exploration and search is to be applied.<sup>1</sup>

The expected answer is the *set* of  $k$  best alternatives (wrt. a weighted function of price and duration), where each solution contains the actual connection data (flight and train numbers, departure/arrival times). Furthermore, it should in general be possible to extend the process specification in such a way that the best available one is actually booked automatically.

*Structure of the paper.* Section 2 introduces the RelCCS language. In Section 3, we illustrate the use of the approach by implementing the above example workflow. Related work is discussed in Section 4 before we draw a short conclusion.

## 2 RelCCS: The Relational Dataflow Process Language

RelCCS is a variant tailored to relational dataflow of the well-known *Calculus of Communicating Systems (CCS)* process algebra [14]. It has been designed as a part of the *MARS (Modular Active Rules for the Semantic Web)* framework [13] whose central metaphor is a model and an architecture for active rules and processes that use *heterogeneous* event, query, and action languages. This distinctive feature of MARS proves useful in the present paper, too: it allows to embed sublanguages for queries and even supplementary generic data structures via APIs expressed as actions and queries into the workflows to be specified.

Here we present MARS only as far as it is necessary to get the ideas that are relevant for the realization of RelCCS. The MARS meta-model distinguishes *rules* (not relevant in this paper), *events* (that may also occur in CCS workflows as described in this paper), *queries*, *tests*, and *actions/processes* (cf. Section 2.1); the dataflow is based on sets of tuples of variable bindings (like in Datalog; cf. Section 2.2). The MARS meta-language concept relies on an XML markup for nested expressions of *different* languages throughout whole MARS.

### 2.1 The Process Model: Processes and Their Constituents

*The CCS Process Algebra.* Processes can formally be described by process algebras; the most prominent ones are *CCS – Calculus of Communicating Systems*

---

<sup>1</sup> Experiences with conference travels showed that real travel agencies are often challenged with finding the potential nearest airports to rather unknown destinations (e.g., St.Malo/France), and are rather weak in finding non-direct flight connections using different airlines (e.g., Lufthansa + AirFrance) or via non-expected intermediate airports (via Stansted to reach Dinard/France), or surprising connections (fly to Jersey Island and take the ferry to St.Malo) – actually, ferries are contained in the railway portals, so it is not necessary to find out about individual ferry lines. The latter shows also that it would not be advantageous to try to save time by predefining the set of destination airports by the user, but to use a fully algorithmic search that is not biased in any way.

[14] and *CSP – Communicating Sequential Processes* [10]; we chose CCS as the base to develop RelCCS. A CCS algebra with a carrier set  $\mathcal{A}$  (its atomic constituents) is defined as follows (we consider here the asynchronous variant of CCS that allows for implicit delays), using a set of process variables:

With  $a \in \mathcal{A}$ ,  $X$  a process variable,  $P$  and  $Q$  process expressions,  $X := P$  is a process definition,  $a, X, a.P$  (prefixing; sequential composition),  $(P, Q)$  (sequential composition),  $P|Q$  (concurrent composition), and  $P_1+P_2$  (alternative composition; generally written as  $\sum_{i \in I} P_i$  for a set  $I$  of indexes) are process expressions. The semantics is defined in [14] by transition rules that immediately induce an implementation strategy. By carrying out an action, a process changes into another process:

$$\begin{aligned}
 &a.P \xrightarrow{a} P, \quad \frac{P \xrightarrow{a} P'}{(P, Q) \xrightarrow{a} (P', Q)}, \quad \frac{P_i \xrightarrow{a} P}{\sum_{i \in I} P_i \xrightarrow{a} P} \text{ (for } i \in I \text{)}, \\
 &\frac{P \xrightarrow{a} P'}{P|Q \xrightarrow{a} P'|Q}, \quad \frac{Q \xrightarrow{a} Q'}{P|Q \xrightarrow{a} P|Q'}, \quad \frac{X := P \quad P \xrightarrow{a} P'}{X \xrightarrow{a} P'}.
 \end{aligned}$$

Note that prefixing  $a.Q$  is actually a special case of sequence  $(P, Q)$  where  $P$  is atomic. While in CCS, the state of a process is encoded in its behavior (via the possible actions), we generalize the definition to processes with an explicit state described by sets of tuples of variable bindings in Sections 2.2 and 2.3.

*Atomic Constituents.* While in the basic formalism of CCS, all atomic constituents are considered to be actions, in our approach, atomic constituents are event specifications, queries, tests, and atomic actions:

- atomic actions: these are actually executed as actions, e.g., by Web Services;
- event specifications as atomic constituents: executing an event specification means to wait for an occurrence of the specified event, incorporate the results in the state of the process, and then continue;
- executing a query means to evaluate the query, incorporate the results in the state of the process, and continue the process;
- executing a test means to evaluate it, and incorporate the results in the state of the process, and continue appropriately.

The approach is parametric in the languages used for expressing the constituents. Users write their processes in RelCCS, embedding atomic constituents in sub-languages of their choice. While the semantics of RelCCS provides the global semantics, the constituents are handled by specific services that implement the respective languages.

## 2.2 State, Communication, and Data Flow via Variable Bindings

The state of a process, and the dataflow through the process and to/from the processors of the constituents is provided by *logical variables* in the style of deductive rules, production rules etc.: The state of the computation of a process



is represented by a set of tuples of variable bindings, i.e., every tuple is of the form  $t = \{v_1/x_1, \dots, v_n/x_n\}$  with  $v_1, \dots, v_n$  variables and  $x_1, \dots, x_n$  elements of the underlying domain (which is in our case the set of strings, numbers, and XML literals). Thus, for given active variables  $v_1, \dots, v_m$ , such a state can be seen as a relation whose attributes are the names of the variables. We denote a process expression  $P$  to be executed in a current state  $R$  by  $P[R]$ .

By that, the approach does only minimally constrain the embedded languages. For instance, all paradigms of query languages, following a functional style (such as XPath/XQuery), a logic style (such as Datalog or SPARQL [17]), or both (F-Logic [11]) can be used. The semantics of the event part (that is actually a “query” against an event stream that is evaluated incrementally) is –from that aspect– very similar, and the action part takes a set of tuples of variable bindings as input.

### 2.3 Syntax and Semantics of RelCCS

RelCCS combines the constructs of CCS with relational data flow. Syntactically, it uses mnemonic names (which are also used in its XML markup) instead of the CCS symbol operators.

Let  $\mathcal{P}$  denote the set of process expressions, let  $\mathbb{V}$  denote the set of variable names. For a given finite set  $\text{Var} \subseteq \mathbb{V}$ ,  $\text{Tuples}(\text{Var})$  denotes the set of possible tuples over  $\text{Var}$ . As usual,  $2^{\text{Tuples}(\text{Var})}$  denotes the set of sets of tuples over  $\text{Var}$ . A given set  $R$  of tuples of variable bindings is thus an element  $R \in 2^{\text{Tuples}(\text{Var})}$ .

The mapping  $\llbracket \ ] : \mathcal{P} \times 2^{\text{Tuples}(\text{Var})} \rightarrow 2^{\text{Tuples}(\text{Var})}$  specifies the formal semantics by mapping a process expression  $P \in \mathcal{P}$  and a set  $R$  of tuples of variable bindings to the set  $\llbracket P[R] \rrbracket$  of tuples of variable bindings that result from execution of a process  $P$  for an initial state  $R$ . The definition of this denotational semantics  $\llbracket P[R] \rrbracket$  by structural induction over  $\mathcal{P}$  is given below.

**Example 1.** Consider a simple query  $q$  whose answers are all pairs  $(c, b)$  such that  $c$  is a country and  $b$  is a city in  $c$  with more than one million inhabitants:

$$\begin{aligned} \llbracket q(\{\{c/"Germany"\}, \{c/"Austria"\}, \{c/"Switzerland"\}, \{c/"Joe"\}\}) \rrbracket = \\ \{\{c/"Germany", b/"Berlin"\}, \{c/"Germany", b/"Hamburg"\}, \\ \{c/"Germany", b/"Munich"\}, \{c/"Austria", b/"Vienna"\}\} \end{aligned}$$

There is no resulting tuple for “Switzerland”, because there are no cities with more than one million inhabitants in Switzerland, and there is no resulting tuple for “Joe” since “Joe” is not a country at all. On the other hand, answer tuples to  $q$  like  $\{c/"France", b/"Paris"\}$  do not belong to the result because their value for  $c$  does not match any value of  $c$  of the initial tuples.

Note that  $\llbracket \ ]$  is just the declarative semantics that does neither depend on, nor prescribe the operational details of actual evaluation:  $q[R]$  may be answered by computing  $R \bowtie \sigma[\text{population} > 1000000](\text{City})$  for a suitable database relation  $\text{City}$ , or iteratively a Deep Web query  $q'$  can be stated for every country, yielding e.g.  $q'(\text{"Germany"}) = \{\text{"Berlin"}, \text{"Hamburg"}, \text{"Munich"}\}$  and generating the result set incrementally from the answers.

The situation is similar to the definition of the formal semantics of the relational algebra, and actual query optimization and evaluation.

*Atomic Constituents.* For atomic constituents  $p$ ,  $\llbracket p[R] \rrbracket$  extends  $R$  (Queries, Events), restricts  $R$  (Tests), or (Actions) just uses  $R$  and leaves it unchanged:

- Actions: executing  $\text{Action}(a)[R]$  means to execute  $a$  for each tuple in  $R$  without changing the state  $R$ .  $\llbracket \text{Action}(a)[R] \rrbracket := R$ , plus external side effects of  $a$ .
- Query( $q$ )[ $R$ ]:  $R$  is used to provide the *input parameters* to the query  $q$ . A query  $q$  can be seen as a predicate  $q_0$  (its *characteristic predicate*, which contains all input/output mappings) over variables  $\overline{qv} = \{qv_1, \dots, qv_n\}$ , from which some variables  $\overline{qin} = \{qin_1, \dots, qin_k\} \subseteq \{qv_1, \dots, qv_n\}$  act as input variables, the others  $\overline{qout} = \{qout_1, \dots, qout_m\} = \overline{qv} \setminus \overline{qin}$  act as output variables. Given a tuple  $t \in R$ , the input tuple for  $q$  is  $t_q := \pi[\overline{qin}](t)$ <sup>2</sup> and  $\llbracket \text{Query}(q)[t_q] \rrbracket := \{t' \in q_0 : t_q \subseteq t'\}$ . With this, let  $\llbracket \text{Query}(q)[t] \rrbracket := \{t\} \bowtie \llbracket \text{Query}(q)[t_q] \rrbracket$ , and analogously,  $\llbracket \text{Query}(q)[R] \rrbracket := \bigcup_{t \in R} \llbracket \text{Query}(q)[t] \rrbracket = R \bowtie q_0$ .
- Test( $c$ )[ $R$ ]: the tuples  $r \in R$  that satisfy the test survive:  $\llbracket \text{Test}(c)[R] \rrbracket = \sigma[c](R)$ , like SQL's `SELECT * FROM R WHERE c`.  
Optionally, the test can be parameterized with a quantifier (exists|notExists|all) where the whole set  $R$  of tuples is taken and if one, none, or all tuples  $t \in R$  satisfy the test, the result is  $R$ , otherwise  $\emptyset$ . E.g.,  $\llbracket \text{Test}[\text{exists}](x = 3)[R] \rrbracket = R$  if for some tuple  $t$  in  $R$ , the value of the variable  $x$  is 3, otherwise  $\llbracket \text{Test}[\text{exists}](x = 3)[R] \rrbracket = \emptyset$ .
- Event( $ev$ )[ $R$ ] is analogous to queries: for each tuple of  $R$ , events matching the given event specification are caught and the variable bindings are appropriately extended. For the present application for query answering, events actually play a minor role; they can be used for designing complex workflows manually. Here we just give the semantics for the sake of completeness:  
Given an event occurrence  $ev_0$  that matches the event specification  $ev$  for a certain tuple  $t \in R$  resulting in a set of tuples  $ev_0(t)$ ,  $\llbracket \text{Event}(ev)[R] \rrbracket$  contains  $R \bowtie ev_0(t)$  (the actual semantics of “matching” depends on the embedded event specification language). Thus, at a given timepoint  $\tau$ ,  $\llbracket \text{Event}(ev)[R] \rrbracket = R \bowtie \{ev_0(t) : ev_0 \text{ occurred between “starting” } ev[R] \text{ and } \tau\}$ .

*Operators.* For every evaluation  $P[R]$ , the set  $R$  of initial tuples is modified by executing the process  $P$ , resulting in a new relation  $\llbracket P[R] \rrbracket$  as “outcome” that is returned to the superior process.

- Prefixing, Sequence: execute  $\text{Seq}(P, Q)[R]$  by executing  $P[R]$ , yielding  $R'$ , and then execute  $Q[R']$ . This “common” interpretation of sequence builds actually upon the inner join:  $\llbracket \text{Seq}(P, Q)[R] \rrbracket := \llbracket Q[\llbracket P[R] \rrbracket] \rrbracket$ . More explicitly,  $\llbracket \text{Seq}(P, Q)[R] \rrbracket = \llbracket P[R] \rrbracket \bowtie \llbracket Q[\llbracket P[R] \rrbracket] \rrbracket$  (analogously,  $\llbracket \text{Seq}(P_1, \dots, P_n)[R] \rrbracket$  is defined inductively).

<sup>2</sup> As usual,  $\pi$ ,  $\sigma$ ,  $\rho$  denote relational projection, selection, and renaming.

As a more general idea, tailored to (more accidentally sequential) evaluation of queries, instead of  $\bowtie$ , also left/right/full outer joins make sense, and even a modified form of relational difference as negation: For that, we parameterize  $\text{Seq}$  as  $\text{Seq}[\text{join}]$  (default),  $\text{Seq}[\text{(left|right|full)-outer-join}]$ , and  $\text{Seq}[\text{minus}]$ .

E.g., the semantics of  $\text{Seq}[\text{minus}](P_1, \dots, P_n)$  is defined as follows: Assume  $R_i := \llbracket \text{Seq}[\text{minus}](P_1, \dots, P_i)[R] \rrbracket$  after step  $i$  (for  $i = 0$ :  $R_0 := R$ ) and  $S := \llbracket P_{i+1}[R_i] \rrbracket$  of step  $i+1$ , let  $\llbracket \text{Seq}[\text{minus}](P_1, \dots, P_{i+1})[R] \rrbracket := R_i \setminus (R_i \bowtie S)$ . For example, the query  $q_1(A, B, X) \wedge \neg \exists C, Y : q_2(B, C, Y)$  can be evaluated as  $\text{Seq}[\text{minus}](\text{Query}(q_1(A, B, X)), \text{Query}(q_2(B, C, Y)))$ .

- $\text{Alternative}(P_1, \dots, P_n)[R]$  and  $\text{Union}(P_1, \dots, P_n)[R]$ : each branch is started with  $R$ .

For the (full) union, the result tuples of an alternative or union are the union  $R_1 \cup \dots \cup R_n$  of the results of its branches,  $\llbracket \text{Union}(P_1, \dots, P_n)[R] \rrbracket = \llbracket P_1[R] \rrbracket \cup \dots \cup \llbracket P_n[R] \rrbracket$ .

For the alternative, the following operational restriction holds: All branches have to be guarded, i.e., before the first *action* is executed, a test must be executed (optionally preceded by queries to obtain additional information). For instance, in  $\text{Alternative}(\text{Seq}(\text{Test}(c), P_1), \text{Seq}(\text{Test}(\neg c), P_2))$ , all tuples that satisfy  $c$  will actually run through the first branch, and the others run through the second branch.

If the guards of the branches are exclusive, the alternative is equivalent to the union. If the guards are non-exclusive, the actual outcome is non-deterministic: for each tuple  $t$ , the quicker branch will preempt the others, and exclusively contribute  $\llbracket P_i[t] \rrbracket$  to the result.

- $\text{Concurrent}(P_1, \dots, P_n)[R]$ : each branch is started with  $R$ . The result is  $\llbracket \text{Concurrent}(P_1, \dots, P_n)[R] \rrbracket := \llbracket P_1[R] \rrbracket \bowtie \dots \bowtie \llbracket P_n[R] \rrbracket$ , i.e., *each* tuple runs through *all* branches (possibly being extended with further variables), and the results are joined. Note that if a tuple is removed in some branch, it will not occur at all in the result.

Like for sequences, the operator is also parameterized: in addition to  $\bowtie$ , left/right/full outer join and relational difference are also allowed.

*Complete vs. Partial Answers.* An intuitive and simple model is that the whole set of tuples proceeds synchronously through the process, like the view on relational algebra when taught in courses. The actual execution also covers asynchronous remote services and even *partial* answers, where services return tuples that can be computed quickly, and later send back further tuples.

## 2.4 Recursive Processes in RelCCS

Recursive processes extend the expressiveness from that of relational algebra (trees) to that of recursive Datalog, which e.g. allows to compute transitive closure. Recursive processes are defined by (i) giving and naming a *process definition*, and (ii) then *using* this definition somewhere in the process/tree.

Since logical variables can be bound only once, variables that are bound to different values in each iteration must be considered to be *local* to the current

iteration. They can be bound either when starting the process, or in some step inside the process. Only the final result is then bound to the actual logical variable. For a process expression  $P \in \mathcal{P}$ ,  $pname[\text{local: } lv_1, \dots, lv_n] := P$  defines  $pname$  to be  $P$  where the variables  $lv_1, \dots, lv_n$  are local. Syntactically, the use of process definitions is of the form (e.g. in a sequence)

$$\text{Seq}(\dots, \text{UseDefinition}(pname[lv_{k_1} \leftarrow v_{\ell_1}, \dots, lv_{k_m} \leftarrow v_{\ell_m}]), \dots)$$

with the following semantics: let  $\text{Var}$  denote the set of active variables used in the surrounding context. The definition of  $pname$  is invoked based on the current tuples, where each tuple is extended or modified by initializing the local variables  $lv_{k_1}, \dots, lv_{k_m}$  ( $k_i \in \{1, \dots, n\}$ ) with the values of the variables  $v_{\ell_1}, \dots, v_{\ell_m} \in \text{Var}$ . Formally,

$$\begin{aligned} & \llbracket \text{UseDefinition}(pname[lv_{k_1} \leftarrow v_{\ell_1}, \dots, lv_{k_m} \leftarrow v_{\ell_m}])[R] \rrbracket := \\ & \llbracket P[\{t \in \text{Tuples}(\text{Var} \cup \{lv_{k_1}, \dots, lv_{k_m}\}) \mid \text{exists } t' \in R \text{ s.t.} \\ & \quad \rho[v_{\ell_1} \leftarrow lv_{k_1}, \dots, v_{\ell_m} \leftarrow lv_{k_m}](\pi[\{lv_{k_1}, \dots, lv_{k_m}\}](t)) = \pi[v_{\ell_1}, \dots, v_{\ell_m}](t') \\ & \quad \text{and } \pi[\text{Var} \setminus \{lv_1, \dots, lv_n\}](t) = \pi[\text{Var} \setminus \{lv_1, \dots, lv_n\}](t')] \rrbracket \end{aligned}$$

which is a subset of  $\text{Tuples}(\text{Var} \cup \{lv_1, \dots, lv_n\})$ . Note that recursive processes call themselves inside their definition; in this case,  $\{lv_{k_1}, \dots, lv_{k_m}\} \subseteq \text{Var}$ .

## 2.5 Data-Oriented RelCCS Operators

While the above operators extend the classical CCS operators that focus on the *control flow* with relational state, additional operators integrate unary relational operators: projection, duplicate elimination, and top- $k$ .

*Projection and Duplicate Elimination.* In relational algebra, projection is a very useful operator to reduce intermediate results when some variables are no longer needed. For RelCCS,  $\text{Projection}(v_1, \dots, v_n)$  with a specification which variables to keep does the analogue during execution of a process.  $\text{Distinct}$  removes duplicate tuples, and is usually applied after a projection.

*The RelCCS Top- $K$  Operator.* The top- $k$  operator is known as a useful extension for many applications. It allows to “take the best  $k$  answers” and continue. For instance, when a set of potentially relevant airports are known, only the nearest 10 to the starting place should be considered for continuing the process. Here we adapt the top- $k$  functionality to asynchronous processing of RelCCS workflows: Applied to a set of tuples over variables  $v_1, \dots, v_n$ ,

$\text{TopK}(k, m, t, \text{mapfct}, \text{datatype}, \text{order}, \text{cont})$  acts as follows:

- wait until either  $m$  tuples are present, or  $t$  time units have passed,
- then, for each tuple, compute  $\text{mapfct}(v_1, \dots, v_n)$  (expressed as an embedded query) which yields a value of  $\text{datatype}$ . Order them according to  $\text{order}$  (which can be either  $\text{asc}$  or  $\text{desc}$ ) and take the top  $k$ , and return them.
- if  $\text{cont}$  is  $\text{true}$ , then for every tuple coming in later, check if it is amongst the best  $k$  up to now. If yes, return it, otherwise discard it.

## 2.6 Embedding Algorithmic Webservices

The design of the MARS framework allows to use Web Services that communicate via the atomic metaphors of MARS: actions, tests, queries, and optionally events. Such auxiliary Web Services can for instance provide the functionality of abstract datatypes that embed algorithmic aspects in form of external support in the declarative specification of RelCCS workflows.

A recurring motive when designing query workflows is the computation of (parts of) transitive closures in graphs, as in our travel scenario. For this, a GDT –Graph Data Type– Service provides a configurable API to graphs. Edges and paths in an application usually have properties where the properties of the paths are defined inductively via its edges. For the present paper, we take a GDT instance as given that provides the following actions and queries:

- a query  $gid \leftarrow \text{newGraph}()$  with the side effect of initializing an empty graph,
- an action  $\text{addEdge}(gid, from, to, [p_1 \leftarrow v_1, \dots, p_n \leftarrow v_n])$ , that adds the respective edge with values of  $v_i$  for parameters  $p_i$ , and computes new paths accordingly,
- an accessor (query)  $v \leftarrow \text{getNewVertices}(gid)$  that returns all vertices that have been added since the preceding call of  $\text{getNewVertices}(gid)$ .
- an action  $\text{reportPaths}(start, dest, [v_1 \leftarrow p_1, \dots, v_m \leftarrow p_m])$  that returns all paths that connect  $start$  with  $dest$  with their parameters  $p_i$  bound to variables  $v_i$ .

## 2.7 Technical Realization

RelCCS has been implemented as a language service within the MARS framework. RelCCS processes are given as XML documents (or as RDF graphs), borrowing the main principles from MARS' ECA-ML markup language [13]. The language markup has the usual form of a tree structure over the CCS composers in the `ccs:` namespace. Every expression (i.e., the CCS process, its CCS subprocesses, the event specification, test, queries, and the atomic actions) is an XML (sub)tree whose namespace (i.e., the URI associated with the prefix) indicates the language.

The services are implemented in Java, using a common set of basic classes that handle e.g. the variable bindings. For larger numbers of tuples, an SQL database is used as backend [12]. The actual data exchange is done in an XML format for results and variable bindings. Determining an appropriate service and organizing the communication is performed by a *Languages and Services Registry (LSR)* and a *Generic Request Handler (GRH)* [6].

## 3 Application Scenario: Travel Planning

The RelCCS process for sample travel planning scenario can now be given: If the overall distance is less than 400km, only train connections are searched for. Otherwise, train connections (for less than 800km) and flights are investigated.

For the latter, train connections to potential airports are searched, and the remaining distance is bridged by connecting flights and, if necessary, a final train connection.

Recall that the whole connection graph is not accessible like a database, but must be explored by Web queries. The design of the *process* is thus significantly different from straightforward bottom-up evaluation of transitive closure queries in Datalog. The relevant fragment of the connection graph is built stepwise online during the workflow, using the GDT service as described above. The strategy is based on reachability by breadth first search for shortest paths. Edges (i.e., connections) and their properties (as slotted name-value pairs; i.e., departure and arrival time and price) are obtained from Web queries, and added to the graph. Note that each single connection can be useful in several combinations.

The process uses variables *start* and *dest* (destination), *date* (which are initialized when calling the process), *ap* (relevant near airports), *dist* (distance to airport), *i, j* (intermediate places), *rd* (remaining distance), *dt* and *at* (departure and arrival time), *pr* (price), and *gid* (graph id). We use the prefixes *ccs* and *gdt* to indicate the respective languages. We abstract from the concrete data sources, which are for this example actually wrapped Deep Web sources, e.g. <http://www.bahn.de> for (not only German) Railways, and flights/airline portals:

```
ccs:Seq(ccs:Query(rd ← distance(start, dest)), # process input: (start, dest)
  ccs:Query(gid ← gdt:newGraph()),
  ccs:Union(
    ccs:Seq(ccs:Test(rd < 800), # consider to go by train
      ccs:Query((dt, at, pr) ← getTrainConnection(start, dest, date)),
      ccs:Action(gdt:addEdge(gid, start, dest,
        [deptTimeLocal ← dt, arrTimeLocal ← at, price ← pr]])),
    ccs:Seq(ccs:Test(rd ≥ 400), # consider also to use flights
      ccs:Query(ap ← getAirports()),
      ccs:Query(dist ← distance(start, ap)),
      ccs:TopK(10,100,null,dist,xsd:decimal,asc,false), # consider 10 nearest airports
      ccs:Query((dt, at, pr) ← getTrainConnection(start, ap, date)),
      ccs:Action(gdt:addEdge(gid, start, ap,
        [deptTimeLocal ← dt, arrTimeLocal ← at, price ← pr]]),
      ccs:Projection(start, dest, date), ccs:Distinct,
      ccs:UseDefinition(runGraph[])),
    gdt:reportPaths(start, dest, [pathId ← pid, price ← price])),
ccs:Definition(runGraph[local:i, rd, dt, at, pr] :=
  # additional global vars gid, dest, date are known
  ccs:Seq(
    ccs:Query(i ← gdt:getNewVertices(gid)), # consider all newly reached places
    ccs:Query(rd ← distance(i, dest)),
    ccs:Union(
      ccs:Seq(ccs:Test(i = dest)), # no recursive call in this case → return
      ccs:Seq(ccs:Test(rd < 400 ∧ i ≠ dest ) # reach destination by train
```

```

ccs:Query((dt, at, pr) ← getTrainConnection(i, dest, date)),
ccs:Action(gdt:addEdge(gid, i, dest,
    [deptTimeLocal ← dt, arrTimeLocal ← at, price ← pr])),
ccs:Projection(start, dest, date), ccs:Distinct,
ccs:UseDefinition(runGraph[])),
ccs:Seq(ccs:Test(rd ≥ 200), # try to get even nearer by flight
ccs:Query((j, dt, at, pr) ← getFlights(i, date)),
ccs:Action(gdt:addEdge(gid, i, j,
    [deptTimeLocal ← dt, arrTimeLocal ← at, price ← pr]))
ccs:Projection(start, dest, date), ccs:Distinct,
ccs:UseDefinition(runGraph[])))
# postcondition: reached, either by train or train+flight+, or train+flight++train

```

The workflow proceeds stepwise, set-oriented:

**Example 2.** Consider to start the workflow with the single tuple

$\{start/“Heidelberg”, dest/“St.Malo”, date/“1.1.2009”\}$  .

The query for the remaining distance extends the tuple to

$\{start/“Heidelberg”, dest/“St.Malo”, date/“1.1.2009”, rd/784\}$  .

It then starts two branches, one for a train-only travel (since  $rd < 800$ ), and one that includes consideration of flight connections (since  $rd > 400$ ); the results of both will be returned at the end. We follow the second one:

It will first state a query for all known airports, which results in a set of (thousands of) tuples, where each is extended in the subsequent step with distance between start and the respective airport, i.e.,

```

{ {start/“Heidelnb.”, dest/“St.Malo”, date/“1.1.2009”, rd/784, ap/“FRA”, dist/85},
  {start/“Heidelnb.”, dest/“St.Malo”, date/“1.1.2009”, rd/784, ap/“STG”, dist/95},
  :
  {start/“Heidelnb.”, dest/“St.Malo”, date/“...”, rd/784, ap/“JFK”, dist/6230}, ... }

```

The next topK step keeps the ten nearest ones, amongst them FRA and STG (and not JFK). For these, the next step looks up (multiple) train connections to each of them, binding price etc., and for each tuple, the connection is put into the graph, and the tuples are projected back down to start, dest, date, and duplicates are removed (so only the single tuple  $\{start/“Heidelnb.”, dest/“St.Malo”, date/“1.1.2009”\}$  remains). Then, the process definition for runGraph is invoked. In its first step, the new vertices, which are the 10 nearest airports, are retrieved from the graph and bound to the variable  $i$  (intermediate):

```

{ {start/“Heidelnb.”, dest/“St.Malo”, date/“1.1.2009”, i/“FRA”},
  {start/“Heidelnb.”, dest/“St.Malo”, date/“1.1.2009”, i/“STG”}, ... }

```

The following iterative process is then concerned with extending the graph in parallel (i.e., set-oriented for all tuples) breadth-first search until connections to dest (i.e., when  $i = dest$ ) are found.

Figure 1 illustrates a fragment of the contents of the GDT. The sample also illustrates that (i) some paths are not needed to be inserted as already better ones are known [\*] and, on the other hand, “longer” (in the number of steps) paths can be better wrt. the user’s criteria (price, duration; [\*\*]).

id	head	tail	label	start	end	dept	arr	price
$p_1$	$e_1$	null	HD→FRA	HD	FRA	7:30	8:10	29.00
$p_2$	$e_2$	null	HD→STG	HD	STG	7:50	8:50	39.00
$p_3$	$e_{13}$	$p_1$	HD→FRA→CDG	HD	CDG	7:30	11:20	229.00
$p_4$	$e_{14}$	$p_1$	HD→FRA→LON	HD	LON	7:30	11:50	129.00
$p_5$	$e_{15}$	$p_2$	HD→STG→CDG	HD	CDG	7:50	13:30	289.00
$p_{71}$	$e_{23}$	$p_3$	HD→FRA→CDG→StM	HD	StM	7:30	+1:08:30	345.00
$p_{85}$	$e_{25}$	$p_3$	HD→FRA→CDG→RNS	HD	RNS	7:30	15:20	459.00
$p_{86}$	$e_{25}$	$p_5$	HD→STG→CDG→RNS	HD	RNS	7:50	17:50	519.00 [*]
$p_{93}$	$e_{29}$	$p_4$	HD→FRA→LON→DNR	HD	DNR	7:30	16:50	159.00
$p_{103}$	$e_{25}$	$p_{85}$	HD→FRA→CDG→RNS→StM	HD	StM	7:30	18:20	487.00
$p_{123}$	$e_{39}$	$p_{93}$	HD→FRA→LON→DNR→StM	HD	StM	7:30	17:50	175.00 [**]
:	:	:	:	:	:	:	:	:

Fig. 1. Sample contents of the GDT data structure

The termination condition, i.e., that when all “open” paths are more expensive than the best  $k$  known paths to the destination is enforced by the insertion policy of the graph. This guarantees that the  $k$  preferable paths will be reported, and that the workflow terminates. Recall that such completeness is not guaranteed by the heuristic methods applied by current travel agencies to prune the search space, since this may result in the fact that “unexpected” connections –like reaching St.Malo via Stansted or Jersey– are excluded.

## 4 Related Work

Two already traditional areas that are related to our work are (i) query plans for relational algebra expressions that work on the operator level, and also on the choice of actual algorithms for, e.g., joins, and (ii) conjunctive queries over homogeneous or heterogeneous sources, including HTML and XML Web sources, for querying issues according to the yet classical wrapper-mediator architecture that provide integrated *views* on data, but without considering process-oriented aspects of data-oriented *workflows*. Since in these areas, the control flow does not play a central role, we do not further discuss them.

*Dataflow and Data Exchange: Comparison to Tuple Spaces.* A frequently asked question is the relationship between the dataflow model in MARS and RelCCS, and *Tuple Spaces* [7] (in the following abbreviated as “TS”) and its variants. TS are a middleware approach for cooperation and coordination between distributed processors, in the TS context usually called agents. A TS is an unstructured collection of tuples without fixed schema that allows for associative access: insert,



read, read with delete; updates are accomplished by removing and inserting. IBM TSpaces [19] support four further types of Queries: MatchQuery, IndexQuery, AndQuery, and OrQuery, that all result in sets of tuples.

Similarities between RelCCS and TS are thus in the support for data exchange between autonomous, distributed processors. Also, in both approaches, the data is decoupled from the programs. In TS, data can explicitly exist without being assigned to a certain agent. Communication is anonymous from the point of view of the processors – they get and put tuples from/to the TS.

TS are in many aspects similar to relational databases, but they are used differently. We shortly analyze the main aspects wrt. MARS and RelCCS:

- TS: Use as communication bus, not permanent storage. This characteristic is shared with MARS and RelCCS.
- TS: Unstructured set of tuples. MARS: sets of tuples that belong together.
- TS: Associative access operations. Not needed by MARS and RelCCS.
- TS: Generally, no predefined schema. In MARS and RelCCS, for each state, all tuples have the same schema, which changes during the processing.

So, MARS and RelCCS do not need some of the features of TS. On the other hand, a core requirement of MARS is not covered by TS: Tuples in MARS are grouped into sets of tuples (the above relations over the active variable names) and usually assigned to a (single) current processor, and exchanged between processors in a directed and controlled way. Moreover, the RelCCS operators require to apply *relational operations* on the *sets of tuples*. Functionally, the definition of sets of tuples belonging together could be emulated in TS by an additional column  $c_0$  of the tuples. Nevertheless, TS do not efficiently support *operations* on such sets of tuples, like e.g. joining the result relation  $R$  of a query with the previous tuples, joins of branches of concurrent subprocesses, projection, duplicate elimination, and top- $k$ . For MARS and RelCCS, using a relational database as “communication bus” is preferable since the operations can easily be mapped to relational operations on database tables [12]. Note that there is also a realization without any middleware (except plain internet communication) using data exchange by XML (i.e., sets of tuples serialized as XML) and operations performed on an internal (Java) data structure.

*Workflow and Dataflow.* Several approaches have been presented that combine dataflow with control flow: The focus of Petri Net-based approaches is to express workflows completely in a uniform graphical formalism, with a concise formal semantics to be able to apply formal analysis and verification techniques. Extensions of Petri Nets with nested relational structures are investigated in [15] (*NR/T-nets*) and [9] (*Workflow Nets/Dataflow Nets*). The language YAWL [1], which has been designed based on an exhaustive analysis of workflow patterns [2] and has its roots in Petri Nets, also treats dataflow as a first class citizen.

Petri nets are, like RelCCS, process-oriented. While RelCCS is based on a set of operators, in Petri Nets, the control flow patterns such as concurrent execution and recursion have also to be encoded within the Petri Net formalism. Additionally, abstract data types, such as the GDT, must also be encoded.

Many current approaches to workflow languages, such as BPEL [5] also provide an XML markup. In contrast to RelCCS in the MARS framework, where the XML markup carries important language information for enabling the processing of embedded language fragments, these languages use XML just as a serialization format. Dataflow in BPEL is described by BPEL variables, which can, using appropriate database products like e.g. IBM WebSphere, reference database tables, and thus be made set-valued. Also datatypes like GDT can be embedded into BPEL processes. In [18], optimization strategies of such approaches are discussed.

In *Transaction Logic TR* [4] and *Concurrent Transaction Logic CTR* [16] the description of a workflow consists of *rules* that make use of temporal connectives instead of just the Datalog conjunction. The semantics of *TR* is inherently set-valued. Such rules can be formulated over embedded literals/atoms (called *elementary transitions*) that are *not* part of Transaction Logic, but are contributed externally. This is similar to the embedding of the use of the GDT data type in RelCCS.

Furthermore, systems for data-oriented workflows in general can be applied for query answering tasks. Such systems usually have a set-oriented dataflow. The Lixto Suite [8] is an integrated system for implementing data-oriented workflows with a focus on data acquisition and integration. Its process model is less explicit, and the workflows are solely built upon Lixto's own modules. *Kepler* [3] is an extensible system for design and execution of scientific workflows whose goal is to capture, formalize, and reuse workflows. It supports a concept of individual, reusable workflow steps.

Although these approaches *can* encode the same behavior, the advantage of RelCCS is that it provides both the primitives for control structures and data flow on the same level of the language. A further feature of the language is provided by its embedding in the MARS meta model: RelCCS fragments can be used e.g. as action part in MARS' ECA rules, and fragments in other languages for specifying complex events, queries and atomic actions can be embedded in RelCCS processes without having to revert to Web Services as intermediate wrappers.

## 5 Conclusion

In this paper, we presented the RelCCS approach for specifying and executing data-oriented workflows and discussed its use for solving tedious, repetitive, although complex tasks related to answering queries based on Web data. For such tasks, it is often simpler to design the process *how* to solve the problem, than stating a single query. We also illustrated how complementing module-like data structures can be embedded to support the algorithmic issues of such processes, and gave an impression what processes in this framework look like. Apart from the use for query answering as described above, RelCCS can also be applied for specifying data-oriented workflows in general.

RelCCS is implemented in a prototype which can be found with sample processes and further documentation at <http://www.semwebtech.org/mars/frontend/> → run CCS Process.

## References

1. van der Aalst, W.M.P., ter Hofstede, A.H.M.: YAWL: Yet Another Workflow Language. *Inf. Syst.* 30(4), 245–275 (2005)
2. van der Aalst, W.M.P., ter Hofstede, A.H.M., Kiepuszewski, B., Barros, A.P.: Workflow patterns. *Distributed and Parallel Databases* 14(1), 5–51 (2003)
3. Altintas, I., Berkley, C., Jaeger, E., Jones, M., Ludäscher, B., Mock, S.: Kepler: An extensible system for design and execution of scientific workflows. In: *SSDBM 2004*, pp. 423–424 (2004)
4. Bonner, A.J., Kifer, M.: An overview of Transaction Logic. *Theoretical Computer Science* 133(2), 205–265 (1994)
5. Business Process Execution Language (BPEL), <http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html>
6. Fritzen, O., May, W., Schenk, F.: Markup and Component Interoperability for Active Rules. In: Calvanese, D., Lausen, G. (eds.) *RR 2008. LNCS*, vol. 5341, pp. 197–204. Springer, Heidelberg (2008)
7. Gelernter, D.: Generative communication in Linda. *ACM TOPLAS* 7(1), 80–112 (1985)
8. Gottlob, G., Koch, C., Baumgartner, R., Herzog, M., Flesca, S.: The Lixto data extraction project - back and forth between theory and practice. In: *ACM PODS*, pp. 1–12 (2004)
9. Hidders, J., Kwasnikowska, N., Sroka, J., Tyszkiewicz, J., den Bussche, J.V.: DFL: A Dataflow Language Based On Petri Nets and Nested Relational Calculus. *Inf. Syst.* 33(3), 261–284 (2008)
10. Hoare, C.: *Communicating Sequential Processes*. Prentice-Hall, Englewood Cliffs (1985)
11. Kifer, M., Lausen, G., Wu, J.: Logical foundations of object-oriented and frame-based languages. *Journal of the ACM* 42(4), 741–843 (1995)
12. May, W.: *A Database-Based Service for Handling Logical Variable Bindings. Databases as a Service*, Technical Report, Univ. Münster, Germany (2009)
13. May, W., Alferes, J.J., Amador, R.: Active rules in the Semantic Web: Dealing with language heterogeneity. In: Adi, A., Stoutenburg, S., Tabet, S. (eds.) *RuleML 2005. LNCS*, vol. 3791, pp. 30–44. Springer, Heidelberg (2005)
14. Milner, R.: *Calculi for synchrony and asynchrony*. *Theoretical Computer Science*, 267–310 (1983)
15. Oberweis, A., Sander, P.: Information system behavior specification by high-level Petri Nets. *ACM TOIS* 14(4), 380–420 (1996)
16. Roman, D., Kifer, M.: Reasoning about the Behavior of Semantic Web Services with Concurrent Transaction Logic. In: *VLDB*, pp. 627–638 (2007)
17. SPARQL Query Language for RDF (2006), <http://www.w3.org/TR/rdf-sparql-query/>
18. Vrhovnik, M., Schwarz, H., Suhre, O., Mitschang, B., Markl, V., Maier, A., Kraft, T.: An Approach to Optimize Data Processing in Business Processes. In: *VLDB*, pp. 615–626 (2007)
19. Wyckoff, P., McLaughry, S.W., Lehman, T.J., Ford, D.A.: T Spaces. *IBM Systems Journal* 37(3), 454–474 (1998)

# ETL Workflow Analysis and Verification Using Backwards Constraint Propagation

Jie Liu<sup>1,2</sup>, Senlin Liang<sup>3</sup>, Dan Ye<sup>2</sup>, Jun Wei<sup>2</sup>, and Tao Huang<sup>2</sup>

<sup>1</sup> University of Science and Technology of China, Anhui Hefei, China

<sup>2</sup> Institute of Software, Chinese Academy of Sciences, Beijing, China  
{ljie,yedan,wj,tao}@otcaix.iscas.ac.cn

<sup>3</sup> Department of Computer Science  
State University of New York at Stony Brook  
Stony Brook, NY 11794, USA  
sliang@cs.sunysb.edu

**Abstract.** One major contribution of data warehouses is to support better decision making by facilitating data analysis, and therefore data quality is of primary importance. ETL is the process that extracts, transforms, and ultimately loads data into target warehouses. Although ETL workflows can be designed by ETL tools, data exceptions are largely left to human analysis and handled inadequately. Early detection of exceptions helps to improve the stability and efficiency of ETL workflows. To achieve this goal, a novel approach, Backwards Constraint Propagation (BCP), is proposed that automatically analyzes ETL workflows and verifies the target-end restrictions at their earliest points. BCP builds an ETL graph out of a given ETL workflow, encodes the target-end restrictions as integrity constraints, and propagates them backwards from target to sources through the ETL graph by applying constraint projection rules. It is showed that BCP supports most relational algebra operators and data transformation functions.

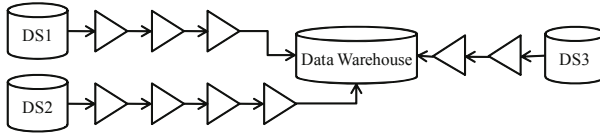
**Keywords:** ETL, Workflow Analysis, Data Quality, Data Warehouse, Constraint Propagation.

## 1 Introduction

ETL (Extract, Transform, and Load) is the important process to build data warehouses [1], and it involves identifying relevant information at data sources, extracting the relevant data, transforming it to fit business needs and ultimately loading the data into the target data warehouse. Initially, ETL processes were hard-coded, and thus difficult and expensive to maintain. Nowadays most database and data integration systems vendors offer powerful ETL tools, most of which can be classified into two categories. One provides independent engines to run ETL tasks, and its representatives are PowerCenter [2] from Informatica,

---

<sup>1</sup> [http://www.informatica.com/products\\_services/powercenter/Pages/index.aspx](http://www.informatica.com/products_services/powercenter/Pages/index.aspx)



**Fig. 1.** An Example of ETL Workflows

Data Stage<sup>2</sup> from IBM, and SSIS<sup>3</sup> from Microsoft. The other category is also named ELT (Extract, Load, Transform), which delegates the task of running ETL workflows to DBMS. Its representative is Oracle Data Integrator<sup>4</sup>.

An example ETL workflow is presented in Figure 1, which integrates data from several data sources (DS1, DS2 and DS3) into target data warehouse by applying series of transformations (triangles). ETL workflows are designed by ETL tools manually, using schema mappings between sources and target. Due to the diversity of source schema and complexity of ETL workflows, some dirty data may cause exceptions when going through workflows or being loaded into the target.

There have been many researches in exceptional data detection and data cleaning, and most vendors provide data quality control suites together with their ETL tools. They mainly employ the following methods: *data analysis*, *data sampling*, *data quality monitoring* and *target-end integrity checking*.

**Data Analysis** analyzes real data instances to obtain data characteristics and value patterns, which help detect data outliers and build schema mappings [2]. There are two approaches to data analysis: *data profiling* and *data mining*. Data profiling derives the information of each attribute (e.g. type, uniqueness, null values), and data mining discovers data patterns (e.g. relationships among several attributes).

**Data sampling** applies ETL workflows on a sample data set to find exceptional data and improves the design of ETL workflows. Data sampling is commonly used in practice, and its performance heavily depends on the quality of the sampling data set.

**Data quality monitoring** imposes quality control by defining *data quality rules* over ETL workflows. The system decides whether to abort the ETL workflows or to perform data cleaning on detecting the violation of these rules. However, these rules have to be manually defined, and most of time it is infeasible to define rules to capture all possible exceptions in real applications.

**Target-end integrity checking** filters out exceptional data, which violates the target-end integrity constraints, when loading into the target warehouse and keeps a record of them for later processing.

Only target-end integrity checking can guarantee the satisfiability of all the constraints at the target. However, all involved data (exceptional or not) are

<sup>2</sup> <http://www-01.ibm.com/software/data/infosphere/datastage>

<sup>3</sup> <http://code.msdn.microsoft.com/SSIS/Wiki/View.aspx>

<sup>4</sup> <http://www.oracle.com/technology/products/oracle-data-integrator/index.html>

transferred to the target, which incurs much extra cost. Moreover, data analysis and data quality monitoring require manual analysis of ETL processes and manual encoding of mappings between the target and sources, which are expensive, error-prone and time consuming. Thus we want an approach that can *automatically*: 1) analyze ETL workflows and the target-end restrictions, and verify these restrictions at their earliest possible points; 2) on the violation of these restrictions, call predefined handlers according to user specified exception handling policies.

Most restrictions at the target warehouse can be expressed by a set of *integrity constraints*, which are also referred as *data quality rules*. In order for the data in the ETL workflow to satisfy the integrity constraints at the target, it should have certain properties when it moves through the ETL workflow. These properties can be derived from the target's integrity constraints.

In this paper, we proposed a novel approach, **Backwards Constraint Propagation (BCP)**, that automatically analyzes ETL workflows and verifies target-end integrity constraints at their earliest possible points.

1. BCP analyzes ETL workflows and target-end restrictions using an abstract interpretation, *ETL graph*.
2. It reports the earliest points in ETL workflows where these restrictions can be verified.
3. On detecting the violations of these restrictions, BCP calls predefined exception handlers.
4. BCP verifies whether ETL workflows contain contradictions.

Given an ETL process, *BCP* builds an *ETL graph*, which extends the *query tree* in [3] and the *system graph* in [4], [5]. Then it pushes target-end restrictions backwards to the data sources by *constraints projections* [6], [7].

BCP can be used as a preprocessing step and applied to both categories of ETL tools. To evaluate its performance, we have implemented a prototype in our ETL tool OnceDI [8]. The preliminary results show that 1) BCP propagates most target-end restrictions back through ETL workflows, and places them at the earliest points where they can be verified; 2) our constraint propagation method supports all the relational algebra operators and data manipulation functions if they are monotonic and have inverse functions, which most involved functions in ETL workflows satisfy.

This paper is organized as follows. Section 2 gives a motivating example. Section 3 describes the BCP framework including constraints projection, automatic data quality rule generation and verification. Section 4 discusses related works, and Section 5 concludes the paper.

## 2 One Motivating Example

This section gives a motivating example, which is used as the running example in this paper.

*Example 1. employee-department example:* suppose a company has two headquarters, and both keep their employee and department information locally at two data sources  $DS_1$  and  $DS_2$ . We want to integrate them into target data warehouse  $DW$ .

DS<sub>1</sub> is located in America, and contains two tables `employee` and `department`:

```
employee(EId, EName, Address, DeptId, StartDate, Salary)
department(DeptId, DeptName, DirectorId, Revenue)
```

Relation `employee` has a tuple for every employee: `EId` is a unique integer ID assigned to each employee and the primary key; `EName` is the employee name; `Address` records the employee's home address; `DeptId` is the ID of the employee's department and a foreign key referring to the attribute `DeptId` of table `department`; `StartDate` is the date when the employee started working in department `DeptId`, and it is of `DATE` type and "MM-DD-YYYY" format; `Salary` is the annual salary in US dollars. Relation `department` has a tuple for every department: `DeptId` is a unique integer ID assigned to each department and the primary key; `DeptName` is department name; `DirectorId` is the ID of the department's director and a foreign key referring to the attribute `EId` of table `employee`; `Revenue` is the annual revenue of the department in US dollars.

DS<sub>2</sub> is located in China, and also contains two tables `employee` and `department`:

```
employee(EId, EName, DeptId, StartDate, Salary)
department(DeptId, DeptName, DirectorId, Revenue)
```

Their differences from DS<sub>1</sub> are: 1) attribute `Address` is not recorded; 2) `StartDate` is of "YYYY-MM-DD" format, such as "2001-11-24"; 3) `Salary` and `Revenue` are in Chinese RMB instead of US dollars.

DW is located in America, and contains one relation defined as:

```
seniorEmpInLargeDept(EId, EName, StartDate, Salary,
DeptName, Size, Revenue, Source)
```

It has a tuple for each employee, and the primary key is `EId`. `StartDate` is in the format of MM-DD-YYYY, `Salary` and `Revenue` are in US dollars, `Size` is the number of employees in department `DeptName`, and `Source` denotes its data source.

In this problem data from both sources have to be processed before being loaded into DW. Without loss of generality, we used the ETL logic model introduced in [9] to represent the ETL process. The ETL workflow is shown in Figure 2, where *triangles* represent SQL operations or data manipulation functions. Assume there is an intermediate result relation `resultTable(X)` of an operation (triangle) X. We detail each operation as:

- A1: join tables `employee` and `department` from DS<sub>1</sub> on `DeptID`.
- A2: count (aggregate operation) `EId` as `Size` grouping by `DeptId`, `DeptName`, `Revenue`.
- A3: join tables `resultTable(A2)` and `employee` from DS<sub>1</sub>.
- A4: add attribute `Source` to table `resultTable(A3)`, and set its values as "DS<sub>1</sub>".
- A5: project out attribute `DeptId` from table `resultTable(A4)`, get `resultTable(A5)` (`EId`, `EName`, `StartDate`, `Salary`, `DeptName`, `Size`, `Revenue`, `Source`).

B1-B5: similar to A1-A5.

B6: convert `StartDate` format from "YYYY-MM-DD" to "MM-DD-YYYY".

B7: convert `Salary` from Chinese RMB to US dollars.

B8: convert `Revenue` from Chinese RMB to US dollars.

C1: union tables `resultTable(A5)` and `resultTable(B8)`.

Suppose the set of restrictions at the target `ic(DW)` consists of the following five integrity constraints:

- DW-IC1: all employees have a minimal salary of 50,000 US dollars.
- DW-IC2: all departments have a minimal revenue of 1,000,000 US dollars.
- DW-IC3: all departments have at least 10 employees.
- DW-IC4: `notNull(Salary)`, salary can not be NULL.
- DW-IC5: `pk(EId)`, `EId` is the primary key.

In order for the data arriving at the target warehouse to satisfy the set of integrity constraints `ic(DW)`, the data at a certain point `p` in the workflow should have some properties derivable from `ic(DW)`. These properties can be encoded as a set of integrity constraints `ic(p)` and verified at point `p`. Our approach to derive the integrity constraints over an ETL workflow and to place them at appropriate points is based on its ETL graph (an abstract interpretation).

An ETL graph is a *labeled* directed acyclic graph (DAG). The construction algorithm is formalized in Section 3. There are two types of nodes: *relation nodes* and *axiom nodes*. Relation nodes represent relations, and axiom nodes represent SQL operations or data manipulation functions. There are a set of data quality rules `rules( $\alpha$ )` and exception handlers `handlers( $\alpha$ )` associated with each relation node  $\alpha$ , and a set of *mapping rules* with each axiom node. The mapping rules of an axiom node  $\beta$ , `mappings( $\beta$ )`, is an abstract interpretation of the operation  $\beta$ . Edges capture the interactions between relation nodes and axiom nodes. Edges are also referred as *ports*: incoming edges are referred as *input ports* and outgoing edges as *output ports*. The intuition is that data goes from data sources to the target along the edges. Relation nodes receive tuples from input ports, verify its set of data quality rules, and send tuples to output ports

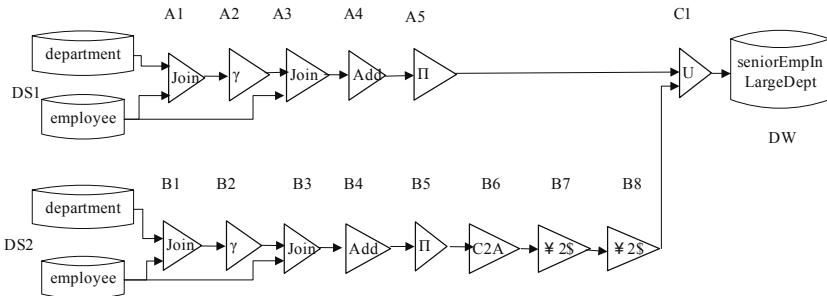


Fig. 2. ETL Workflow of Example 1



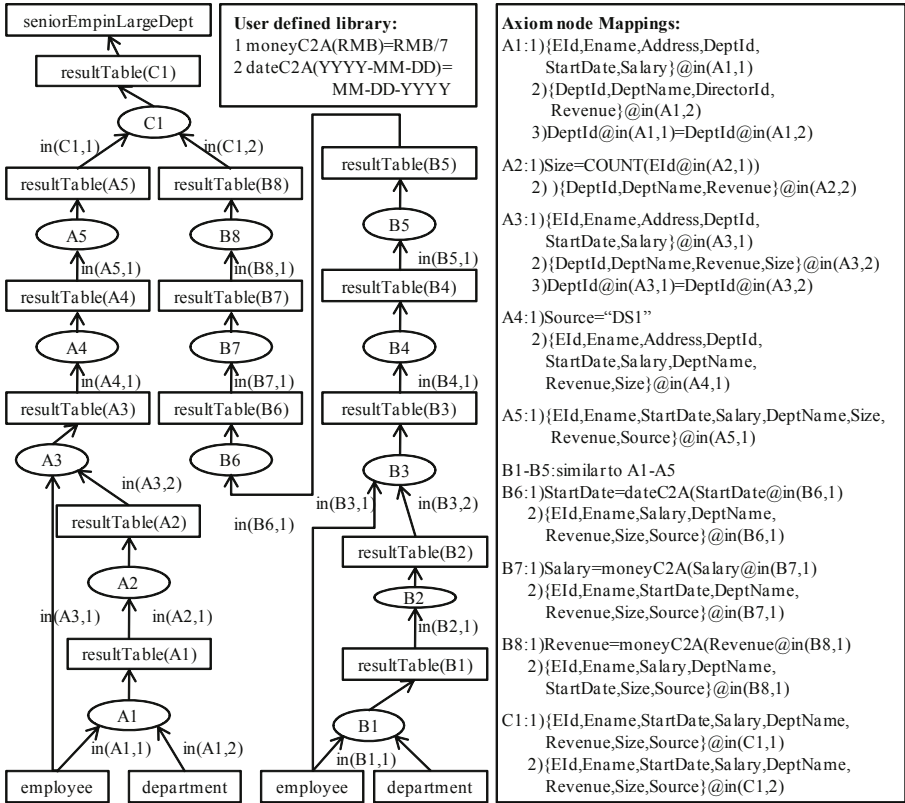


Fig. 3. ETL Graph of Example 11

if there is no exception. Otherwise, predefined exception handlers are called. Axiom nodes receive tuples from input ports, perform the stored operations, and send the resulting tuples to output ports.

The proposed ETL graph of the ETL process is shown in Figure 3. It also includes a user defined *data transformation library*, which defines how to perform the *StartDate* format conversion from China to America (function *dateC2A*), and how to convert Chinese RMB to US dollars (function *moneyC2A*).

BCP builds an ETL graph out of given ETL workflow, automatically derives the data quality rules  $rules(\alpha)$  of relation node  $\alpha$ , and pushes  $rules(\alpha)$  backwards to data sources against the directions of data flows by applying constraint projection rules. Our constraint projection rules can push the constraints further back to data sources through functions if they are bijective and monotonic. Consider DW-IC1, at axiom node B7, we know that

$$Salary = moneyC2A(Salary@in(B7, 1)),$$

and function *moneyC2A* is bijective and monotonically increasing, which gives us

$$moneyC2A(Salary@in(B7, 1)) \geq 50,000.$$

After applying simple mathematical calculation, we know the incoming tuples to axiom node B7 from input port `in(B7, 1)` should have the property that

$$\text{Salary} \geq 350,000$$

which therefore is pushed to `resultTable(B6)` by adding it to `rules(resultTable(B6))`. Finally we can push this constraint down to `DS2`, which is the earliest place it can be verified.

### 3 ETL Workflow Analysis and Verification

#### 3.1 BCP Framework

A framewrok is presented in Figure 4 to demonstrate how to integrate BCP into current ETL tools. The input of BCP module is the ETL graph of and target-end integrity constraints. The output is the data quality rules of each relation node. *One step constraint projection* algorithm is applied to each atom node to push constraints backwards. The *redundancy resolve* unit removes the redundant rules of each relation node. The implementation details of BCP in our ETL tool OnceDI and the preliminary results can be found in [10], and they are omitted here due to space limitations.

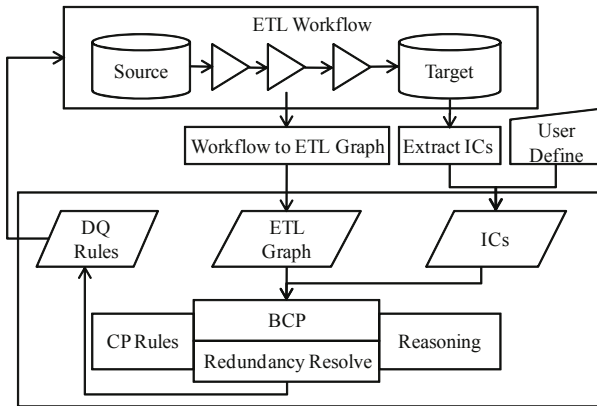


Fig. 4. BCP Framework

#### 3.2 ETL Graph

ETL graph is an extension of the *query tree* in [3] and the *system graph* in [4], [5]. Before proceeding to the algorithm of ETL graph construction, we give some useful definitions (some have been used above informally).

**Definition 1.** Given an operation with label  $\alpha$  in the given ETL workflow  $\Omega$ , its producers,  $\text{producers}(\alpha)$ , is the set of relations that provide data to operation  $\alpha$ , and its consumers,  $\text{consumers}(\alpha)$ , is the set of operations to whom operation  $\alpha$  provides data.

For instance, in Example 1 of Section 2, the producers of operation A3,  $\text{producers}(A3) = \{\text{resultTable}(A2), \text{employee}\}$ .

**Definition 2.** A data quality rule is an integrity constraint supported by DBMS (e.g., not null, unique, primary key, foreign key and check), or a user defined constraint of the form  $R_1(\bar{X}_1) \wedge \dots \wedge R_n(\bar{X}_n) \wedge \varphi(x_1, \dots, x_m)$ , where  $R_i$  is a table,  $\bar{X}_i$  ( $1 \leq i \leq n$ ) is a vector of attributes,  $x_j$  ( $1 \leq j \leq m$ ) is a constant or an attribute in  $\cup_{1 \leq i \leq n}(\bar{X}_i)$ , and  $\varphi$  is a built-in binary predicate that evaluates to true or false (e.g., =,  $\leq$ ,  $\geq$ ).

When the involved relations are obvious, we omit them when expressing a data quality rule. For example we used  $\text{Salary} \geq 50,000$  without mentioning its involved relation `seniorEmpInLargeDept` in Example 1 of Section 2. The set of integrity constraints  $\text{ic}(\text{DW})$  in Example 1 are encoded as data quality rules  $\text{rules}(\text{DW})$ :  $\text{Salary} \geq 50,000$ ,  $\text{Revenue} \geq 1,000,000$ ,  $\text{Size} \geq 10$ ,  $\text{notNull}(\text{Salary})$ , and  $\text{pk}(\text{EId})$ .

Given an ETL workflow  $\Omega$ , the ETL graph  $G^\Omega$  is built in the following steps: 1) create an axiom node  $\alpha$  for each SQL operation or data manipulation operation (triangle)  $\alpha$  in  $\Omega$ , and compute  $\text{mappings}(\alpha)$ ; 2) create a result relation node  $\text{resultTable}(\alpha)$  for each axiom node  $\alpha$ , initialize  $\text{rules}(\text{resultTable}(\alpha)) = \emptyset$ , and set its exception handling policies  $\text{handlers}(\text{resultTable}(\alpha))$ ; 3) add edges between axiom nodes and relation nodes. The algorithm is as follows:

**Algorithm 1.** ETL Graph Construction

```

for (each operation with label  $\alpha$  in  $\Omega$ ) {
  create an axiom node with the same label  $\alpha$ ;
  compute  $\text{mappings}(\alpha)$ ;
  create a result relation node  $\text{resultTable}(\alpha)$ ;
  initialize  $\text{rules}(\text{resultTable}(\alpha)) = \emptyset$ ;
  set  $\text{handlers}(\text{resultTable}(\alpha))$ ;
}
for (each axiom node  $\alpha$ ) {
  add edge  $\text{out}(\alpha)$  from axiom node  $\alpha$  to  $\text{resultTable}(\alpha)$ ;
  index = 1;
  for (each element  $i \in \text{producers}(\alpha)$ ) {
    add edge  $\text{in}(\alpha, \text{index})$  from relation node  $i$  to axiom node  $\alpha$ ;
    index ++;
  }
}

```

For a given atom node  $\alpha$  associated with operation  $\text{op}$ , we suffix each table and its attributes in  $\text{op}$  with its input port label, and create  $\text{mappings}(\alpha)$  to capture the mappings from its input attributes to output attributes. The mappings for different operations are generated differently as shown in Figure 5, where  $\text{attrs}(X)$  returns the set of attributes involved in  $X$ , which can be a relation, condition, rule, or function;  $\text{annotate}(X, \text{inport})$  annotates each attribute  $y$  in  $X$  with its input port label by replacing  $y$  with  $y@inport$ ;  $S@in(\alpha, i)$

Operations		Mappings
Selection	Select * From rel@in Where condition	1) attrs(rel)@in 2) annotate(condition,in)
Projection	Select col1,...,coln From rel@in	1) (col,...,coln)@in
Rename	Select oldname as newname From rel@in	1) newname=oldname@in 2) attr(rel)-{oldname}@in
Add field	Add newname to rel@in	1) attrs(rel)@in
Ordering	Select * From rel@in Order by ....	1) attrs(rel)@in
Aggregate	Select groupByAttrs,f(oldname) as newname From rel@in Where condition Group by groupByAttrs Having ...	1) {groupByAttrs}@in 2) annotate(condition,in) 3) newname=f(oldname)
Function	newname = f(oldname)	1) newname=f(oldname) 2) attrs(rel)-{oldname}@in
Join/ Intersection/ Union/-/×	rel1@in1 join/intersect/union /-/× rel2@in2	1) attrs(rel1)@in1 2) attrs(rel2)@in2

Fig. 5. Mapping Generation of ETL Operations

means the set of output attributes  $S$  of axiom node  $\alpha$  are from its input port  $\text{in}(\alpha, i)$ . For example of the second mapping of axiom node A3 in Figure 3,  $\{\text{DeptId}, \text{DeptName}, \text{Revenue}, \text{Size}\}@\text{in}(\text{A3}, 2)$  means that attributes DeptId, DeptName, Revenue, and Size are from input port  $\text{in}(\text{A3}, 2)$ .

### 3.3 Constraint Propagation

After the ETL graph of a given workflow is constructed, we initialize the set of data quality rules at the target warehouse ( $\text{rules}(\text{DW})$  in Example 1) as its restrictions. Then our approach applies constraint projection rules to propagate these rules backwards to data sources. The constraint propagation algorithm support all relational algebra operators and most ETL workflow data manipulation functions (they are monotonic and have inverse functions).

**Definition 3.** The closure of a set of rules  $S$ ,  $\text{closure}(S)$ , is the set of all rules that are logically implied by  $S$ .

For example of rules  $S = \{x \leq 10, x = y\}$ ,  $\text{closure}(S) = \{x \leq 10, x = y, y \leq 10\}$ , since  $y \leq 10$  is logically implied by  $S$ .

**Definition 4.** Given an axiom node  $\alpha$  with input ports  $\text{in}(\alpha, 1), \dots, \text{in}(\alpha, n)$  from tables  $\text{table}_1, \dots, \text{table}_n$  respectively, one step constraint projection

$\Gamma(\alpha)$  propagates data quality rules  $\text{rules}(\text{resultTable}(\alpha))$  to tables  $\text{table}_1, \dots, \text{table}_n$  through axiom node  $\alpha$ .

One step constraint projection  $\Gamma(\alpha)$  works in the following three steps:

1. compute the union  $\Psi(\alpha) = \text{rules}(\text{resultTable}(\alpha)) \cup \text{mappings}(\alpha)$ ;
2. compute the closure of the union  $\text{closure}(\Psi(\alpha))$ ;
3. project each data quality rule in  $\text{closure}(\Psi(\alpha))$  onto every producer in  $\text{producers}(\alpha)$ .

In step 1, before performing union, each mapping in  $\text{mappings}(\alpha)$  of the format  $\{\text{attr}_1, \dots, \text{attr}_n\}@\text{inport}$  is replaced with a set of rules  $\{\text{attr}_1 = \text{attr}_1@\text{inport}, \dots, \text{attr}_n = \text{attr}_n@\text{inport}\}$ . After this step, there are built-in predicates in  $\Psi(\alpha)$ . The attributes with “@inport” are from  $\text{producers}(\alpha)$ .

In step 2, we compute the closure  $\text{closure}(\Psi(\alpha))$ . If a contradiction is detected when computing the closure, the whole constraint propagation is aborted and an error is reported: *there are errors in the ETL workflow design*. For instance, if  $x < 10 \in \text{rules}(\text{resultTable}(\alpha))$  and  $\text{mappings}(\alpha) = \{x@\text{inport} > 20, x = x@\text{inport}\}$ , then the union  $\Psi(\alpha) = \{x < 10, x@\text{inport} > 20, x = x@\text{inport}\}$ . We can infer that both  $x@\text{inport} < 10$  and  $x@\text{inport} > 20$  are in  $\text{closure}(\Psi(\alpha))$ , which is a contradiction.

Step 3 works differently for different data quality rules and axiom nodes. We elaborate below on how to project a data quality rule  $r$  onto table  $\text{table}_i$  from input port  $\text{in}(\alpha, i)$ .

**Operations without functions:** *selection, projection, rename, union, intersection, difference, product, join, not null, add field, ordering*. There is no function application involved in these operations. Therefore, it is simpler to handle and the constraint projection rule, Rule 1, can be applied.

**Rule 1.** If  $\text{attrs}(r) \subseteq \text{attrs}(\text{table}_i)$ , then data quality rule  $r$  is added to  $\text{rules}(\text{table}_i)$ .

*Example 2.* Consider the data quality rules in Example [1](#), we know:

$$\begin{aligned} \text{rules}(\text{DW}) &= \text{rules}(\text{resultTable}(\text{C1})) \\ &= \{\text{Salary} \geq 50,000, \text{Revenue} \geq 1,000,000, \\ &\quad \text{Size} \geq 10, \text{notNull}(\text{Salary}), \text{pk}(\text{EId})\}. \end{aligned}$$

Since operations  $\{C1, A5, A4\}$  do not change the involved set of attributes, by Rule 1, we get:

$$\begin{aligned} \text{rules}(\text{resultTable}(\text{C1})) &= \text{rules}(\text{resultTable}(\text{A5})) = \\ \text{rules}(\text{resultTable}(\text{A4})) &= \text{rules}(\text{resultTable}(\text{A3})). \end{aligned}$$

But operation A3 changes the involved set of attributes via  $\text{mappings}(\text{A3})$ , which pushes  $\text{rules}(\text{resultTable}(\text{A3}))$  onto input port  $\text{in}(\text{A3}, 2)$  as:

$$\text{rules}(\text{resultTable}(\text{A2})) = \{\text{Revenue} \geq 1,000,000, \text{Size} \geq 10\}$$

We also notice that some rules are relaxed in this process. If a data quality rule involves more than one tuples in some relation, it is relaxed when it is pushed through *union*, *intersection*, *product*, and *join*. For example,  $\text{pk}(\text{EId})$  in  $\text{rules}(\text{resultTable}(\text{A5}))$  and  $\text{pk}(\text{EId})$  in  $\text{rules}(\text{resultTable}(\text{B8}))$  cannot guarantee  $\text{pk}(\text{EId})$  in  $\text{rules}(\text{DW})$ .

**Operations with functions:** *aggregation*, *function application*. There are function applications involved in these operations, and thus more complex.

**Rule 2.1.** If axiom node  $\alpha$  is aggregate operation on  $\text{table}_i$   $\text{min}(\text{attr})$  and  $r$  is  $\text{min}(\text{attr}) \geq \text{val}$  (or  $\text{min}(\text{attr}) > \text{val}$ ), then rule  $\text{attr} \geq \text{val}$  (or  $\text{attr} > \text{val}$ ) is added to  $\text{rules}(\text{table}_i)$ .

**Rule 2.2.** If axiom node  $\alpha$  is aggregate operation on  $\text{table}_i$   $\text{max}(\text{attr})$  and  $r$  is  $\text{max}(\text{attr}) \leq \text{val}$  (or  $\text{max}(\text{attr}) < \text{val}$ ), then rule  $\text{attr} \leq \text{val}$  (or  $\text{attr} < \text{val}$ ) is added to  $\text{rules}(\text{table}_i)$ .

**Rule 2.3.** If axiom node  $\alpha$  is aggregate operation on  $\text{table}_i$  which is grouped by  $\text{groupByAttrs}$  and  $\text{attrs}(r) \subseteq \text{groupeByAttrs}$ , then data quality rule  $r$  is added to  $\text{rules}(\text{table}_i)$ .

*Example 3.* In Example 2, we know  $\text{rules}(\text{resultTable}(\text{A2})) = \{\text{Revenue} \geq 1,000,000, \text{Size} \geq 10\}$ . Axiom node A2 is an aggregate operation on attributes  $\{\text{DeptId}, \text{DeptName}, \text{Revenue}\}$ , by Rule 2.3,  $\text{Revenue} \geq 1,000,000$  is pushed down to  $\text{resultTable}(\text{A1})$ , while  $\text{Size} \geq 10$  is not.

**Rule 3.** If axiom node  $\alpha$  is a function application  $\text{newname} = f(\text{oldname})$ ,  $f$  has an inverse function and monotonic, and  $\text{attrs}(r) \subseteq \text{attrs}(\text{table}_i)$ , then  $\text{replace}(r, \text{newname}, f(\text{oldname}))$  is added to  $\text{rules}(\text{table}_i)$ , where  $\text{replace}(r, \text{newname}, f(\text{oldname}))$  replaces every occurrence of  $\text{newname}$  in rule  $r$  with  $f(\text{oldname})$ .

*Example 4.* In Example 1, we know that

$$\text{rules}(\text{resultTable}(\text{B8})) = \{\text{Salary} \geq 50,000, \text{Revenue} \geq 1,000,000, \text{Size} \geq 10, \text{notNull}(\text{Salary}), \text{pk}(\text{EId})\}.$$

Axiom node B8 converts Revenue from Chinese RMB to US dollars by applying

$$\begin{aligned} \text{Revenue} &= \text{moneyC2A}(\text{Revenue}@_{\text{in}}(\text{B8}, 1)) \\ &= \text{Revenue}@_{\text{in}}(\text{B8}, 1)/7 \end{aligned}$$

We know  $\text{moneyC2A}$  is monotonically increasing and bijective. By Rule 3, it can be pushed down to  $\text{resultTable}(\text{B7})$  as

$$\text{Revenue}@_{\text{in}}(\text{B8}, 1) \geq 7,000,000$$

A similar process is applied to axiom node B7 for the conversion of attribute Salary, as shown in Example 1 of Section 2.

Let queue `candidates` contain the set of relation nodes whose `rules` are still to be pushed backwards. After defining one step constraint projection rules, we formalize the proposed backwards constraint propagation algorithm below.

**Algorithm 2.** *Constraint Propagation*

```

for (each element  $\alpha \in \text{producers}(\text{DW})$ ) {
  rules( $\alpha$ ) = rules(DW);
  candidates.push( $\alpha$ );
}
while (! empty(candidates)) {
  i = candidates.pop();
  set handlers(i);
  let axiom node  $\alpha$  output result table i;
  perform  $\Gamma(\alpha)$ ;
  mark each input port of  $\alpha$  as processed;
  for (each element  $j \in \text{producers}(\alpha)$ ) {
    if (all the output ports are processed) {
      candidates.push(j);
    }
  }
}
for (each relation node  $\alpha$ ) {
  minimize rules( $\alpha$ );
}

```

The last step of constraint propagation is to minimize the set of data quality rules at each relation node. There are two kinds of redundancies: 1) when computing `closure`, we may introduce more than necessary rules; 2) when `rules(resultTable( $\alpha$ ))` are pushed down to `producers( $\alpha$ )`, where they are verified, it is redundant to verify them again at `resultTable( $\alpha$ )`. Redundancies of the first kind can be removed by the known technique *transitive reduction*, which repeatedly removes a rule if it is logically implied by the rest of the closure. For the second kind of redundancies, if a data quality rule `r` in `rules(resultTable( $\alpha$ ))` is projected onto all `producers( $\alpha$ )` and not relaxed, it is removed.

*Example 5.* Consider Example [4](#), after constraint propagation, we get handlers and rules of each relation node. The handler `handler( $\alpha$ )` of relation node  $\alpha$  is to *keep records of data exceptions*. The rules of each relation node are:

```

rules(resultTable(C1)) = {pk(EId)}
rules(resultTable(A2)) = {Size  $\geq$  10}
  rules(DS1.employee) = {Salary  $\geq$  50,000, notNull(Salary), pk(EId)}
  rules(DS1.department) = {Revenue  $\geq$  1,000,000}
rules(resultTable(B2)) = {Size  $\geq$  10}
  rules(DS2.employee) = {Salary  $\geq$  350,000, notNull(Salary), pk(EId)}
  rules(DS2.department) = {Revenue  $\geq$  7,000,000}

```

The rules of all other relation nodes are  $\emptyset$ . The data quality rules are propagated down to their earliest verifiable nodes.

### 3.4 Correctness and Complexity

**Theorem 1.** *Suppose  $\Gamma(\alpha)$  projects data quality rule  $r$  onto table  $table_i$  as  $r_i$ . If  $resultTable(\alpha)$  satisfies  $r$ , then  $table_i$  satisfies  $r_i$ . If  $table_i$  does not satisfy  $r_i$ , then  $resultTable(\alpha)$  does not satisfy  $r$ .*

*Proof.* The basic idea is that: for an axiom node  $\alpha$ , in order that  $resultTable(\alpha)$  satisfies its rules, each table in  $producers(\alpha)$  should provide data that satisfies both the projections of  $rules(resultTable(\alpha))$  and the operation conditions, which are captured by  $mappings(\alpha)$ .

**Theorem 2.** *One step constraint projection  $\Gamma(\alpha)$  finishes in polynomial time of the number of rules.*

*Proof.* Step 1 finishes in constant time. Step 2 finishes in polynomial time. Step 3 finishes in constant time. Overall  $\Gamma(\alpha)$  finishes in polynomial time.

**Theorem 3.** *Constraint propagation finish in polynomial time.*

*Proof.* From Theorem 2,  $\Gamma(\alpha)$  finishes in polynomial time, and constraint propagation performs one step constraint projection only once for each constraint. Overall constraint propagation finishes in polynomial time.

## 4 Related Works

There have been some studies in data cleaning and data quality control in ETL workflows, both industrial and academic. We discuss related works below.

**ETL workflow design:** The ETL market is increasing rapidly and so are the powerful features of leading data integration tools [11]. SSIS from Microsoft supports data cleaning operations such as duplicate removal and exceptional data filtering. Oracle Data Integrator allows users to define filters at data sources. Some other ETL tools supports modular design and users can plug in their own data quality control module, such as Informatica Data Quality<sup>5</sup> allows business owners to configure data quality control processes. It supports data analysis, cleaning, matching and data monitoring. None of these system can perform ETL workflow analysis and target-end restriction verification automatically, which is the focus of this paper.

[12] presented a generic and customizable framework of ETL workflow designs. They proposed a metamodel to define ETL activities, and employed declarative database programming language, LDL, to define ETL activity semantics. We used this framework in design the ETL workflow of Example 1.

<sup>5</sup> [http://www.informatica.com/products\\_services/data\\_quality/Pages/index.aspx](http://www.informatica.com/products_services/data_quality/Pages/index.aspx)



**Query optimization:** ETL workflow design can be optimized using *predicate pushdown* techniques [13]. [3] generalized this techniques to handle aggregations and other constructs such as NOT EXISTS. Their methods move predicates bottom-up and then top-down in the query tree, and thereby enables moving predicates to applicable nodes across the whole query tree. [5] further generalized this techniques to query deductive databases. Deductive database programs are modeled as system graphs, and the evaluation of queries are viewed as data flows in system graphs. They impose filters on edges to verify predicates. [9] modeled the logical optimization of ETL workflows as state-space search problems. Two *unary* activities can be *swapped* if it does not affect results. This paper focuses propagating target-end restrictions backwards to data sources. Although the resulting data quality rules help to optimize the design of ETL workflows, it is not our main topic.

**Consistent Query Answering:** It seeks consistent and correct answers when there are data exceptions. Some techniques perform query rewriting to incorporate the given integrity constraints. [14] gave a comprehensive survey. [15] studied data integration with integrity constraints. They focused on global schema analysis with primary key and foreign key constraints, and presented techniques to effectively answer queries in this situation. This paper is dealing with a more complete set of integrity constraints, not just primary and foreign key analysis.

## 5 Conclusions

In this paper, an approach (BCP) is proposed that automatically analyzes ETL workflows and verifies target-end restriction as early as possible. It supports all relation algebra operators and user defined functions if they are monotonic and bijective, which covers most ETL data transformation functions. Our analysis reports the set of integrity constraints that a given ETL workflow must satisfy at every point in the workflow, and it helps to optimize ETL workflow design. Future work includes extending the approach to support more complete ETL operations, implementing it completely in our ETL tool OnceDI, and run more extensive benchmarks.

**Acknowledgements.** This work was partially supported by the National Grand Fundamental Research 973 Program of China under Grant No.2009CB320704; the National Natural Science Foundation of China under Grant No.90718033, 60773028; National High-Tech Research and Development Plan of China under Grant No.2007AA01Z149 and No.2007AA04Z148. We also thank the anonymous reviewers for their very insightful comments.

## References

1. Jarke, M., Lenzerini, M., Vassiliou, Y., Vassiliadis, P.: Fundamentals of Data Warehouses. Springer, New York (2001)
2. Rahm, E., Do, H.H.: Data cleaning: Problems and current approaches. IEEE Data Eng. Bull. 23(4), 3–13 (2000)

3. Levy, A.Y., Mumick, I.S., Sagiv, Y.: Query optimization by predicate move-around. In: VLDB 1994: Proceedings of the 20th International Conference on Very Large Data Bases, pp. 96–107. Morgan Kaufmann, San Francisco (1994)
4. Kifer, M., Lozinskii, E.L.: Filtering data flow in deductive databases. In: Atzeni, P., Ausiello, G. (eds.) ICDT 1986. LNCS, vol. 243, pp. 186–202. Springer, Heidelberg (1986)
5. Kifer, M., Lozinskii, E.L.: On compile-time query optimization in deductive databases by means of static filtering. *ACM Trans. Database Syst.* 15(3), 385–426 (1990)
6. Srivastava, D., Ramakrishnan, R.: Pushing constraint selections. In: PODS 1992: Proceedings of the eleventh ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems, pp. 301–315. ACM Press, New York (1992)
7. Marriott, K.G., Stuckey, P.J.: The 3 r’s of optimizing constraint logic programs: refinement, removal and reordering. In: POPL 1993: Proceedings of the 20th ACM SIGPLAN-SIGACT symposium on Principles of programming languages, pp. 334–344. ACM Press, New York (1993)
8. Jiangang, M., Dan, Y.: Data integration middleware. oncedi. 2.0. Technical Report (2004)
9. Simitsis, A., Vassiliadis, P., Sellis, T.: State-space optimization of etl workflows. *IEEE Trans. on Knowl. and Data Eng.* 17(10), 1404–1419 (2005)
10. Liu, J., Liang, S., Ye, D., Wei, J., Huang, T.: Etl workflow analysis and verification using backwards constraint propagation. Technical Report (2009)
11. Friedman, T., Beyer, M.A., Bitterer, A.: Magic quadrant for data integration tools (2008)
12. Vassiliadis, P., Simitsis, A., Georgantas, P., Terrovitis, M.: A framework for the design of etl scenarios. In: Eder, J., Missikoff, M. (eds.) CAiSE 2003. LNCS, vol. 2681, pp. 520–535. Springer, Heidelberg (2003)
13. Ullman, J.D.: Principles of Database and Knowledge-Base Systems, Volume 1 and 2. Computer Science Press, New York (1989)
14. Chomicki, J.: Consistent query answering: Five easy pieces. In: Proceedings of International Conference on Database Theory, pp. 68–76. Springer, Heidelberg (2007)
15. Cali, A., Calvanese, D., De Giacomo, G., Lenzerini, M.: Data integration under integrity constraints. In: Pidduck, A.B., Mylopoulos, J., Woo, C.C., Ozsu, M.T. (eds.) CAiSE 2002. LNCS, vol. 2348, pp. 262–279. Springer, Heidelberg (2002)

# The Declarative Approach to Business Process Execution: An Empirical Test

Barbara Weber<sup>1</sup>, Hajo A. Reijers<sup>2</sup>, Stefan Zugal<sup>1</sup>, and Werner Wild<sup>3</sup>

<sup>1</sup> Quality Engineering Research Group, University of Innsbruck, Austria  
{Barbara.Weber,Stefan.Zugal}@uibk.ac.at

<sup>2</sup> School of Industrial Engineering, Eindhoven Univ. of Technology, The Netherlands  
H.A.Reijers@tue.nl

<sup>3</sup> Evolution Consulting, Innsbruck, Austria  
Werner.Wild@evolution.at

**Abstract.** Declarative approaches have been proposed to counter the limited flexibility of the traditional imperative modeling paradigm, but little empirical insights are available into their actual strengths and usage. In particular, it is unclear whether end-users are really capable of adjusting a particular plan to execute a business process when using a declarative approach. Our paper addresses this knowledge gap by describing the design, execution, and results of a controlled experiment in which varying levels of constraints are imposed on the way a group of subjects can execute a process. The results suggest that our subjects can effectively deal with increased levels of constraints when relying on a declarative approach. This outcome supports the viability of this approach, justifying its further development and application.

## 1 Introduction

In today's dynamic business environment the economic success of an enterprise depends on its ability to react to various changes, like shifts in customers' attitudes or the introduction of new laws [1]. Process-aware information systems (PAISs) offer a promising perspective on shaping this capability, resulting in a growing interest to align information systems in a process-oriented way [2]. Yet, a critical success factor in applying a PAIS is that it can flexibly deal with process changes [3]. To address the need for flexible PAISs, competing paradigms enabling process changes and process flexibility have been developed, e.g., adaptive processes [4], case handling [5], declarative processes [6], and late binding and modeling [7] – for an overview see [8]. All of these approaches relax the strict separation of build-time (i.e., modeling or planning) and run-time (i.e., execution), which is typical for plan-driven planning approaches as realized in traditional workflow management systems (cf. Fig. 1). By closely interweaving planning and execution the above mentioned approaches allow for a more agile way of planning. In particular, users are empowered to defer decisions regarding the exact control-flow to run-time, when more information is available.

Depending on the concrete approach, planning and execution are interweaved to different degrees, resulting in different levels of decision deferral. The highest degree of decision deferral is fostered by *Late Composition* [8] (e.g., as enabled through a declarative approach) which describes activities that can be performed as well as constraints prohibiting undesired behavior. An example of a constraint in an aviation process would be that crew duty times cannot exceed a predefined threshold. A declarative approach, therefore, seems to be particularly promising to support highly dynamic processes [6,9]. The support for partial workflows [9] allowing users to defer decisions to run-time [8], the absence of over-specification [6], and more maneuvering room for end users [6] are all advantages commonly attributed to declarative processes. Although the benefits of declarative approaches seem rather evident, such approaches are not widely adopted yet in practice. In addition, there is a lack of empirical evidence on how well declarative approaches perform in real-world settings. In particular, it is unclear how well users can cope with the gained flexibility provided by declarative approaches, especially when processes and their context become rather complex.

The goal of this paper is to pick up on the demand for more empirical insights into the use of declarative approaches. Specifically, we aim to investigate how different levels of constraints may impede on the success that end users will have using a declarative approach for handling a particular business case (i.e., process instance). While it might be expected that the declarative approach will be effective to deal with business cases when few constraints apply, it is not clear whether end users are capable of translating a large number of constraints into effective updates on their initial plans. Because all constraints must be satisfied, one can argue that the sheer number of constraints will obscure from an end user's view what proper actions are still available. But proponents of declarative approaches to process planning and execution expect that end users will have little difficulty in doing this. In fact, this would exactly be one of its strengths [10,11,12], although this has not been established yet in an empirical setting.

To test whether end users can indeed deal with constraints, we conducted a controlled experiment with 41 students from both the Eindhoven Univ. of Technology and the Univ. of Innsbruck. In that test, we have been particularly interested in the potential impact of such constraints on the *effectiveness* of executing a process. This paper reports on the results of what we assume to be the first empirical work testing the declarative modeling paradigm. The structure of this paper is as follows. Section 2 provides backgrounds, while Section 3 describes our experimental framework. Section 4 covers the execution and results of our experiment. Finally, Section 5 discusses related work, followed by a conclusion.

## 2 Background

This section provides background information on planning approaches, presents different techniques for decision deferral, introduces declarative processes as well as the software we used for our experiment, the Alaska Simulator.

## 2.1 Planning Approaches

The flexibility of existing PAISs is significantly influenced by the underlying planning approach. In the following, we differentiate between *plan-driven*, *agile* and *chaotic* planning approaches, each of which assumes a completely different view on planning. Both plan-driven and agile approaches consider planning to be an essential activity, while chaotic approaches often lack a sufficient degree of planning and regard plans as unnecessary paperwork. In plan-driven approaches, the planning is usually done at the beginning and is not a repeated effort like in agile approaches (cf. Fig. 1). Although in both plan-driven and agile approaches the value of planning is appreciated, they have entirely different perceptions of a plan. In the former a plan is viewed as a schema for execution. Uncertainty is addressed by carefully planning everything upfront, which is appropriate for highly predictable processes. In contrast, agile approaches use a plan more like a guideline supporting decision making and recognize that in dynamic environments plans are quickly outdated and become inaccurate [13]. Decisions are made at the last responsible moment, when most information is available [1].

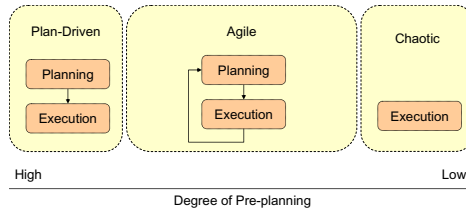


Fig. 1. Different Planning Approaches

## 2.2 Dealing with Uncertainty by Deferring Decisions

When examining existing PAISs, four different patterns for deferring decisions to the last responsible moment can be identified [8]. The *Multi-Instance Activity* pattern offers the least amount of freedom during run-time. It allows users to defer the decision on how often a specific activity should be executed to run-time, while the activity itself needs to be predefined. The *Late Binding* pattern offers slightly more flexibility by deferring the selection of the implementation of a particular process activity to run-time. Prior to execution, only a placeholder activity has to be provided; the concrete implementation is selected during run-time from a set of predefined fragments. The *Late Modeling* pattern goes one step beyond this and allows for the modeling of selected parts of the process schema at run-time. Prior to execution, a placeholder activity as well as a set of modeling constraints has to be defined. Most flexibility is offered by the *Late Composition* pattern, which allows users to compose process fragments from the process repository on the fly. No predefined model is required, as the business case can be created in an ad-hoc way by selecting activities from a repository,

while respecting all existing constraints. Consequently, *Late Composition* allows users to freely switch between process modeling and execution. The focus of this paper will be on *Late Composition* as enabled by declarative processes, which allows for the maximum level of flexibility.

### 2.3 Declarative Processes

There is a long tradition of modeling business processes in an *imperative* way. Process modeling languages supporting this paradigm, like BPMN, BPEL and UML Activity Diagrams, are widely used. Recently, *declarative* approaches have received increased interest and suggest a fundamentally different way of describing business processes [14]. While imperative models specify exactly how things have to be done, declarative approaches only focus on the logic that governs the interplay of actions in the process by describing (1) the activities that can be performed, as well as (2) constraints prohibiting undesired behavior. Imperative models take an ‘inside-to-outside’ approach by requiring all execution alternatives to be explicitly specified in the model. Declarative models, in turn, take an ‘outside-to-inside’ approach: constraints implicitly specify execution alternatives as all alternatives have to satisfy the constraints [15]. Adding additional constraints means discarding some execution alternatives (cf. Fig. 2). This results in a coarse up-front specification of a process, which can then be refined iteratively during run-time. Typical constraints described in literature can be roughly divided into three classes (e.g., [7,14]): constraints restricting the *selection* of activities (e.g., the minimum or maximum occurrence of activities, mutual exclusion, co-requisite), the *ordering* of activities (e.g., pre-requisite or response constraints) or the use of *resources* (e.g., execution time of activities, time difference between activities, budget, etc.).

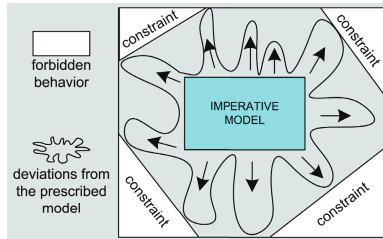


Fig. 2. Declarative Approaches to Process Modeling [11]

### 2.4 The Alaska Simulator

To foster the comparison of different approaches for process flexibility the Alaska Simulator [1] has been implemented, which takes a *journey* as metaphor for a *business process*. The similarities being exploited here are that regardless whether

<sup>1</sup> Developed at the University of Innsbruck, <http://www.alaskasimulator.org>

a journey or a business process is executed, various steps must be planned and carried out, even if the actual execution of those steps may be different from what is initially foreseen. Furthermore, journey planning is an attractive context for many people to become engaged in, which highly improves their willingness to use the system for experimental purposes.

In the Alaska Simulator, a plan can either be created in a plan-driven or in a more agile way (cf. Section 2.1). The Alaska Simulator also provides support for *Late Composition* as enabled by declarative processes. The actions of a journey, like travel activities, routes and overnight stays correspond to activities in the business process. For optimizing the execution of a particular business case, information about benefits (i.e., business value), cost and duration of activities is essential. Incomplete information prior to execution is a characteristic of both journeys and highly flexible business processes and is best handled by waiting until more information is available (cf. Section 2.2). The business value of a travel activity is not predefined, but can vary depending on the weather conditions during the journey. Thereby, the degree of variation depends on the activity's reliability. When composing a concrete business case, different constraints like *selection constraints*, *ordering constraints* or *resource constraints* have to be considered (cf. Section 2.3), similar constraints also exist when planning a journey (e.g., mandatory activities, dependencies between activities).

Fig. 3 depicts the graphical user interface of the Alaska Simulator. Users can compose their individual travel plan by dragging available actions from the Available Actions View (3) onto the travel itinerary (1). Actions are usually only available at a particular location on the map (4). Existing constraints are

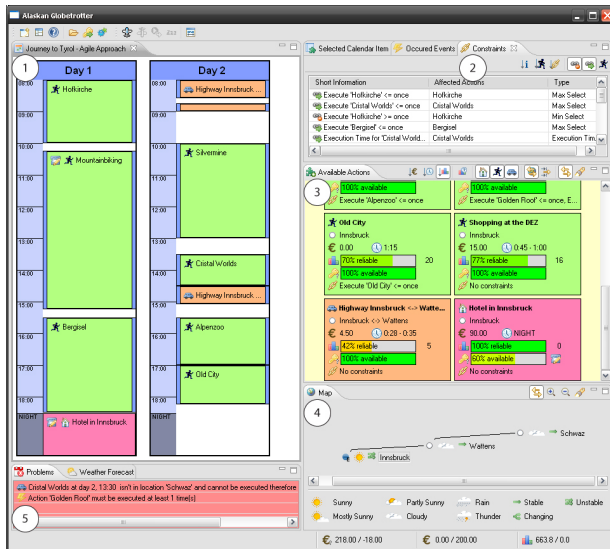


Fig. 3. Screenshot of the Alaska Simulator

displayed in the Constraint View (2) and have to be considered when composing a concrete journey. After each user action, the journey is validated and the user is informed about any constraint violations and plan inconsistencies(5).

### 3 Experiment Definition and Planning

The main goal of our experiment is to evaluate the outcome of end-users executing a business process that is guided by a declarative approach under varying numbers of constraints. This section explains the setup of our experiment (cf. Section 3.1) and introduces its specific design (cf. Section 3.2). Factors threatening the validity of the experiment results, as well as potential mitigations, are discussed in Section 3.3. We follow the recommendations given in [16] in setting up and describing an experiment throughout this section.

#### 3.1 Experiment Setup

This section describes the subjects, objects and selected variables of our experiment, introduces our hypothesis and presents the instrumentation and data collection procedure.

**Subjects:** Subjects are 25 students of a graduate course on Business Process Management at Eindhoven University of Technology and 16 students of a similar course at the University of Innsbruck.

**Objects:** The objects to be modeled and executed are two declarative process models representing two journeys that must be planned and executed (referred to as Configuration Alaska and Configuration California). These configurations comprise activities to be executed, constraints that restrict their execution as well as their ordering, and events that might occur during run-time. For example, between a **Diving** activity and a **Flightseeing** activity there must be a rest period of two days to prevent aeroembolism. For each of the configurations, two variants are created: A and B, differing in the number of constraints only. To be specific, variant A contains a true subset of the constraints of variant B. An overview of the different variant characteristics is given in Fig. 4. Note that in addition to the constraints mentioned, in both variants all travel activities but one could be executed at most once, e.g., not visiting the Golden Gate bridge twice. To cover a broad and representative set of constraints we ensured that both configurations comprise constraints belonging to all three typical constraint classes (i.e., selection, ordering and resource constraints).

**Factor and Factor Levels:** The number of constraints in the model is the considered factor with levels “low” and “high”. Variant A of a configuration corresponds to factor level “low” and variant B to factor level “high”.

**Response Variable:** The response variable is the *business value* the subjects achieve when executing the journey. Thereby, the maximum achievable business



Variants	Actions	Constraints	Detailed Constraints
Alaska A	26	2	1 budget constraint, 1 end-location constraint
Alaska B	26	12	1 budget constraint, 1 end-location constraint, 3 mandatory actions, 3 execution time constraints, 1 constraint requiring a minimum delay between two actions, 1 pre-requisite constraint, 1 mutual exclusion constraint, 1 constraint requiring A to be followed by n times B and a final C
California A	22	2	1 budget constraint, 1 end-location constraint
California B	22	10	1 budget constraint, 1 end-location constraint, 4 mandatory actions, 2 execution time constraints, 2 constraints requiring a minimum delay between two actions

Fig. 4. Characteristics of the Configuration Variants

value for each activity is known upfront, whereas the value actually gained depends on the weather conditions during the journey (in the upfront phase only statistical data about the weather and the degree to which the business value can vary are known<sup>2</sup>). To ensure comparability of results, weather conditions are the same for each subject.

**Hypothesis Formulation:** In our experiment we investigate whether adding additional constraints has an influence on the response variable *business value*. Based on this the following hypothesis is derived:

- **Null hypothesis  $H_{0,1}$ :** There is no significant difference in the business values of configurations with a low and a high level of constraints.
- **Alternative hypothesis  $H_{1,1}$ :** There is a significant difference in the business values of configurations with a low and a high level of constraints.

**Instrumentation:** To precisely measure our response variable we have implemented a logging function in the Alaska Simulator, which automatically records the required data.

**Data Analysis Procedure:** For data analysis well-established statistical methods and standard metrics are applied (cf. Section 4.2 for details).

### 3.2 Experiment Design

Literature on software experiments provides various design guidelines for setting up an experiment (e.g., [17]). Considering these design criteria, we accomplish our experiment as a *balanced single factor* experiment with *repeated measurement* (cf. Fig. 5). Our experiment is denoted as single factor experiment, since it investigates the effects of one *factor* (i.e., number of constraints in a configuration) on a common *response variable* (e.g., business value). Our experiment design also allows us to analyze variations of a factor called *factor levels* (i.e., configurations with few and with many constraints). The response variable is

<sup>2</sup> A detailed description on how to calculate the business value can be found in the Alaska Simulator’s documentation (<http://www.alaskasimulator.org>).

determined when the participants of the experiment (i.e., *subjects*) apply the factor or factor levels to an *object* (i.e., Configuration Alaska or Configuration California). We denote our experiment as *balanced*, as all factor levels are used by all participants of the experiment, i.e., each subject has to plan a journey with both few and many constraints. This enables *repeated measurements* and thus the collection of more precise data, since every subject generates data for every treated factor level. Generally, repeated measurements can be realized in different ways. We use a frequently applied variant which is based on two subsequent runs (cf. Fig. 5). During the first run, half of the subjects (referred to as Group 1) apply few constraints to the treated object, while the other half (referred to as Group 2) uses many constraints. After having completed the first run, the second run begins. During this second run each subject applies the factor level not treated so far to the object. In order to avoid learning effects we use two different configurations for the two runs. Each subject was randomly assigned to either Group 1 or Group 2.

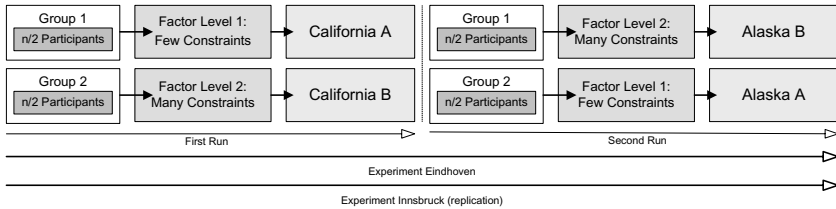


Fig. 5. Design of our Balanced Single Factor Experiment

### 3.3 Risk Analysis and Mitigations

When accomplishing experimental research related risks have to be taken into account as well. Generally, factors exist that threaten the internal validity (“Are the claims we made about our measurements correct?”), as well as the external validity (“Can the claims we made be generalized?”) of an experiment. In our context, threats to *internal validity* are:

**People:** The students participating in our experiment differ in their skills and productivity. In particular, different experience levels in terms of process modeling, planning and scheduling may have an influence on the students’ performance. This issue can only be balanced by conducting the experiment with a sufficiently large and representative set of students and to perform replications of the experiment. The number of 41 students seems sufficiently large to achieve such a balance. Furthermore, the experiment was replicated in the setting of the Innsbruck students after its initial conduct in the Eindhoven setting.

Besides, there are threats to the *external validity* of experiment results:

**Students instead of professionals:** Involving students instead of professionals can be critical. However, [18] has shown before that the results of student

experiments are transferable and can provide valuable insights into an analyzed problem domain. Moreover, for a journey planning and execution exercise as in this experiment, which requires no knowledge of a business domain, graduate students will probably have a similar ability as professionals.

**Investigation of tools instead of concepts:** In our experiment, the Alaska Simulator was used as a representative for a tool providing modeling and execution support for declarative business processes. Obviously, the achieved results to some degree depend on the quality of the used tool. Problems in understanding the tool as well as poor user support might influence the results. To mitigate this risk, considerable effort was put into designing an intuitive and easy to use user interface. As the Alaska Simulator was used at the 2008 Austrian Research Night by around 300 lay users of all ages (i.e., 7 to 70) without any considerable problems, tool understandability is not a major issue. However, it has to be recognized that our results cannot be automatically transferred to tools with less elaborated user support.

**Choice of object:** To mitigate the risk that the two variants of a configuration (i.e., few and many constraints) do not differ enough in terms of complexity, we performed pre-tests with several subjects (who did not further participate in the experiment) and repeatedly refined the configurations based on their feedback. We involved test persons both with in-depth knowledge of the Alaska Simulator and novices.

## 4 Performing the Experiment

By now, the set-up of the experiment has been explained. Section 4.1 describes the preparation and execution of the experiment. Then, the analysis and interpretation of the experiment data is presented in Section 4.2. Finally, in Section 4.3, a discussion of the experiment results is provided.

### 4.1 Experimental Operation

**Experimental Preparation:** As part of the set-up of the intended experiment, we prepared two travel configurations, i.e., Configuration California and Configuration Alaska. For each of the configurations, two variants were created: A (few constraints) and B (many constraints). To ensure that each configuration is correct and can be executed in the available amount of time, we involved several persons with different backgrounds in its pre-tests. Based on their feedback, the configurations were refined in several iterations. Finally, we compiled a “starter kit” for each participant, consisting of a screencast explaining the main features of the simulator and a test configuration to casually explore.

**Experimental Execution:** The experiment was conducted at two distinct, subsequent events. The first event took place during October 2008 in Eindhoven, a replication was performed three weeks later in Innsbruck. Prior to the start of

the experiment, all students had to attend an introductory lecture to obtain an overview on declarative processes. During this lecture, we further informed them about the goals and rules of the experiment. Afterwards, each student received his/her “starter kit”. Having watched the screencast and having gone through the test configuration, the actual experiment started. In the first run, the students had to model and execute Configuration California. Half of them were assigned to the A variant of the configuration (containing few constraints), the other half to the more complex B variant. For all students, a familiarization phase of 25 minutes was available, in which they could explore the configuration and gather relevant domain knowledge. After this phase, the students had another 20 minutes to plan and execute the journey exactly once, with the goal to optimize the business value of the journey. In the second run of the experiment, Configuration Alaska had to be executed and the variants for the groups were switched, i.e., the students who had worked on an A variant had to work on the B variant and vice versa. Again, the actual execution of the configuration to obtain a high business value was preceded by a familiarization phase of 25 minutes.

**Data Validation:** After having conducted the experiment, the logged data was analyzed. We discarded the data of one Eindhoven student as the journey could not be properly executed due to a bug in the software that occurred only for this student. In addition, we did not consider the data of one Innsbruck student who performed the wrong variant of Configuration Alaska (A instead of B) in the second run. Finally, data provided by 25 Eindhoven students and 16 Innsbruck students were used in our data analysis.

### 4.2 Data Analysis

In this section, we describe the analysis of the gathered data and interpret the obtained results.

**Descriptive Analysis:** Based on raw data from the log of the Alaska Simulator we calculated some descriptive statistics for the response variable *business value* (cf. Fig. 6). By analyzing Fig. 6 one can observe the following:

	N	Minimum	Maximum	Mean	Standard Deviation
Experiment Eindhoven					
California A	13	0	4778,59	3477,39	1298,44
California B	12	0	4956,12	2278,75	2083,20
Total	25	0	4956,12	2902,04	1790,41
Alaska A	12	0	7254,13	5147,90	1887,27
Alaska B	13	0	7580,83	5117,30	2441,38
Total	25	0	7580,83	5131,99	2147,77
Experiment Innsbruck					
California A	8	0	6473,07	4563,34	1997,49
California B	8	2356,34	5816,09	4571,13	1179,95
Total	16	0	6473,07	4567,26	1584,84
Alaska A	8	4966,54	8328,13	6620,14	1107,89
Alaska B	8	5650,97	9043,25	7409,52	1035,02
Total	16	4966,54	9043,25	7014,83	1113,05

Fig. 6. Descriptive Statistics for Response Variable ‘Business Value’

- For the Eindhoven sample, the mean business value for Configuration California A is higher than that for Configuration California B.
- For the Eindhoven sample, the mean business values for Configurations Alaska A and Alaska B are rather similar.
- For the Innsbruck sample the mean business value for Configuration California A is rather similar to that for Configuration California B.
- For the Innsbruck sample the mean business value for Configuration Alaska B is higher than that for Configuration Alaska A.

The question is whether the noted differences are statistically significant.

**Data Plausibility:** We analyzed data plausibility based on *box-whisker-plot diagrams*, which visualize the distribution of a sample and particularly show outliers. The diagram takes the form of a box that spans the distance between the 25% quartile and the 75% quartile (the so called *interquartile range – IQR*) surrounding the median which splits the box into two parts. The “whiskers” are straight lines extending from the ends of the box, the length of a whisker is at most 1.5 times the interquartile range. All results outside the whiskers can be considered as outliers. Fig. 7A shows the outliers for the Eindhoven sample. For all configurations except for California B outliers exist with respect to the obtained business values. As can be seen in Fig. 7B, for the Innsbruck sample only outliers exist for Configuration California A, while all data from the remaining configurations lie within the boxed areas. At a first glance, the number of outliers may appear rather high. However, this result can be explained by the fact that journeys which were finished with constraint violations were assigned a business value of 0. When these values are not considered, only one outlier remains for the Eindhoven sample and no outliers for the Innsbruck sample. Thus, plausible data distributions seem to be in effect.

**Testing for Differences in Business Value:** To test for differences in business values, we compare the business values obtained by the subjects using

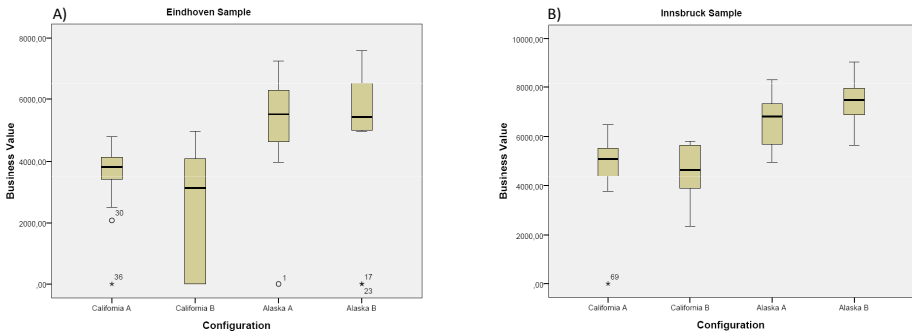


Fig. 7. Data Distribution (Box-Whisker-Plot Diagrams)

Configuration California A in the first run with those using California B. Furthermore, we compare the business values for the Alaska variants in the second run.

**Eindhoven Experiment:** Data for Configuration California A from the Eindhoven sample is not normally distributed, with standardized skewness and standardized kurtosis values outside the normal range. Therefore, a t-test cannot be applied to determine any differences in this case and the non-parametric Mann-Whitney test [19] – which can be thought of as comparing the medians of the distributions – is applied. With an obtained P-value of 0.125 ( $> 0.05$ ), hypothesis  $H_{0,1}$  cannot be rejected at a confidence level of 95%. Data for Configurations Alaska A and B for the Eindhoven sample is also not normally distributed. Thus, the non-parametric Mann-Whitney test is applied, resulting in a P-value of 0.643 ( $> 0.05$ ). Like for the first test run, hypothesis  $H_{0,1}$  cannot be rejected. So, for both Configuration Alaska and Configuration California there is no statistically significant difference between the business values obtained within their simple (A) and complex (B) variants.

**Innsbruck Experiment:** Data for California A from the Innsbruck sample is not normally distributed, thus a t-test is not applicable to determine any differences. Again, we apply the non-parametric Mann-Whitney test and obtain a P-value of 0.636 ( $> 0.05$ ). Based on this result, hypothesis  $H_{0,1}$  cannot be rejected at a confidence level of 95%. Data for Configurations Alaska A and Alaska B of the Innsbruck sample are both normally distributed and have the same variance. Therefore, we apply the t-test and obtain a P-value of 0.163 ( $> 0.05$ ). Based on this, hypothesis  $H_{0,1}$  cannot be rejected at a confidence level of 95%. The results of the replication confirm the results of the Eindhoven experiment. Again, for both the Alaska and the California configuration there is no statistically significant difference between the business values obtained within their simple (A) and complex (B) variants.

**Overall Conclusion:** Considering the results, strong support emerges for the conclusion that hypothesis  $H_{0,1}$  cannot be rejected, i.e., no significant difference can be found between obtained business values in configurations with a low and a high level of constraints.

### 4.3 Discussion of Results

The major finding from our data analysis is that through our experiment no statistically significant differences can be found in the outcome of planning and executing a journey when we considerably vary the level of constraints that end-users have to take into account. This is supportive of the argument that end-users can use agile planning as enabled by a declarative approach to effectively deal with substantially varying levels of constraints.

The most plausible alternative explanation for the absence of any differences is that the configuration variants were not sufficiently distinguishable. We would

like to recall, however, that we refined the various configurations and their variants until the involved subjects in our pre-tests perceived a notable and considerable difference in difficulty between them (see Section 3.3).

Another alternative explanation, and a more technical one, is that the range of potential business values does not have enough spread to identify any differences. Yet, the coefficient of variation across the obtained business values for the various samples has ranged from 14% to 91%. This potentially offers sufficient variation to identify statistical differences, if there is any.

Considering that the alternative explanations do not appear stronger than the explanation we propose, i.e., the suitability and robustness of agile planning, the question that needs to be raised is to what extent our findings can be generalized.

First of all, planning a journey in the Alaska Simulator is not quite the same as collectively executing a business process. Furthermore, business processes can take on widely different forms and the required mix of support and flexibility may vary likewise. At the same time, the journey metaphor seems not to be a major threat to the validity of our results as similarities outweigh existing differences and the configurations used in the experiment range well beyond the size of toy examples, as they typically cover 22 to 26 actions to be planned and executed. It is clear to us that additional work is required to extend the scope and content of the experiment matter towards even more realistic settings. As is often the case, raising the level of external validity may be difficult without affecting the internal validity. In other words, it is questionable how the size and duration of a similar experiment could be extended to better reflect a realistic scenario without suffering from bad responses. Therefore, alternative empirical evaluations, such as gaming or case studies, may be more attractive instruments for further empirical research in this area.

Secondly, the Alaska Simulator provides its users means for creating a rough plan which is then incrementally validated. As such, users are well supported to become aware of constraint violations and resolve them. From a manual analysis of the planning actions (which were all logged during the experiment) we can establish that every subject created such a rough plan at the beginning. Based on some earlier experiences using the Alaska Simulator where subjects were unsuccessful when not following this approach, we suspect that the incremental validation of constraint violations and the ability to create a rough plan is essential for a good performance. So, it is unlikely that our results can be replicated by using a declarative system that does not provide this support, e.g., DECLARE [6], which is a clear restriction on generalizing our results.

## 5 Related Work

Most existing work about flexibly dealing with exceptions, changes, and uncertainty in the context of PAISs and related technologies is strongly *design-centered*, i.e., aiming at the development of tools, techniques, and methodologies. For overviews and discussions of these approaches, see [8,20,21].

Only few empirical investigations exist that aim to establish the suitability of the various proposed artifacts. In [22], the results of a controlled experiment comparing a traditional workflow management system and case-handling are described. The systems are compared with respect to their associated implementation and maintenance efforts. In turn, the impact of workflow technology on PAIS development and PAIS maintenance is investigated in [23]. However, these existing works primarily focus on traditional workflow technology, while this paper is the first one investigating the declarative modeling paradigm. Other empirical works with respect to PAISs mainly deal with establishing its contribution to business performance improvement, e.g. [24,25], and the way end-users appreciate such technologies, e.g. [26,27].

Worth mentioning here is a stream of research that relates to so-called *change patterns* [8]. It provides a framework for the qualitative comparison of existing flexibility approaches. This paper is complementary to that work by providing empirical findings in addition to the qualitative data presented earlier.

## 6 Summary and Outlook

Although the advantages attributed to declarative processes are manifold (e.g., support for partial workflows allowing users to defer decisions to run-time, the absence of over-specification as well as more room for end users to maneuver), their practical application is still limited. Furthermore, strengths and weaknesses of the declarative modeling paradigm are not yet well understood. In particular, it is unclear how well users can cope with the flexibility gained, especially when processes and their context become rather complex as the number of constraints increases. This paper reports on the results from what is presumably the first controlled experiment on the declarative process modeling paradigm. Our results indicate end-users can effectively use agile planning as enabled by a declarative approach over a considerable spectrum of constraints. However, incremental validation of constraint violations and the ability to create a rough plan seem essential ingredients for a good performance.

Our future work will aim at further investigating the practical suitability of declarative processes, in particular their maintainability. Although declarative workflows allow for easy changes of both business cases and process models by modifying constraints [6], it is notoriously difficult to determine which constraints have to be modified and then to test the newly adapted set of constraints. Acceptance testing, which is well established in software engineering, is known to facilitate communication regarding the intent of the developed software. To ensure that constraint changes are performed as intended, we consider transferring ideas from acceptance testing to the modeling of constraints.

In addition to this future line of research, we aim to investigate different techniques for improving understandability of declarative process models (e.g., through modularization or the definition of higher-level constraints) and we plan to validate our approach through further experiments.



Finally, we believe that there is a need for a benchmark to compare different declarative approaches. After all, it can be noted that different declarative approaches vary in respect to the extent that they hide procedural information from the modeler or the end user. It can be expected that results as we have reported in this paper may vary along that spectrum.

To conclude this paper, we wish to express our hope that the presented results will serve as an incentive for others to continue the promising development and application of declarative approaches for the planning and execution of business processes.

**Acknowledgements.** We thank G. Molina, M. Netjes, M. Song, J. Pinggera and T. Schrettl for their much appreciated help in preparing and executing the experiment.

## References

1. Poppendieck, M., Poppendieck, T.: *Implementing Lean Software Development: From Concept to Cash*. Addison-Wesley, Reading (2006)
2. Weske, M.: *Business Process Management: Concepts, Methods, Technology*. Springer, Heidelberg (2007)
3. Lenz, R., Reichert, M.: IT Support for Healthcare Processes - Premises, Challenges, Perspectives. *Data and Knowledge Engineering*, 39–58 (2007)
4. Reichert, M., Dadam, P.: ADEPT<sub>flex</sub> – Supporting Dynamic Changes of Workflows Without Losing Control. *JGIS* 10, 93–129 (1998)
5. Van der Aalst, W., Weske, M., Grünbauer, D.: Case handling: A new paradigm for business process support. *Data and Knowledge Engineering* 53, 129–162 (2005)
6. 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)
7. Sadiq, S., Sadiq, W., Orłowska, M.: A Framework for Constraint Specification and Validation in Flexible Workflows. *Information Systems* 30, 349–378 (2005)
8. Weber, B., Reichert, M., Rinderle-Ma, S.: Change patterns and change support features -enhancing flexibility in process-aware information systems. *Data and Knowledge Engineering*, 438–466 (2008)
9. Wainer, J., Bezerra, F., Barthelmess, P.: Tucupi: a flexible workflow system based on overridable constraints. In: Handschuh, H., Hasan, M.A. (eds.) *SAC 2004*. LNCS, vol. 3357, pp. 498–502. Springer, Heidelberg (2004)
10. Hull, R., et al.: Declarative workflows that support easy modification and dynamic browsing. *Software Engineering Notes* 24, 69–78 (1999)
11. Pesic, M., van der Aalst, W.: A declarative approach for flexible business processes. In: Eder, J., Dustdar, S. (eds.) *BPM Workshops 2006*. LNCS, vol. 4103, pp. 169–180. Springer, Heidelberg (2006)
12. Mulyar, N., Pesic, M., van der Aalst, W., Peleg, M.: Declarative and procedural approaches for modelling clinical guidelines: Addressing flexibility issues. In: *BPM 2007 International Workshops*, pp. 335–364 (2008)
13. Cohn, M.: *Agile Estimating and Planning*. Prentice Hall Professional, Englewood Cliffs (2006)

14. van der Aalst, W., Pesic, M.: DecSerFlow: Towards a Truly Declarative Service Flow Language. Technical report, BPMcenter.org (2006)
15. Pesic, M.: Constraint-Based Workflow Management Systems: Shifting Control to Users. PhD thesis, Eindhoven University of Technology (2008), <http://alexandria.tue.nl/extra2/200811543.pdf>
16. Wohlin, C., Runeson, R., Halst, M., Ohlsson, M., Regnell, B., Wesslen, A.: Experimentation in Software Engineering: an Introduction. Kluwer, Dordrecht (2000)
17. Juristo, N., Moreno, A.M.: Basics of Software Engineering Experimentation. Springer, Heidelberg (2001)
18. Runeson, P.: Using students as experiment subjects - an analysis on graduate and freshmen student data. In: Proc. EASE 2003, pp. 95–102 (2003)
19. Siegel, S.: Nonparametric statistics for the behavioral sciences. McGraw-Hill, New York (1956)
20. Kammer, P., Bolcer, G., Taylor, R., Hitomi, A., Bergman, M.: Techniques for Supporting Dynamic and Adaptive Workflow. Computer Supported Cooperative Work (CSCW) 9(3), 269–292 (2000)
21. Reijers, H., Rigter, J., van der Aalst, W.: The case handling case. International Journal of Cooperative Information Systems 12, 365–391 (2003)
22. Mutschler, B., Weber, B., Reichert, M.: Workflow management versus case handling - results from a controlled software experiment. In: Proc. SAC 2008, pp. 82–89 (2008)
23. Kleiner, N.: Supporting usage-centered workflow design: Why and how?. In: Desel, J., Pernici, B., Weske, M. (eds.) BPM 2004. LNCS, vol. 3080, pp. 227–243. Springer, Heidelberg (2004)
24. Oba, M., Onoda, S., Komoda, N.: Evaluating the quantitative effects of workflow systems based on real case. In: Proc. HICSS 2000 (2000)
25. Reijers, H., van der Aalst, W.: The effectiveness of workflow management systems: Predictions and lessons learned. International Journal of Information Management 25, 458–472 (2005)
26. Bowers, J., Button, G., Sharrock, W.: Workflow from within and without: technology and cooperative work on the print industry shopfloor. In: Proc. CSCW 1995, pp. 51–66. Kluwer Academic Publishers, Dordrecht (1995)
27. Poelmans, S.: Workarounds and distributed viscosity in a workflow system: a case study. ACM SIGGROUP Bulletin 20, 11–12 (1999)

# Configurable Process Models: Experiences from a Municipality Case Study

Florian Gottschalk<sup>1</sup>, Teun A.C. Wagemakers<sup>1</sup>, Monique H. Jansen-Vullers<sup>1</sup>,  
Wil M.P. van der Aalst<sup>1,2</sup>, and Marcello La Rosa<sup>2</sup>

<sup>1</sup> Eindhoven University of Technology, The Netherlands  
{f.gottschalk,m.h.jansen-vullers,w.m.p.v.d.aalst}@tue.nl,  
teun.wagemakers@pallas-athena.com

<sup>2</sup> Queensland University of Technology, Brisbane, Australia  
m.larosa@qut.edu.au

**Abstract.** Configurable process models integrate different variants of a business process into a single model. Through configuration users of such models can then combine the variants to derive a process model optimally fitting their individual needs. While techniques for such models were suggested in previous research, this paper presents a case study in which these techniques were extensively tested on a real-world scenario. We gathered information from four Dutch municipalities on registration processes executed on a daily basis. For each process we identified variations among municipalities and integrated them into a single, configurable process model, which can be executed in the YAWL workflow environment. We then evaluated the approach through interviews with organizations that support municipalities in organizing and executing their processes. The paper reports on both the feedback of the interviewed partners and our own observations during the model creation.

**Keywords:** Business Process Models, Configuration, YAWL, Registration Process, Questionnaires, Case Study.

## 1 Introduction

Many processes in public administration are driven by legislation, e.g. the process of renewing a drivers licence is constrained by law. Therefore, the processes executed in the administration of municipalities are extensively regulated. Although legislation is establishing the important steps, some freedom is left regarding the concrete implementation of such processes. Hence, municipalities can still adapt their processes to local needs and preferences, e.g. depending on the size of the municipality, or on the services provided along with these processes.

*Configurable process models* were developed to align the variation options of widely standardized processes with small variations like the ones executed by municipalities. Further, they enable software providers to support the execution of these process variations through their service-oriented software. For this, configurable process models integrate several process variants into a single process model. To adapt this integrated model to individual needs, a configurable

process model can be configured allowing an organization to disable all the unnecessary process parts. In this way, an organization can derive a process model that fits its individual needs without actually performing any process modeling activities. If the configurable model is defined as a workflow specification, the resulting models can then be executed through a workflow engine [5].

As disabling unnecessary process parts of the workflow definitions still requires insights into the process modeling notation, this can be steered through a questionnaire with domain-related questions. In this way, domain experts without such skills can configure and derive executable workflow specifications fitting their particular needs [7,8].

To evaluate the concept of configurable process models in practice, we performed a case study with four municipalities of different sizes in the Netherlands. Using the YAWL workflow notation and its configuration extension [5], we created configurable process models for four registration processes that are executed within each of the selected municipalities on a daily basis. The four configurable models incorporate all the variations in the execution of these processes among the municipalities as well as the suggestions of a reference model for these processes, i.e.  $5 \times 4 = 20$  processes were used as input for creating these models. Afterwards, we evaluated the practical usefulness of the resulting models through focus group interviews with software providers, consultants, and the municipalities themselves. During these interviews the stakeholders could derive individual process models for these four processes. To test whether the resulting models conform to what they intended by answering the questionnaires, they could execute the resulting process definitions using the YAWL system [1].

The results of the case study are presented in this paper which is structured as follows. Section [2] will provide some background information on configurable process models and configurable YAWL, as well as on how these models can be configured through questionnaires. Next, Section [3] first depicts how we created the configurable models for the municipality processes before summarizing our practical experiences during the model creation phase. Section [4] provides details on the interviews we performed with the stakeholders, as well as the conclusions from these interviews. The paper ends with a brief overview on similar case studies in Section [5], and draws overall conclusions in Section [6].

## 2 Background

Configurable extensions have been suggested for several process modeling notations. In this case study we used configurable YAWL, i.e. an extension of the YAWL notation aiming at configuration [5,7]. Using a workflow notation for implementing the configurable model comes with the advantage that we can execute the configured process models in the corresponding workflow engine and thus test and demonstrate the practical feasibility of the suggested methodology even to users unfamiliar with process modeling. In addition, the steering

---

<sup>1</sup> <http://www.yawlfoundation.org>

of the configuration through a questionnaire aims at providing such domain experts with the ability to derive configurations for process models. In this way, the adaptation of a business process model for later execution can be achieved without extensive modeling skills.

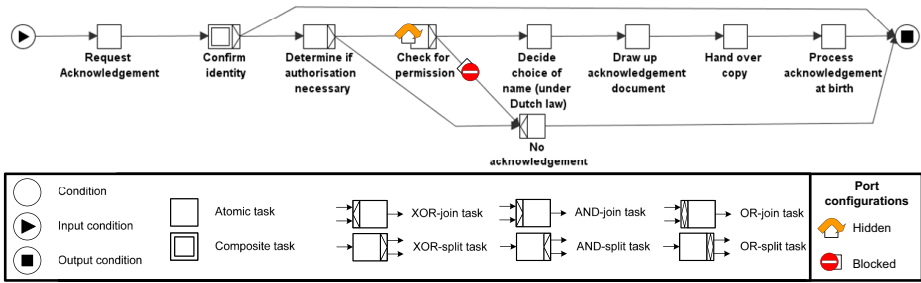
## 2.1 Configurable YAWL

YAWL is a process modeling notation and workflow environment based on Petri nets but extended with powerful features for cancelation, OR-joins, etc. It has been developed with the aim to provide a notation with formal semantics that supports all desired workflow patterns [1]. The YAWL system is open-source and supports the execution and work distribution of workflows depicted in such models even in production environments. Thus, although originally developed as a proof of concept, the YAWL system can be used for practical applications.

Figure 1 depicts a simple YAWL model for the process executed by municipalities when a man registers that he will become the father of a not-yet-born child although he is not married to the mother.<sup>2</sup> In this model tasks are depicted as rectangles while circles represent conditions like the initial and final condition in this example. Conditions mark the states between tasks but can be omitted for simplicity (like in the example). Composite tasks enable the hierarchical specification of sub-processes while split and join types of tasks allow the specification of how the process should proceed in case a task splits or joins the process's control flow. For this, YAWL distinguishes an XOR-split (as in the example in Figure 1) allowing the triggering of only one of the subsequent paths, an AND-split requiring the triggering of all subsequent paths, and an OR-split requiring the triggering of at least one subsequent path but allowing also for path combinations. Similarly, a task with an XOR-join can be executed as soon as one of its incoming paths is triggered, an AND-join requires that all incoming arcs are triggered, and a task with an OR-join allows for the execution of the task as soon as no further incoming paths can potentially be triggered at any future point in time (see [1] for further details).

This routing behavior can be restricted by *process configuration*. For this purpose, *input ports* are assigned to each task depicting how the task can be triggered and *output ports* are assigned to depict which paths can be triggered after the completion of the task. A task with an XOR-join can be triggered via each of its incoming paths. Thus, it has a dedicated input port for each of these paths. Tasks with AND-joins and OR-joins can only be executed if all paths (that can potentially be triggered) are triggered, i.e. there is only one way these tasks can be triggered and thus only one input port. A task with an XOR-split has an output port for each subsequent path as each of these paths can be triggered individually while a task with an AND-split has only one output port as all subsequent tasks must always be triggered. A task with an OR-split can trigger a subset of the outgoing paths, i.e. in this case a separate output port exists for each of these combinations.

<sup>2</sup> Note that this process is specific for the Netherlands and constrained by Dutch law.



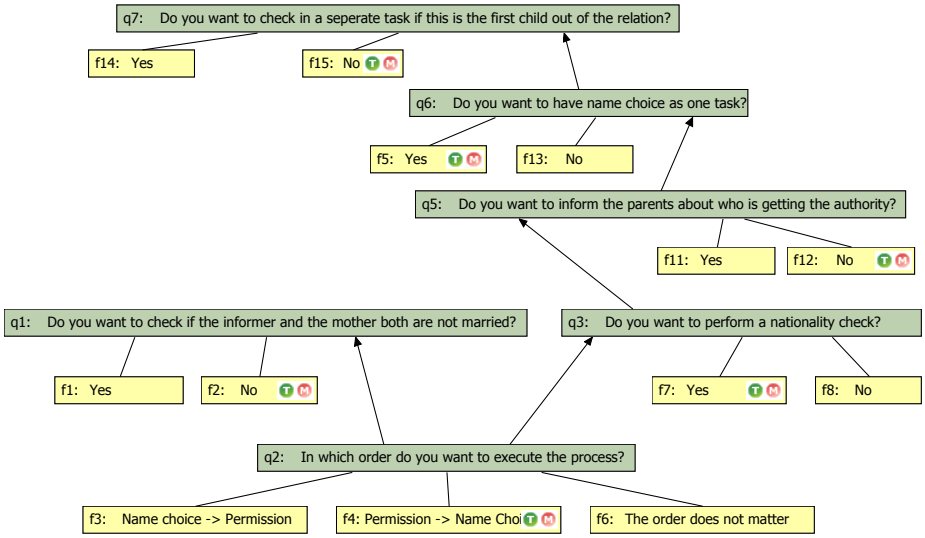
**Fig. 1.** A YAWL process model for acknowledging an unborn child. The input port of *check permission* is configured as hidden and one output port is blocked.

The process flow can be restricted at these ports. A *blocked* port prevents the process flows through it, i.e. a blocked input port prevents the triggering of the task through the port while a blocked output port prevents that the corresponding output paths can be triggered. In the model in Figure 1, we blocked the output port from *Check for permission* to *No acknowledgement*. Thus, the task *Check for permission* must always be followed by the task *Decide choice of name (under Dutch law)* as the path to the task *No acknowledgement* can no longer be triggered. Input ports can not only be blocked but also be configured as *hidden*. Similarly, the subsequent task can then not be triggered through this port anymore. However, in this case the process flow is not completely blocked, but only the execution of the corresponding task is skipped. The process execution continues afterwards. In Figure 2 the input port of the task *Check for permission* is hidden. Thus, the execution of this task is skipped which also explains why we blocked one of the task's output ports: the configuration results in skipping the check. Hence, it can no longer fail and the process must continue normally. Further details on configurable YAWL can be found in [5].

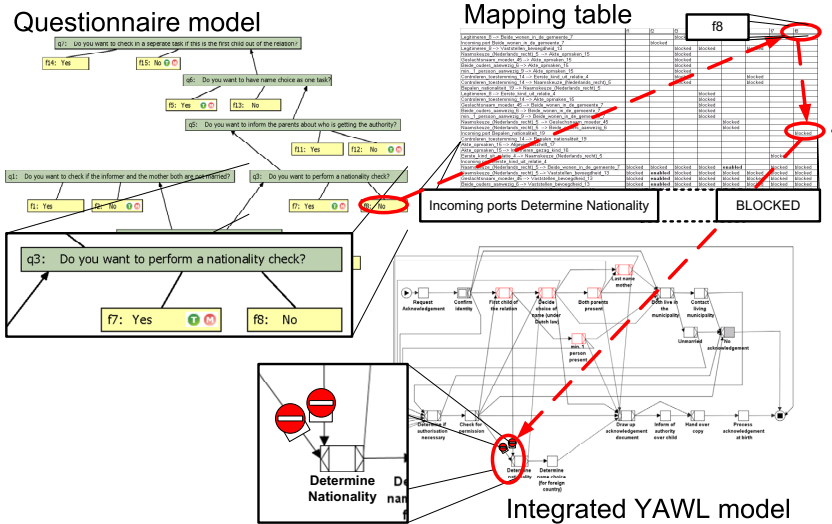
As we can observe from this example, the configurations of ports are often not independent from each other but rather driven by more general domain-related aspects. It is therefore suggested to steer the configuration through questions on these domain-related aspects as is discussed next.

## 2.2 Steering Process Configuration through Questionnaires

In principle, the variability of the domain can be depicted independently of the process flow by means of a set of domain *facts* that form the space of possible answers to a set of *questions*. A domain fact is a boolean variable representing a feature of the domain, e.g. *Perform a check of the nationality*, that can be enabled or disabled. Questions can group domain facts according to their content, so that all the facts of the same question can be set at once by answering the question. Interdependencies between questions can specify a partial order in which the questions should be posed to the user. Figure 2 depicts such a questionnaire model for the various options in the process of acknowledging an unborn child.



**Fig. 2.** The questionnaire model addressing the various options in performing the process of acknowledging an unborn child



**Fig. 3.** The setting of a domain fact through answering a question leads to the selection of a particular port configuration via the so-called mapping table

Each configuration of a port in the process model can then depend on such domain facts. For example, the input ports of the task in which the nationality check is performed must be set to allowed when the corresponding domain fact is set to *true* while it must be hidden or blocked when the domain fact is set

to *false*. Such a port configuration might also be dependent on a combination of answers, i.e. domain facts. For this, the facts can be combined in propositional logic expressions that capture their interplay. It is then important to make sure that a single port will never have two configuration values at the same time (e.g. blocked and hidden) which can be achieved through corresponding, additional constraints. These can then also imply that certain answers given in the questionnaire automatically define the answers to further questions.

Figure 3 depicts and summarizes the steering of process configuration through questionnaires. Further details on the approach can be found in [78].

### 3 Creating Configurable Process Models

In the first project phase we created configurable process models for four registration processes. These processes are executed on a daily basis in each of the municipalities. In this way, we evaluated the feasibility of configurable process models in public administration. The steps taken during the creation of the models are explained in the first part of this section. Afterwards, we summarize the challenges we were confronted with during the creation of such models and how we addressed them.

#### 3.1 Building the Models

In total we created configurable process models for the following four registration processes:

- *Acknowledging an unborn child*: This process is executed when a man wants to register that he will be the father of a child still to be born while he is not married to his pregnant partner.
- *Registering a newborn*: This process is executed by the municipality to register a newborn child and handing out a birth certificate.
- *Marriage*: This process includes all steps necessary before a couple can get married in a Dutch municipality.
- *Decease*: This process is executed when a person deceases to provide the relatives with the documentation necessary to bury the deceased.

Reference process models for these processes are available from the Nederlandse Vereniging Voor Burgerzaken (NVVB<sup>3</sup>), i.e. the Dutch association for services to the public. These models describe a single “best-practice” version of how the particular process should be executed. While these models are available in several notations, we used the notation of the business process modeling tool Protos. Protos is very popular among Dutch municipalities: these reference models are used by over 100 Dutch municipalities (mainly for auditing purposes).

To detect the variations of the processes in the daily practice we visited four municipalities in the Netherlands. We selected the municipalities such that they

<sup>3</sup> <http://www.nvnb.nl>



vary in the size of their population (between 26.000 and 201.000 inhabitants) and such that they use software from different providers to support the process execution. Without confronting the process owners of the selected municipalities with the reference models, we asked them to explain how they execute the various processes. We then used again Protos to create a separate process model for each process in each municipality. During this phase, some of the municipalities provided us with process models which they created to document their processes. In these cases, we based our models on the models which were provided by them. We only made modifications where it became clear from our visits to the particular municipality that a process model did not reflect what was actually happening. To make sure that we correctly depicted the processes, we asked the process owners at the end of this phase to validate the models. Figure 4 shows the four Protos models we derived from the four municipalities for the process of acknowledging an unborn child. While the control flow of these four processes is similar, the number of steps taken as well as the concrete order of executing tasks varies among municipalities.

For each of the four selected business processes we then identified all the differences among the five process variants (the reference model plus the models of the four municipalities) by comparing them with each other. Based on this information, we created for each business process a single Protos model that incorporates all the variations from the five input models as ordinary runtime choices. The integrated model derived from the reference model and the four process variants for acknowledging an unborn child shown in Figure 4 is shown in Figure 5. Note that out of the four business processes we analyzed in this case study, the process of acknowledging an unborn child is the simplest, i.e. the three other combined process models include both more tasks and more arcs.

To be able to configure and execute these models, we then switched to a workflow environment that supports both the configuration and execution of Protos models. In particular we chose YAWL here as our ideas on process configuration [5,8] are implemented in this environment. The translation from Protos to YAWL was done manually as we not only translated the pure control flow from the Protos models, but also implemented the data upon which the process relies and which is only available in a descriptive way in the Protos models. In this way, we created the basis to route cases through the process model according to the data collected during the process execution. That means, the resulting YAWL models are fully executable in the YAWL workflow engine. The YAWL model for the acknowledgement of an unborn child is shown in Figure 6a.

The resulting four YAWL models integrating all the variations of the processes were of course far too complex to be used and configured by the stakeholders of the municipalities. Thus, we also created a questionnaire for each of the four business processes as explained in Section 2.2. In the questionnaires we addressed each variation possibility at a particular stage of the process by at least one question. The questionnaire model for the process of acknowledging an unborn child was already shown in Figure 2. The answers to the questions were then mapped to allowing, hiding, or blocking the process flow through various ports. In this

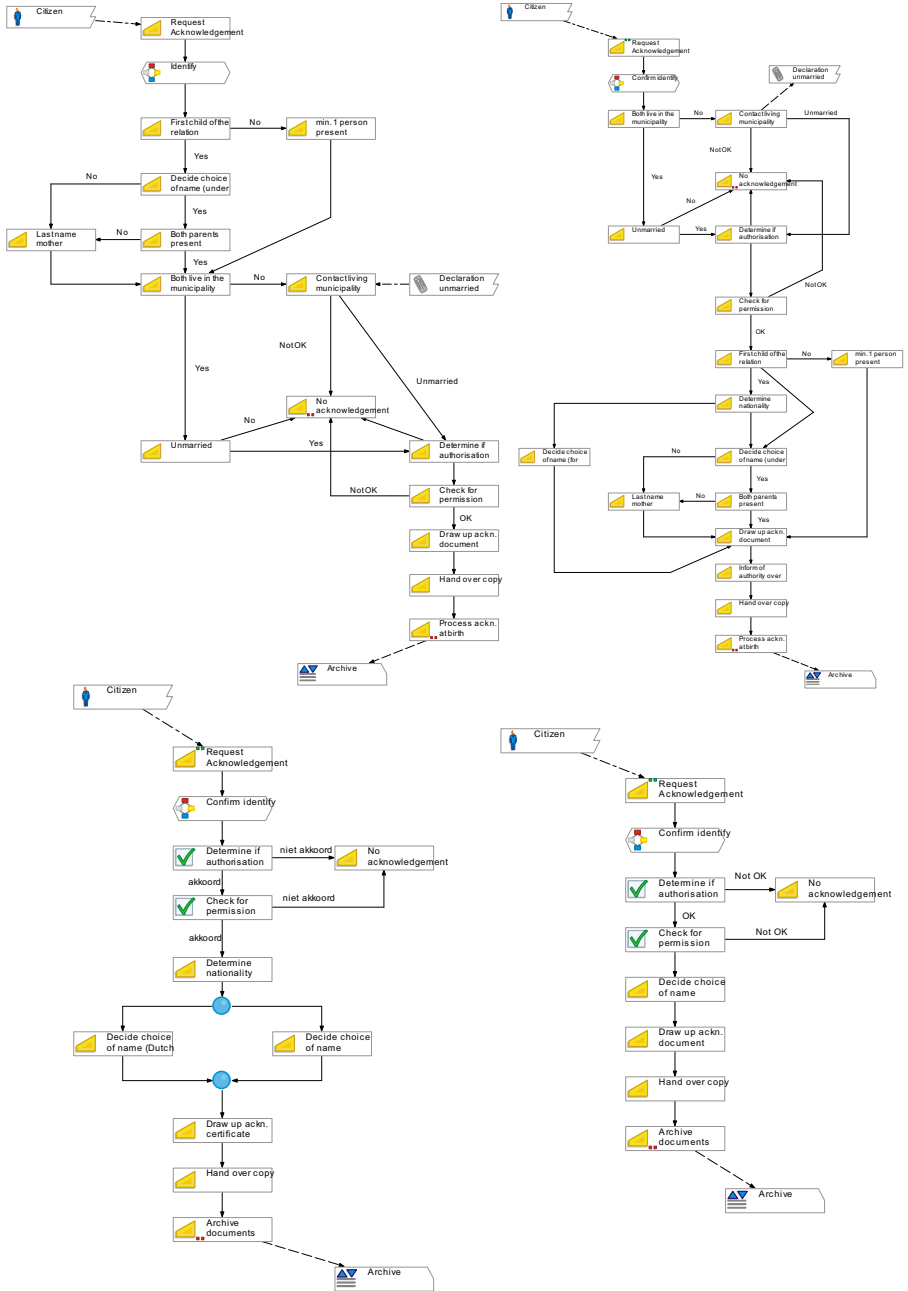
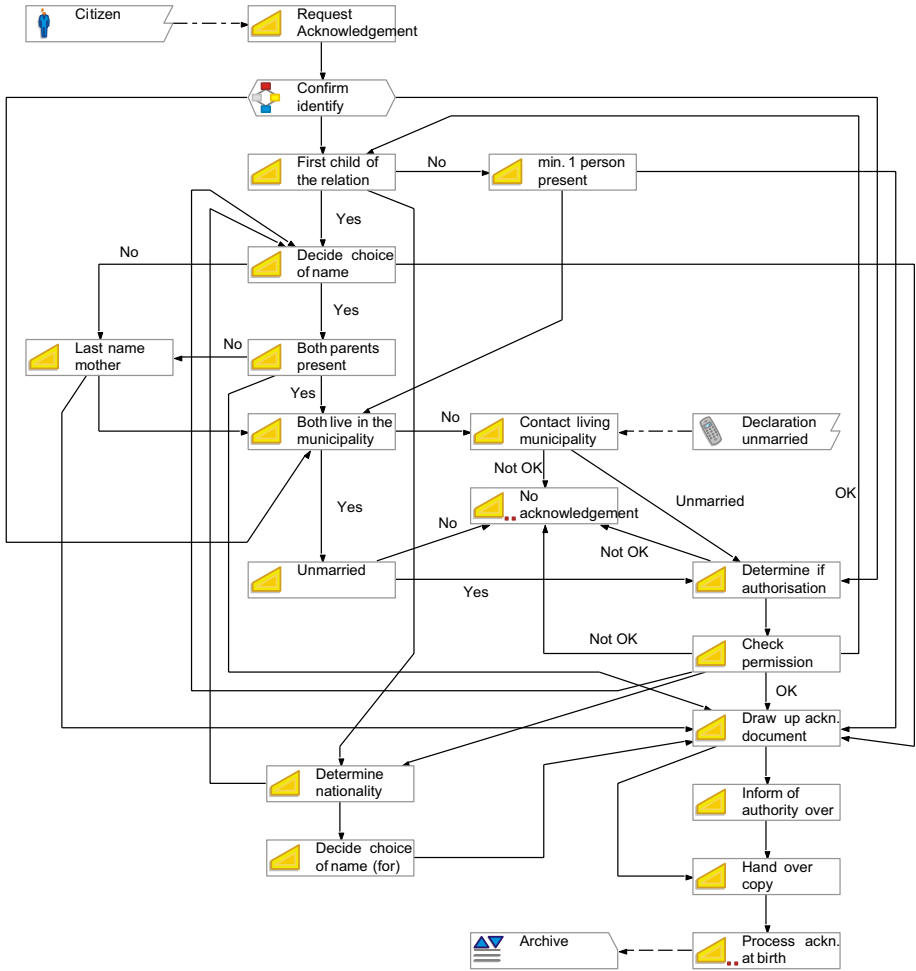


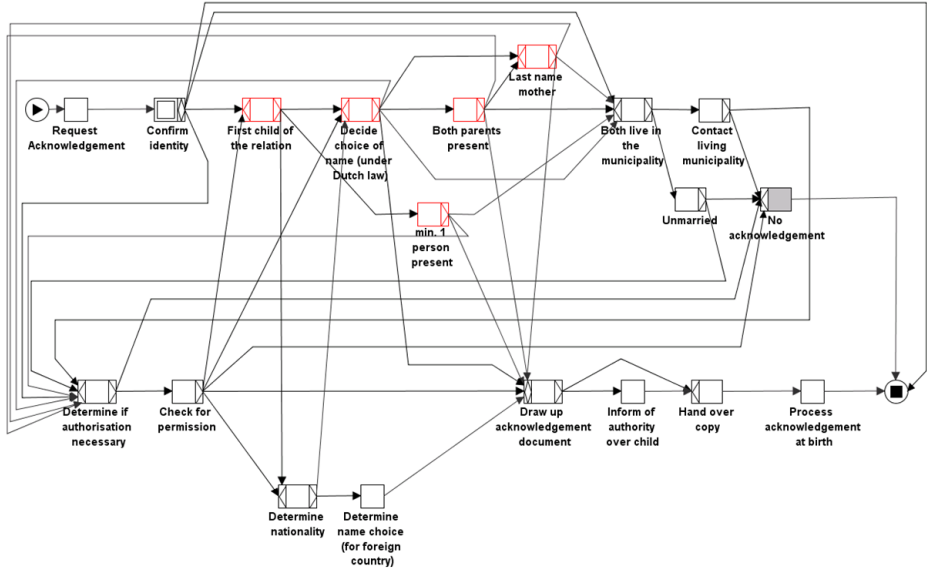
Fig. 4. The different process variants of how municipalities perform the acknowledgement of an unborn child



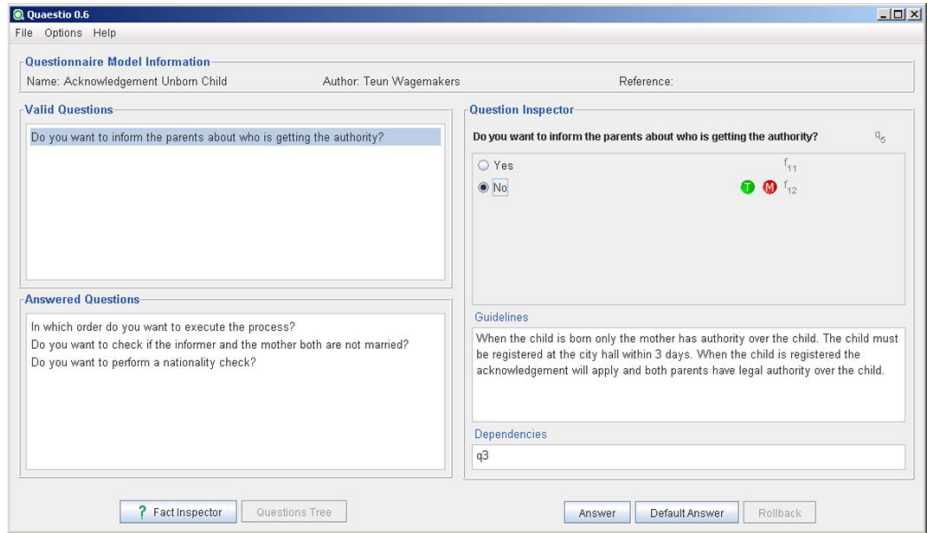
**Fig. 5.** All variants for acknowledging an unborn child integrated into a single Protos model

way, the configuration of the process model integrating the five process variants can be done by the stakeholders through simply answering the questionnaire (see Figure 6b). There is no need for the stakeholders to understand the implications of blocking or hiding certain ports. In fact, they do not even need to be confronted with the integrated process models as the configuration decisions resulting from the answers to the questionnaire can be applied automatically to this model. For example, the process model we showed in Figure 1 is in fact derived from the integrated model in Figure 6a using the answers given by one of the involved municipalities. In our approach, the stakeholders only receive these individually relevant models which are also directly executable using the YAWL

a)



b)



**Fig. 6.** The model from Figure 5 translated into YAWL (a) and the corresponding questionnaire in the Questio tool (b) which allows user to configure the model through the given answers

workflow engine. Then, the users of the model will be filling out forms generated based on the information of the configured process model. Thus, these users do not see but benefit from the configured model.

### 3.2 Observations

First of all, it should be noted that for all four business processes we were able to create integrated process models and questionnaires that allow users to derive an individual model. For each process and each municipality we were able to generate a model equivalent to the original Protos model by answering the questionnaire and applying the resulting configuration to the YAWL model. This illustrates that it is possible to integrate several process variants such that all desired individual variants can be derived from it.

Still, we had to master several challenges during the creation of the configurable models. While deriving the individual process variants was straight forward, the first challenges arose when integrating the different variants into a single process model as matching identical tasks among the variants was often only possible after comparing the exact task descriptions. Moreover, during the manual compilation of the integrated model some paths, i.e. process flows, of the individual models were easily overlooked, and thus not incorporated into the integrated model. Only by carefully “re-playing” the processes of the individual models in the combined models these forgotten arcs were discovered.

Due to the extensive support of control flow patterns, translating the control flow from the Protos models to YAWL models was easy. Tricky was however the implementation of the data perspective for determining the precise runtime routing of cases through the integrated process model. This was especially the case when a choice between various options was introduced in the integrated model while in fact there is no such run-time decision in any of the municipalities. The variation is thus a pure configuration decision based on the difference between the municipalities. For example, this applies for the task *Confirm identity* in Figure 6 which uses an OR-split to branch into four outgoing paths. The decision, which combination of paths should be triggered after the completion, is partly a run-time decision and partly a configuration decision. During run-time it is decided if the identification was successful or not. If not, the process completes immediately. However, the decision which combinations of the remaining three arcs are triggered in case the identification was successful is already a configuration decision (it might be desired to transform this into a run-time decision, but this was not the case in any of the involved municipalities). A correct definition of the process flow details in such situations requires the implementation of a “default” decision as well as a very good anticipation of the implications when this default decision has to change due to a configuration decision.

Questions in the questionnaire abstract from the control flow of the process and usually address larger process parts. Thus, the interdependencies between the answers that can be given in the questionnaire are not always obvious or immediately derivable from the process’s control flow. Hence, ordering of questions and the definition of constraints between the answers turned out to be challenging and required a good anticipation of the desired impact of the configuration decisions which becomes more difficult the more complex the model is.

This phase was mainly performed by one core project member and took, including his familiarization with the used techniques, approximately six months.

## 4 Evaluation of the Approach

To get insights into the practical applicability of the models we derived, we performed an additional analysis using three approximately two-hour-long focus group interviews with one to three employees of the following three organizations:

- Pallas Athena as the supplier of Protos which is actively used by over 250 of the in total 441 Dutch municipalities,
- PinkRoccade Local Government who provides a software to execute municipality processes used by more than 50% of the Dutch municipalities, and

Interview partner	Potential applications and advantages (+) as well as concerns (-)
Pallas Athena	<p>(+) Configurable Process models would have been useful for the development of a “one point of contact” workflow product for municipalities developed based on a new law that requires municipalities to re-structure the customer interaction of their business processes</p> <p>(+) Potential applications in highly regulated, publicly documented and accessible, or non-core business processes like HR processes.</p> <p>(-) The integrated model must be complete. Is this possible and how can this information be derived from existing processes?</p>
PinkRoccade Local Government	<p>(+) Questionnaire answers can be linked to other configurable elements, like the configuration of software screens and windows as well as data fields.</p> <p>(+) Configuration through questionnaires enables software providers to create applications that prevent that users can fail during the process configuration.</p> <p>(+) A user sees in the questionnaire the configuration freedom she has rather than the limitations the configuration is subject to.</p> <p>(+) Clients often ask for software adaptation and modifications for a better support of their desired business processes which is currently expensive due to the need for external consultants. Currently, this often results into workarounds.</p> <p>(-) A configuration of the resources that are involved in a process is not possible.</p>
Consultancy Firm	<p>(+) Best-practice reference models are often not sufficient: there is no single best-practice.</p> <p>(+) It would have been useful in a world-wide role-out of new business processes where it was a headquarter policy that 80 % of the processes needed to remain conform to the global process while it was allowed to deviate by 20 % to make the process compliant to local regulations.</p> <p>(+) In some industries production processes are so standardized that the technique might here even be applicable to core processes.</p> <p>(-) The creation of such models seems to require big efforts, sponsoring for this might be difficult to find.</p> <p>(-) The identification of variations between processes is difficult, i.e. tools are necessary for this.</p>

**Fig. 7.** The main comments of the interviewed stakeholders

- a world-wide operating consultancy firm who adapts their own reference process models during process implementations for their clients.

All interview partners were first given a presentation on the techniques we used during this project as well as on details of how we created the four configurable models and their questionnaires. Afterwards, the interview partners had the opportunity to derive their own executable process models through answering the questionnaires. The models resulting from the answers given in the questionnaire during the interview were immediately presented to them. Not all the interview partners were domain experts for the given processes. Thus, it was possible for them to ask questions on the implications of the various possible configuration decisions in the questionnaire.

Subsequently, we triggered a discussion with the interview partners focussing on potential practical needs for adaptable process models, on the feasibility of creating such configurable models in real-life environments, and on the practical usefulness of applying such configurable models. Key results from these interviews are summarized in Figure 7. In general, we can summarize that all interview partners immediately saw a potential value of the technique of configurable process models for past or current projects. The steering of the actual process configuration is seen as a useful tool to assist end users, but even without this support, direct process configuration might prove to be beneficial in various projects where process adaptation is necessary. The main concerns raised by the interview partners were the efforts necessary to create questionnaires and establishing the links between the potential answers and all the ports as well as the incorporation of resource assignments to tasks during the configuration process.

## 5 Related Work

The case study reported on in this paper uses our earlier work on process configuration [5,7,8]. Similar techniques for adapting process models were suggested by Becker et al. [3,4]. Their approach links adaptation parameters and their possible values to model elements to indicate which sections of the model are relevant or not to a specific application scenario. Thus, a user can configure a process model by setting parameters, i.e. the process model does not need to be consulted. Compared to the approach used here, the approach of Becker et al. is applied to Event-driven Process Chains instead of YAWL, i.e. to a notation that is mostly used for process visualization (like the Protos models we created) and not for the enactment of these processes. Moreover, it lacks steering of the parameter setting through an interactive questionnaire.

The use of questionnaires to guide the configuration of process models is inspired by similar configuration processes for software applications. For example, the CML2 language, designed to capture configuration processes for the Linux kernel, guides the configuration process through a structured set of questions that lead to a given symbol being given a value [10]. Also, in CML2 the validity of these values can be ensured by constraints. More generally, variability of

large software systems has been studied in the field of Software Product Line Engineering (SPLE) [9].

Algermissen et al. performed a case study with municipalities to identify best-practice in public administration [2]. Similar to our approach, they initially visited a number of municipalities to observe and depict their business processes. Different from our approach, they do not focus on providing a model with various configuration options, but rather aim at deriving a single, “ideal” process model from these variants. Thus, their approach is similar to the one taken by the NVVB, whose best-practice recommendation we incorporated in our models.

Karow et al. provide guidelines specifically for the construction of reference models in public administration [6]. While our goal here was to test the feasibility and identify the opportunities of using configurable process models in a reference modeling context, we would need to address such guidelines more rigorous if we want to extend our work to providing a complete reference model in the future.

Best-practice reference models have been investigated in several other case studies. For example, Thomas et al. developed a reference model for event management [13], and Scheer designed a reference model for industrial enterprises [11]. A case study on developing a business process reference model for the screen business was performed by Seidel et al. [12]. Also template repositories as provided by vendors of BPM solutions like the ones of SAP and IBM can be considered as such best-practice reference models.

## 6 Conclusions

In this case study, we developed configurable process models for four business processes of municipalities based on information from four different municipalities and a corresponding reference model. Afterwards, we performed expert interviews with various stakeholders about the potential use of these models and the underlying techniques.

During the case study, the suggested techniques proved to be suitable for the intended purposes: we achieved our goal to be able to derive all the initial, individual models of the various municipalities as well as further model variants from the integrated models by answering simple questionnaires. Despite that, the creation of the configurable models required significant efforts, modeling experience, and domain knowledge. Thus, the simplified adaptation of process models is at the expense of a complex creation of the configurable model.

It was obvious during the case study that many issues that arose during the model creation could be improved or even avoided by further tool support, e.g. ensuring consistent identifiers or automatically identifying and integrating process variations. Thus, all interview partners were also interested in techniques that can help here. In general, they all saw potential value for themselves in the technique, which they stressed by mentioning current or past projects where configurable process models could have provided additional benefits. But the interviewees also made clear that process configuration should not be restricted to the control flow perspective of business processes, but should also be integrated with the resource and data perspectives to provide a strong and universal configuration tool.



**Acknowledgements.** We would like to thank the NVVB, Pallas Athena, and PinkRocade Local Government as well as the municipalities, consultants, and software developers involved in this project for their input and feedback.

## References

1. van der Aalst, W.M.P., ter Hofstede, A.H.M.: YAWL: Yet Another Workflow Language. *Information Systems* 30(4), 245–275 (2005)
2. Algermissen, L., Delfmann, P., Niehaves, B.: Experiences in Process-oriented Reorganisation through Reference Modelling in Public Administrations — The Case Study Regio@KomM. In: *Proceedings of the 13th European Conference on Information Systems (ECIS)*, Regensburg (2005)
3. Becker, J., Delfmann, P., Dreiling, A., Knackstedt, R., Kuropka, D.: Configurative Process Modeling – Outlining an Approach to increased Business Process Model Usability. In: *Proceedings of the 15th IRMA International Conference*, New Orleans. Gabler (2004)
4. Becker, J., Delfmann, P., Knackstedt, R.: Adaptive Reference Modelling: Integrating Configurative and Generic Adaptation Techniques for Information Models. In: Becker, J., Delfmann, P. (eds.) *Reference Modeling. Efficient Information Systems Design Through Reuse of Information Models*, pp. 27–58. Springer, Heidelberg (2007)
5. Gottschalk, F., van der Aalst, W.M.P., Jansen-Vullers, M.H., La Rosa, M.: Configurable Workflow Models. *International Journal of Cooperative Information Systems (IJCIS)* 17(2), 177–221 (2008)
6. Karow, M., Pfeiffer, D., Räckers, M.: Empirical-Based Construction of Reference Models in Public Administrations. In: *Proceedings of the Multikonferenz Wirtschaftsinformatik 2008. Referenzmodellierung*, pp. 1613–1624 (2008)
7. La Rosa, M., Gottschalk, F., Dumas, M., van der Aalst, W.M.P.: Linking Domain Models and Process Models for Reference Model Configuration. In: ter Hofstede, A.H.M., Benatallah, B., Paik, H.-Y. (eds.) *BPM Workshops 2007. LNCS*, vol. 4928, pp. 417–430. Springer, Heidelberg (2008)
8. La Rosa, M., van der Aalst, W.M.P., Dumas, M., ter Hofstede, A.H.M.: Questionnaire-based Variability Modeling for System Configuration. *Software and Systems Modeling* (forthcoming) (2008)
9. Pohl, K., Böckle, G., van der Linden, F.: *Software Product-line Engineering – Foundations, Principles and Techniques*. Springer, Berlin (2005)
10. Raymond, E.S.: *The CML2 Language* (2000), <http://catb.org/esr/cml2/cml2-paper.html>
11. Scheer, A.-W.: *Business Process Engineering, Reference Models for Industrial Enterprises*. Springer, Berlin (1994)
12. Seidel, S., Rosemann, M., ter Hofstede, A.H.M., Bradford, L.: Developing a Business Process Reference Model for the Screen Business - A Design Science Research Case Study. In: *Proceedings of the 17th Australasian Conference on Information Systems (ACIS 2006)*, Adelaide, Australia (2006)
13. Thomas, O., Hermes, B., Loos, P.: Towards a Reference Process Model for Event Management. In: ter Hofstede, A.H.M., Benatallah, B., Paik, H.-Y. (eds.) *BPM Workshops 2007. LNCS*, vol. 4928, pp. 443–454. Springer, Heidelberg (2008)

# Business Process Modeling: Current Issues and Future Challenges

Marta Indulska<sup>1</sup>, Jan Recker<sup>2</sup>, Michael Rosemann<sup>2</sup>, and Peter Green<sup>1</sup>

<sup>1</sup> UQ Business School, The University of Queensland, St Lucia, QLD, 4072, Australia  
{m.indulska, p.green}@business.uq.edu.au

<sup>2</sup> Business Process Management Group, Faculty of Science and Technology,  
Queensland University of Technology, Brisbane, QLD, 4000, Australia  
{j.recker, m.rosemann}@qut.edu.au

**Abstract.** Business process modeling has undoubtedly emerged as a popular and relevant practice in Information Systems. Despite being an actively researched field, anecdotal evidence and experiences suggest that the focus of the research community is not always well aligned with the needs of industry. The main aim of this paper is, accordingly, to explore the current issues and the future challenges in business process modeling, as perceived by three key stakeholder groups (academics, practitioners, and tool vendors). We present the results of a global Delphi study with these three groups of stakeholders, and discuss the findings and their implications for research and practice. Our findings suggest that the critical areas of concern are standardization of modeling approaches, identification of the value proposition of business process modeling, and model-driven process execution. These areas are also expected to persist as business process modeling roadblocks in the future.

**Keywords:** business process modeling, Delphi study, issues, challenges.

## 1 Introduction

Business process modeling – an approach to graphically display the way organizations conduct their business processes – has emerged as an important and relevant domain of conceptual modeling [1]. It is considered a key instrument for the analysis and design of process-aware Information Systems [2], organizational documentation and re-engineering [3], and the design of service-oriented architectures [4]. To that end, business process models typically describe in a graphical way at least the activities, events/states, and control flow logic that constitute a business process. Additionally, the models may also include information regarding the involved data, organizational and IT resources, and potentially other artifacts such as external stakeholders, goals, risks and performance metrics (e.g., [5]).

While much academic literature is dedicated to various topics related to business process modeling, indications exist that practitioners struggle with various process modeling aspects and find limited support from academic literature in guiding their efforts. Overall, there is a lack of empirical studies in business process modeling that guide future research directions [6]. In line with this observation, the main goal of the

study reported in this paper is to identify and explore *the core issues* with business process modeling as they are perceived by the three main stakeholder groups, i.e. practitioners, vendors and academics. In addition to the identification of the current issues, we aim to explore the upcoming issues, i.e., the process modeling *challenges* that are expected to be problematic in the future. In reaching such a goal, we are able to present those items that are perceived as most critical for the further development of process modeling. Accordingly, our study is based on the following two main research questions:

- R1. What are the current business process modeling issues?; and
- R2. What are the challenges in business process modeling likely to be in 5 years time?

We choose to explore the two research questions in a Delphi study setting with three separate groups of participants, *viz.*, *academics* in the business process modeling domain, *business process modeling practitioners*, and *vendors* of business process modeling tool and consultancy offerings. Our objective is to identify and prioritize the most significant issues and future challenges of business process modeling, reach consensus about these, and compare the issues and challenges across the three distinct stakeholder groups.

We proceed as follows. Sections 2 and 3 detail the research design and methodology, the selection of the three groups of participants, and the specifics of the three rounds of the Delphi study. Section 4 presents a discussion of the top issues in business process modeling. Similarly, Section 5 presents the expected business process modeling challenges. In Section 6, we discuss the results from our study and detail implications for practice and research. We conclude in Section 7 with a summary of our findings.

## 2 Research Approach

### 2.1 Delphi Study Design

The technique chosen to facilitate the collection of, and consensus on, the key issues and challenges in process modeling was the Delphi technique [7] – a multiple-round approach to data collection. Delphi studies are useful when seeking consensus among experts, particularly in situations where there is a lack of empirical evidence [8]. The anonymous nature of a Delphi study can lead to creative results [9], reduces common problems found in studies that involve large groups [8] and allows for a wider participant scope due to the reduction of geographic boundaries [10]. In the case of our study, the Delphi technique is appropriate for three main reasons:

1. It facilitates obtaining expert consensus on current issues and future challenges of process modeling (and their definitions);
2. it facilitates the involvement of a large number of expert participants, in a short period of time, across many geographical boundaries and time zones; and
3. the objective of the study aligns with the general application area of the Delphi technique, which is that of forecasting and issue identification.

One of the main determinants of success of a Delphi study is the selection of the expert panel, i.e., the study participants [11]. Instead of utilizing a statistical, representative sample of the target population, a Delphi study requires the selection and consideration of qualified experts who have deep understanding of the domain or phenomenon of interest [10]. It also requires consideration of required levels of agreement. Moreover, careful planning of the schedule of contact with participants is also required to keep the study within a relatively short period of time so as to reduce non-response.

## 2.2 Participant Selection

To understand the perceived issues and future challenges of business process modeling, it is important to acknowledge different key stakeholders. The nature, or criticality, of any business process modeling issue may vary considerably depending upon the perspective taken by the respondent. We identify three groups of stakeholders: first, the *practitioners* of business process modeling, that is, the business analysts, system designers and other staff that actively use business process modeling approaches in their organizations. Second, the *vendors* of business process modeling tools and consulting solutions providing support to the end users. Third, the *academics* in the business process modeling domain, who develop next generation business process modeling artifacts and provide educational services.

Acknowledging these three groups, we designed a Delphi study that was conducted in three rounds separately for each of these stakeholder groups. The risk of being unable to obtain consensus between heterogeneous panelists [12] was further motivation to divide the study into the three related groups of stakeholders. Invitations were based on the expertise of the potential participants. For academics, we screened the program committee of the Business Process Management conference series ([www.bpm-conference.org](http://www.bpm-conference.org)), the most reputable conference in this area. Key selection criterion was the related research track record of a PC member. For vendors, we contacted key management staff from leading tool and methodology providers, as reported in current market studies (e.g., [13, 14]). For practitioners, we contacted the process managers, and similar positions, of large international corporations, who the research team knew through previous collaborations.

Regarding an appropriate panel size per expert group, typically, involvement rates of 10 participants are recommended [15] to overcome personal bias in consensus seeking. Seeking to surpass this recommendation, overall, invitations to the study were sent to 134 carefully screened experts (40 practitioners, 34 vendors, 60 academics), including 11 invitations based on referrals from invited participants. Of

**Table 1.** Response rates across all rounds of the Delphi study

Panel group	Response to initial contact	1 <sup>st</sup> round response	2 <sup>nd</sup> round response	3 <sup>rd</sup> round response
Academics	28	26	26	25
Vendors	21	21	18	18
Practitioners	24	23	22	19
<b>Total</b>	<b>73</b>	<b>70</b>	<b>66</b>	<b>62</b>

these, initially 73 experts agreed to participate, a 54.48% response rate. Table 1 shows the ongoing response rates over the three rounds of the Delphi study. By the 3rd round of the study, 62 experts were involved – an 84.93% ongoing participation rate.

### 3 Study Conduct

#### 3.1 Delphi Study Rounds

Our objective in conducting the Delphi Study was three-fold: First, to identify the key issues and future challenges of business process modeling, as perceived by the different panels. Second, to establish consensus on the issues and challenges. Third, to obtain and compare the rankings of the issues and challenges based on their perceived relative importance. According to our three objectives, our study was carried out over three rounds, matching recommendations for a relatively complete Delphi study [16].

In the first round, each participant was asked to list five current issues and five future challenges in business process modeling, together with a brief description of each issue/challenge. Overall, we received 70 (participants) x 2 (issues/challenges) x 5 (items) = 700 individual response items. To overcome challenges related to the number of response items, differences in terminology, term connotation and writing styles, we then codified each response item into higher level categories. For instance, we received two separate issue response items “No universal standard, and / or not knowing which standard to use, e.g. UML, BPMN, XPDL, etc.” and “Lack of a standard modeling language”. Both items can be coded to a higher-order issue “standardization of modeling notations, tools, and methodologies”.

In ensuring reliability and validity of this coding, we performed the exercise in multiple rounds. First, three researchers independently coded each of the 700 response items into a higher level category. In a second round, two researchers independently were exposed to the three codifications from the 1<sup>st</sup> coding round, and created individual, revised 2<sup>nd</sup> round coding drafts. In a third round, the fourth research group member consolidated the revised codifications and resolved any classification conflicts. We believe that through this multi-round approach we ensured inter-coder reliability as well as validity of the codification exercise.

The second round of the study was designed to obtain consensus from the participants on the codified issues and challenges, as well as on the definitions of the new higher-order categories. The communication for this round provided each participant with a personalized email containing his or her original responses, the agreed classifications per response item, and descriptions of the classifications. The participants were asked to indicate their level of satisfaction with the classification of their responses and the definitions of the classifications, and to provide additional information or suggestions if they were not satisfied with the classification. We received mostly positive responses on our codification (e.g., “Your categorization is close to the mark.”) as well as a small number of coding and/or definition improvement suggestions (e.g., “Tool support is misleading. I think something like tool complexity would be more appropriate.”).

It has been recognized that there are times when consensus between panelists is not possible [12]. However, there is also a lack of indication in the literature as to possible

**Table 2.** Satisfaction ratings for response codification

	Academics	Vendors	Practitioners
<i>Issues</i>			
Average satisfaction score	8.338	9.000	8.791
Standard deviation	1.853	1.185	1.143
<i>Challenges</i>			
Average satisfaction score	8.442	8.638	8.883
Standard deviation	1.520	1.468	1.150

measures for determining consensus. A recent Delphi study [17] utilized a satisfaction rating of 7.5 (out of 10). In our study, we asked the participants to rate their satisfaction on a scale of 1 to 10 (10 being highest) and assumed consensus at an average satisfaction level of 8 and a standard deviation below 2.0. As shown in Table 2, average satisfaction scores ranged from 8.338 (Issues, Academics) to 9.000 (Issues, Vendors), with standard deviations ranging from 1.853 (Issues, Academics) to 1.143 (Issues, Practitioners).

While our initial study plan allowed for multiple rounds of consensus building during this second stage of the study, the results obtained indicate that our multiple-coder approach to data classification resulted in the participants achieving the required consensus levels at the first iteration of the second round, which, in turn, allowed us to stop the consensus-building process at this stage. At the end of round two, and after making required changes to categories and/or definitions, where appropriate, all response items were ranked in descending order of ‘frequency of occurrence’, with items such as value of business process modeling (15 times), training (13 times), standardization (11 times) and model-driven process execution (9 times) being most frequently mentioned.

We recognize that frequency of occurrence is not an accurate measure of criticality, importance or priority. Accordingly, in the third round of the Delphi study, the experts were asked to assign to the response items a weighting that reflects the respondent’s relative importance of the particular item. In this round, data collection was carried out via a study website, with separate log-ins for the different expert panels. The participants were provided with the list of frequently mentioned issues and a separate list of frequently mentioned challenges (we defined ‘frequently mentioned’ as each item that was mentioned more than once in the first two rounds), together with their definitions. Overall, practitioners received a list of 14 issues and 13 challenges, while academics received lists of 21 and 16 items, and vendors received lists with 13 and 10 items. Each participant was given 100 points to assign across any of the issues, and 100 points to assign across any of the future challenges. The participants were free to assign the 100 points in any distribution, with the only condition being that exactly one hundred points were assigned across each of the lists. The online submission was only enabled when the participant met this condition for each of his/her two lists.

The collected data was then analyzed, and the average weightings of each issue and challenge were derived. From these calculations, we were able to derive top 10 lists, based on the average weightings, for process modeling issues and challenges for each of the three Delphi study groups. The results are listed in the Appendix.

### 3.2 Classification of Results

To better understand the nature, and implications, of the issues and challenges, we were interested in identifying the key capability area to which an issue or challenge applies. For instance, a challenge, ‘tool support’, clearly pertains to the availability (or lack thereof) of appropriate IT-based solutions to support the act of modeling, while a challenge ‘governance’ pertains to the establishment of appropriate organizational roles, duties and responsibilities for business process modeling.

In order to identify to which capability area the issues and challenges relate, we adopted a well-established and empirically tested model of the capability areas that are required to establish, and progress, Business Process Management (BPM) in an organization (e.g., [17, 18]). This model informs six capability areas, viz., strategic alignment, governance, method, IT, people, and culture. With business process modeling being an essential component of BPM, we adopted the capability area definitions to the more specific business process modeling context as follows (scope modifications highlighted in *italic*):

- **Strategic Alignment** is the continual tight linkage of *business process modeling* to organizational priorities and processes, enabling achievement of business goals.
- **Governance** establishes relevant and transparent accountability and decision-making processes to align rewards and guide actions *in business process modeling*.
- **Methods** are the approaches and techniques that support and enable consistent business process *modeling* actions and outcomes.
- **Information Technology** is the software, hardware and information management systems that enable and support business process *modeling* activities.
- **People** are the individuals and groups who continually enhance and apply their business process *modeling*-related expertise and knowledge.
- **Culture** is the collective values and beliefs that shape business process *modeling*-related attitudes and behaviors.

This model allowed us to map each of the top ten issues and challenges to one of the six capability areas, and, in turn, to provide a clear representation of which aspects of process modeling are considered by the respective panel groups. Similar to the coding exercise reported above, the mapping of the top 10 lists of issues and challenges to the capability areas utilized a multi-coder approach in order to reduce bias in the classification. Three members of the research group separately classified the issues and challenges lists for each of the three study groups. The classifications were consolidated and agreement statistics were calculated. We calculated an inter-rater agreement using Cohen’s Kappa [19] and achieved average Kappas of 0.809 for issues and 0.872 for challenges, indicating ‘excellent’ inter-rater agreement [20].

## 4 Business Process Modeling Issues

In a first analysis, we consider the current issues in business process modeling, as perceived by the three expert panels in our study. The Appendix lists the three top ten lists derived, and displays the rankings of the items as per their perceived relative importance. Visual inspection of these lists confirms our expectation that indeed the three stakeholder groups differ in terms of their perceived issues. Most notably,

	Strategic Alignment	Governance	Method	Information Technology	People	Culture
1		Standardization		Service Orientation Model-driven Process Ex.		
2	Value of Process Modeling Value of Process Modeling			Model-driven Process Ex.		
3	Business-IT-Divide		Flexibility			Buy-in
4	Expectations Management	Compliance Standardization				
5			Methodology		Training	Process Orientation
6		Governance	Modeling Views Modeling Level of Detail			
7		Standardization	Methodology Modeling Level of Detail			
8			Model Management Multi-perspective Modeling Model Management			
9			Model Management		Ease of Use	Adoption
10		Governance	View Integration	Model Integration		

**Fig. 1.** Business process modeling issues, mapped to capability areas. Academic issues are highlighted dark grey, vendor issues highlighted black and practitioner issues light grey.

practitioners ranked ‘Standardization’ as the most significant issue (mean rating 14.316), while vendors ranked ‘Model-driven process execution’ (mean rating 12.222) most important, with academics perceiving ‘Service orientation’ (mean rating 8.440) as most important. It is further interesting to note that the number one issue for practitioners (Standardization) overall received the highest average rating of relative importance across all three lists. In contrast, the number one issue voiced by academics (Service orientation), on average, was only the tenth most important issue when considering all three lists combined. In relation to the different capability areas relevant to process modeling, Fig. 1 shows how we mapped each of the thirty issues to the capability areas as per the model by de Bruin and Rosemann [17].

Several interesting observations can be drawn. First, overall 36% of the identified top issues address methodological aspects of business process modeling. Second, five of the ten issues voiced by academics fall into this area, indicating a strong focus on the methodology of modeling. Third, the ten practitioner and vendor issues cover all six capability areas, while academics’ issues do not address strategic alignment or culture. These findings suggest that vendors and practitioners are concerned with issues related to the purpose and adoption of process modeling while academics tend to concentrate on issues related to the development and evaluation of artifacts.

Regarding similarities in perceived issues across the three groups, we note that of the overall thirty top issues, the three lists contain 21 unique items, with five issues appearing in two lists (e.g., ‘model-driven process execution’, ‘value of process modeling’) and ‘Standardization’ and ‘Model management’ being the two issues that appear in each of the three top ten lists. In Table 3 we present a consolidated ordered list of perceived issues, determined by the combined average rating of each issue.

Computation of the data displayed in Table 3 allowed us to identify the most important issue in process modeling across all stakeholder groups. As can be seen,



**Table 3.** Overall top 10 business process modeling issues

Rank	Issue	Description	Mean Rating	Std. Dev.
1	Standardization	Issues related to the standardization of modeling notations, tools, and methodologies.	9.525	4.465
2	Value of process modeling	Issues related to the value proposition of process modeling to the business.	8.091	7.007
3	Model-driven process execution	Issues related to the model-driven development of executable process code and the lifecycle of process modeling to execution.	6.874	6.252
4	Model management	Issues related to the management of process models such as publication, version, variant or release management.	5.729	0.666
5	Modeling level of detail	Issues related to the definition, identification or modeling of adequate levels of process abstraction.	4.934	4.351
6	Methodology	Issues related to the process of process modeling.	4.690	4.202
7	Governance	Issues related to the governance of process modeling efforts or projects.	4.192	3.727
8	Buy-in	Issues related to the acquisition or ongoing assurance of buy-in and commitment from process modeling sponsors.	3.167	5.485
9	Business-IT-divide	Issues related to the use of process modeling in IT versus business scenarios, application areas or communities.	2.944	5.100
10	Process orientation	Issues related to the development or education of a process-aware perspective in relevant stakeholders or organizational units.	2.889	5.004

standardization is the most significant issue in business process modeling, followed by its value, and model-driven development of executable process code. Interestingly, standardization (e.g., [21]) and model-driven process execution (e.g., [22]) are topics fervently debated in academia at present, while the value of business process modeling has attracted only little academic attention as yet.

## 5 Business Process Modeling Challenges

In a second analysis, we considered the future challenges in business process modeling, defined as issues emerging over the next five years. The Appendix lists the three top ten lists derived, and displays the rankings of the items as per their perceived relative importance. Again we note interesting results. Similar to the case of the perceived issues, the three lists contain overall 22 different challenges. However, it would appear vendors and academics perceive similar challenges. Most notably, both groups voice ‘Model-driven process execution’ to be the number one challenge in the future (average ratings 16.222 and 10.960), with practitioners perceiving the establishment of a business value proposition as the key future challenge (average rating 16.632). Again, the number one item of the practitioners’ lists is the overall most important item as per the average rating.

	Strategic Alignment	Governance	Method	Information Technology	People	Culture
1	Value of Process Modeling			Model-driven Process Ex. Model-driven Process Ex.		
2	Business-IT-Alignment		Methodology			Buy-in
3	Value of Process Modeling	Standardization		Service Orientation		
4	Expectations Management		View Integration		Ease of Use	
5	Value of Process Modeling	Standardization Governance				
6		Standardization	Collaborative Modeling		Training	
7	Process Architecture		Model Management		Training	
8			Data-centric Modeling	Service Orientation Model Integration		
9		Compliance	Model Management			Adoption
10			Ontology	Tool Support		Re-use

**Fig. 2.** Business process modeling challenges, mapped to capability areas. Academic challenges are highlighted dark grey, vendor challenges highlighted black and practitioner challenges light grey.

Regarding the capability areas addressed, Fig. 2 shows the results from our mapping of the challenges to the six business process modeling capability areas.

We again identify a number of interesting observations. Most notably, the challenges of the different stakeholder groups, while overlapping to some extent, pertain to different areas of business process modeling capability. Three of the practitioners' ten challenges (buy-in, adoption and re-use) address the organizational culture, while neither academics nor vendors perceive this area to be problematic in the future. Instead, a combined seven challenges of academics and vendors address methodical aspects of business process modeling – an area apparently not expected by practitioners to be problematic. Also, while a 'people' focus is apparent in some of the challenges voiced by vendors and practitioners ('training', most notably), this capability area is not perceived as a critical challenge by academics. This group focuses its perceived challenges on the areas of method and IT, with seven of the top ten challenges falling into these two capability areas. In contrast, only one practitioner challenge (Model integration) falls in this area, with the remaining nine challenges addressing all other capability areas.

Considering a holistic view of process modeling challenges, Table 4 shows a consolidated list of the top ten future challenges across all participant groups. Similar to the case of current process modeling issues, we found that four items (Model-driven process execution, Service orientation, Model management, and Training) appeared in two of the lists, and two challenges (Value of process modeling and Standardization) were perceived as critical by all three expert panels. Interestingly, comparison of Table 3 and Table 4 shows that the overall top three issues and challenges are the same, with only the ranking as first, second or third, differing between the current state of process modeling and the future state in five years time. This finding suggests the key criticality of these current and future issues, and presents a strong call for increased attention to these aspects both in industry practice, and in process modeling research.

**Table 4.** Overall top 10 business process modeling challenges

Rank	Issue	Description	Mean Rating	Std. Dev.
1	Value of process modeling	The establishment of a business value proposition of process modeling.	12.893	5.041
2	Model-driven process execution	The support for process enactment, automation or execution based on process models.	9.061	8.276
3	Standardization	The standardization of process modeling approaches, methodologies, tools, methods, techniques or notations.	8.340	1.221
4	Business-IT-alignment	The use of process modeling to support alignment between business and IT stakeholders, viewpoint or approaches.	5.111	8.853
5	Service orientation	The support for aspects relevant to the management of web services, service-oriented architectures or quality of services.	5.039	4.477
6	Training	The establishment of process modeling expertise.	4.543	3.936
7	Model management	The management of process model variants, versions, releases, changes etc.	4.264	3.736
8	Buy-in	The acquisition or ongoing assurance of buy-in and commitment from process modeling sponsors.	4.114	7.126
9	Ease of use	The complexity or easiness of process modeling methodologies, tools or notations.	3.648	6.319
10	Collaborative modeling	The involvement of multiple people in the modeling of processes.	3.000	5.196

## 6 Discussion and Implications

### 6.1 Discussion

Through the analysis presented above, we identify zones of concordance and discordance between key stakeholder groups in business process modeling. Our findings suggest that the endeavors of academics and vendors are not always aligned to current or future needs of industry.

Notably, our study identified that the top three issues in business process modeling at the moment, considering rankings from all three participant groups, are those of standardization of process modeling, identification of the value of process modeling, and also model-driven process execution. Interestingly, the participants felt that these issues were so significant that they will still be challenges in five years to come. Our study also identified that the three groups of process modeling stakeholders have different opinions of the critical issues and challenges in the business process modeling domain. For example, while practitioners rank standardization of modeling notations to be the top critical current issue, academics perceive service-orientation as the main issue, despite the standardization issue still being largely unsolved.

While we would agree that to a large extent the endeavors of academics and tool vendors should be visionary in nature, i.e., setting the ground work for solving challenges that practitioners are likely to face in the future, our study finds only limited indication of this situation occurring in actual industry practice. The practitioners consider their current top three issues viz. standardization, value of process modeling, and buy-in, to still be the top three challenges in five years time (albeit in a different order). This situation indicates that these issues are indeed critical and more guidance is expected on how to proceed. On the flip side, the academics consider service-orientation, model-driven process execution, and flexibility to be the current top three issues. If we consider that research takes a few years to be assimilated into industry and products, none of those issues are mentioned at all in the top ten current issues, nor future challenges, by the practitioners. The vendors have somewhat better alignment with practitioners in terms of the perceived most critical issues, with value of process modeling being the #2 current issue. Even consideration of some of the lower ranked issues still shows lack of alignment between the current foci of the academics and vendors, as compared to the future challenges identified by practitioners. Standardization, for example, which is ranked only #7 on the current critical issues list for academics, is the #3 expected future challenge for practitioners.

Another interesting situation emerges when analyzing the differences within the same group of stakeholders in terms of current critical issues and future challenges. Eight of the current issues for practitioners are still expected to persist as top ten challenges in the future. The situation for academics, while considering a different set of topics, is similar, with seven current issues still expected to be in the top ten challenges in five years time.

## 6.2 Implications for Practice and Research

Our study provides implications for the industry ecosystem of end user organizations as well as vendors of tools and consultancy offerings. Through the presentation of the current issues, these stakeholder groups are informed about the key critical factors that could potentially undermine success or value generation of business process modeling projects. The identified issues also help to channel attention to the major obstacles persisting in process modeling practice (e.g., model management and standardization), and should motivate practitioners and vendors to consider appropriate solutions or at least workarounds to some of the issues. Most notably, the standardization of process modeling appears to be top on the agenda for process modeling stakeholders. For end users, this finding implies setting up, and using, an appropriately standardized modeling environment and available standards (e.g., BPMN, BPEL etc.), while for vendors it implies importance to adapt their offerings so as to incorporate existing standards.

In addition to the insights we provide to the practice of business process modeling, our work also informs a research agenda for process modeling-related research. On the basic assumption that research should consider relevant topics of future interest to practitioners, the contrast between future challenges identified by business process modeling practitioners and the current issues of interest to academics identifies a number of areas that are of interest to practitioners but do not appear as yet on the radar screen of BPM scholars. Such areas include, for instance:

- **Value of business process modeling:** Research that studies the value proposition, the net benefits or the cost drivers associated with business process modeling.
- **Expectations management:** Research that examines the expectations and pre-conceptions, and the (dis-) confirmation of those, of different stakeholder groups involved in business process modeling.
- **Training:** Research that studies different approaches to building business process modeling expertise, the effects of expertise on the quality of business process modeling, or the key factors determining process modeling expertise.
- **Process architecture:** Research that examines the development, use, composition, or value of architectural models to guide the act of business process modeling.
- **Adoption:** Research that studies the key determinants and impediments associated with the adoption and continued use of business process modeling on an individual or organizational level.

We note that some of these areas of concern to practitioners appear to be similar in nature to a range of the established streams of research in Information Systems in general. For example, adoption [23], expectation [24] or value [25] of Information Technology are well-established domains of IS research. However, it would appear that these areas have, to date, been under-researched in the domain of business process modeling and management. This situation brings forward a challenge as well as an opportunity. Future research in these areas could build upon the body of knowledge existent in the IS domain, and extend or amend existing theories to fit the specific context of business process modeling. Some examples of how such work could be carried out already exist (e.g., [26]).

## 7 Conclusions

Business process modeling is a foundational requirement in many management and IS projects, yet it still represents a significant challenge to many organizations. This paper presents the results of the first global large-scale Delphi study on the current issues and future challenges in the business process modeling domain. The identification of the most critical issues and challenges – from three separate perspectives of academics, practitioners and vendors – enables us to develop deeper insights into the interplay of research and practice, and to propose a set of industry-relevant topics for the research community. Indeed, on the basis of our findings, we would argue that increasing the synergy between the three groups will lead to: (a) industry-relevant research that facilitates increased business process modeling maturity in organizations, in turn generating the need for research in novel modeling approaches, and, (b) the development of tools and supporting methodologies that are better suited to the needs of the market.

We identify the Delphi study approach as a potential limitation in our work. Delphi studies are said to be susceptible to a number of weaknesses including (1) the flexible nature of study design [9], (2) the discussion course being determined by the researchers [7], and (3) accuracy and validity of outcomes [27]. In our study, measures were taken to minimize their potential impact. Such measures included: (1) establishing assessment criteria for measuring inter-rater agreements; (2) use of a

multiple coders; (3) using multiple coding rounds and (4) following established methodological guidelines for the conduct of Delphi studies (e.g., [10, 11, 16]).

In our future work we seek to provide a detailed analysis of additional qualitative responses gathered in a later fourth round of the study, which exposed the top 10 lists to all participant groups and elicited the comments of the participants. In a related stream of research, we will complement this Delphi study with a similar study on the perceived benefits of business process modeling, to provide a balanced perspective.

## References

1. Davies, I., Green, P., Rosemann, M., Indulska, M., Gallo, S.: How do Practitioners Use Conceptual Modeling in Practice? *Data & Knowledge Engineering* 58, 358–380 (2006)
2. Dumas, M., van der Aalst, W.M.P., ter Hofstede, A.H.M. (eds.): *Process Aware Information Systems: Bridging People and Software Through Process Technology*. John Wiley & Sons, Hoboken, New Jersey (2005)
3. Davenport, T.H., Short, J.E.: The New Industrial Engineering: Information Technology and Business Process Redesign. *Sloan Management Review* 31, 11–27 (1990)
4. Erl, T.: *Service-oriented Architecture: Concepts, Technology, and Design*. Prentice Hall, Upper Saddle River (2005)
5. Scheer, A.-W.: *ARIS - Business Process Modeling*, 3rd edn. Springer, Berlin (2000)
6. Eikebrokk, T.R., Iden, J., Olsen, D.H., Opdahl, A.L.: Exploring Process-Modelling Practice: Towards a Conceptual Model. In: *Proceedings of the 41st Annual Hawaii International Conference on System Sciences*, vol. 376. IEEE, Waikoloa (2008)
7. Dalkey, N., Helmer, O.: An Experimental Application of the Delphi Method to the Use of Experts. *Management Science* 9, 458–467 (1963)
8. Murphy, M.K., Black, N.A., Lamping, D.L., McKee, C.M., Sanderson, C.F.B., Askham, J., Marteau, T.: Consensus Development Methods, and their Use in Clinical Guideline Development. *Health Technology Assessment* 2, 1–88 (1998)
9. van de Ven, A.H., Delbecq, A.L.: The Effectiveness of Nominal, Delphi, and Interacting Group Decision Making Processes. *Academy of Management Journal* 17, 605–621 (1974)
10. Okoli, C., Pawlowski, S.D.: The Delphi Method as a Research Tool: an Example, Design Considerations and Applications. *Information & Management* 42, 15–29 (2004)
11. Powell, C.: The Delphi Technique: Myths and Realities. *Journal of Advanced Nursing* 41, 376–382 (2003)
12. Richards, J.I., Curran, C.M.: Oracles on “Advertising”: Searching for a Definition. *Journal of Advertising* 31, 63–76 (2002)
13. Hall, C., Harmon, P.: *The 2007 Enterprise Architecture. Process Modeling, and Simulation Tools Report*. BPTrends.com (2007)
14. Blechar, M.J.: *Magic Quadrant for Business Process Analysis Tools*. Gartner Research Note G00148777. Gartner, Inc., Stamford, Connecticut (2007)
15. Cochran, S.W.: The Delphi Method: Formulation and Refining Group Judgments. *Journal of Human Sciences* 2, 111–117 (1983)
16. Linstone, H.A., Turoff, M. (eds.): *The Delphi Method: Techniques and Applications* [Online Reproduction from 1975]. Addison-Wesley, London (2002)
17. de Bruin, T., Rosemann, M.: Using the Delphi Technique to Identify BPM Capability Areas. In: Toleman, M., Cater-Steel, A., Roberts, D. (eds.) *Proceedings of the 18th Australasian Conference on Information Systems*, The University of Southern Queensland, Toowoomba, Australia, pp. 643–653 (2007)

18. de Bruin, T.: Insights into the Evolution of BPM in Organisations. In: Toleman, M., Cater-Steel, A., Roberts, D. (eds.) Proceedings of the 18th Australasian Conference on Information Systems, The University of Southern Queensland, Toowoomba, Australia, pp. 632–642 (2007)
19. Cohen, J.: A Coefficient of Agreement for Nominal Scales. *Educational and Psychological Measurement* 20, 37–46 (1960)
20. Landis, J.R., Koch, G.G.: The Measurement of Observer Agreement for Categorical Data. *Biometrics* 33, 159–174 (1977)
21. Recker, J.: Opportunities and Constraints: The Current Struggle with BPMN. *Business Process Management Journal* 15 (in press, 2009)
22. Ouyang, C., van der Aalst, W.M.P., Dumas, M., ter Hofstede, A.H.M., Mendling, J.: From Business Process Models to Process-Oriented Software Systems. *ACM Transactions on Software Engineering Methodology* 19 (in press, 2009)
23. Venkatesh, V., Davis, F.D., Morris, M.G.: Dead Or Alive? The Development, Trajectory And Future Of Technology Adoption Research. *Journal of the Association for Information Systems* 8, 267–286 (2007)
24. Bhattacharjee, A.: Understanding Information Systems Continuance: An Expectation-Confirmation Model. *MIS Quarterly* 25, 351–370 (2001)
25. Mukhopadhyay, T., Kekre, S., Kalathur, S.: Business Value of Information Technology: A Study of Electronic Data Interchange. *MIS Quarterly* 19, 137–156 (1995)
26. Recker, J.: Understanding Process Modelling Grammar Continuance: A Study of the Consequences of Representational Capabilities. Faculty of Information Technology, Queensland University of Technology, Brisbane (2008)
27. Ono, R., Wedemeyer, D.J.: Assessing the Validity of the Delphi Technique. *Futures* 26, 289–304 (1994)

## Appendix

Rank	Practitioners		Vendors		Academics	
	Issue	Mean Rating	Issue	Mean Rating	Issue	Mean Rating
1	Standardization	14.316	Model-driven process execution	12.222	Service orientation	8.440
2	Value of process modeling	12.105	Value of process modeling	12.167	Model-driven process execution	8.400
3	Buy-in	9.500	Business-IT-divide	8.833	Flexibility	7.480
4	Expectation management	8.474	Standardization	8.778	Compliance	6.880
5	Training	8.316	Process orientation	8.667	Methodology	5.960
6	Governance	7.132	Modeling level of detail	8.222	Modeling views	5.880
7	Modeling level of detail	6.579	Methodology	8.111	Standardization	5.480
8	Model management	6.368	Multi-perspective modeling	7.333	Model management	5.040
9	Adoption	6.263	Model management	5.778	Ease of use	4.920
10	Model integration	5.632	Governance	5.444	View integration	4.640

Rank	Practitioners		Vendors		Academics	
	Challenge	Mean Rating	Challenge	Mean Rating	Challenge	Mean Rating
1	Value of process modeling	16.632	Model-driven process execution	16.222	Model-driven process execution	10.960
2	Buy-in	12.342	Business-IT-alignment	15.333	Methodology	8.800
3	Standardization	8.632	Value of process modeling	14.889	Service orientation	8.560
4	Expectations management	7.842	Ease of use	10.944	View integration	8.560
5	Governance	7.079	Standardization	9.389	Value of process modeling	7.160
6	Training	6.684	Collaborative modeling	9.000	Standardization	7.000
7	Process architecture	6.316	Training	6.944	Model management	6.960
8	Model integration	6.289	Service orientation	6.556	Data-centric process modeling	6.560
9	Adoption	6.132	Model management	5.833	Compliance	6.160
10	Re-use	5.868	Ontology	4.889	Tool support	6.080

# Deriving Information Requirements from Responsibility Models

Ian Sommerville, Russell Lock, Tim Storer, and John Dobson

School of Computer Science, University of St Andrews, St Andrews, Scotland

[ifs@cs.st-andrews.ac.uk](mailto:ifs@cs.st-andrews.ac.uk)

<http://www.cs.st-andrews.ac.uk/~ifs>

**Abstract.** This paper describes research in understanding the requirements for complex information systems that are constructed from one or more generic COTS systems. We argue that, in these cases, behavioural requirements are largely defined by the underlying system and that the goal of the requirements engineering process is to understand the information requirements of system stakeholders. We discuss this notion of information requirements and propose that an understanding of how a socio-technical system is structured in terms of responsibilities is an effective way of discovering this type of requirement. We introduce the idea of responsibility modelling and show, using an example drawn from the domain of emergency planning, how a responsibility model can be used to derive information requirements for a system that coordinates the multiple agencies dealing with an emergency.

## 1 Introduction

There has been an accelerating trend for information systems development to be based on commercial off-the-shelf (COTS) products or Enterprise Resource Planning systems (ERP). Such systems may incorporate or link with existing information systems, operated by different parts of a business or by different agencies. Examples of such systems include medical records systems, ‘enterprise systems’ that integrate the functions of an enterprise and coordination systems that support different organizations who are working together on a shared problem.

The requirements for these information systems are significantly constrained by the system or systems on which they are based. Rather than focus on functionality and behavioural characteristics, requirements engineering for such systems should be concerned with how they can be configured and used to support the work of an organisation.

Our current research focuses on the engineering of these large-scale, information systems and their use in an organisational environment. The aspect of this research described here is concerned with discovering and understanding the requirements for information systems that coordinate the work of multiple cooperating agencies. The novel features of the work include:



1. A focus on information requirements as the most significant type of requirements for complex organizational systems.
2. The use of models of the responsibilities in organizations to support the process of information requirements discovery.

We argue that the key requirements for many organizational systems are ‘information requirements’, i.e. requirements that define the information provided to stakeholders to help them do their work, requirements for information sharing and access control and requirements for the information that is generated. These are an essential basis for system configuration and for defining the information in different databases that has to be shared.

In this paper, we discuss what we mean by information requirements and introduce the concept of responsibility as a natural abstraction that supports the discovery of information requirements. We discuss how a model of responsibilities in a system can help understand and analyze these information requirements.

Our approach is illustrated using several models of civil emergency planning. We show how information requirements for a coordination system can be discovered from the models. This responsibility-based approach to requirements discovery is compared with viewpoint-oriented and goal-based approaches and we present a qualitative evaluation of the work.

## 2 Information Requirements

Current requirements engineering methods that support requirements discovery, by and large, assume that the outcome of the process is a behavioural description of the proposed system. The resulting requirements document describes the system’s functionality and sets out how the system should behave in response to events and user inputs. However, we have found that there are circumstances where a behavioural approach to requirements description is inappropriate:

1. When the system is constructed using COTS products or ERP systems. Here, the planned system behaviour is determined to a large extent by the underlying generic system.
2. When the requirements are for a system of systems, or where there are existing systems that must integrate with a new system. These constrain the behaviour of any new system. This is particularly true where the different systems are owned by different agencies, who may be reluctant to make changes to them.
3. When the requirements have to reflect the social and organizational context of a system rather than its operational use. The social and organizational context establishes the rules, regulations and policies that apply to a system as well as organization-specific requirements that reflect the power structures, culture and ethical values of an organization.

One motivation for introducing COTS and ERP systems is to support new business processes that reduce information processing inefficiencies by sharing

of information across business units. For example, information about students who apply to a university may be captured on application and automatically made available to other functions such as examination, assessment and alumni tracking. Systems may then be configured to ensure that the people involved in the business processes have access to the information that they need.

The key problem for such systems is understanding the information needs of the people in the organization who are responsible for the different functions of the system. These 'information requirements' are used as a basis for defining the system data models, business rules, information flows and generated reports. They should also include knowledge of who needs what information and when it is required to help avoid information overload and to configure the access controls of the system.

The term 'information requirements' is not widely used. Here, we refer to those requirements concerning information to be provided to system stakeholders including where and how that information is used; or the information that they are expected to create using the system. These are distinct from both presentation requirements, which reflect how that information is delivered to different stakeholders, and processing requirements, which define the automated information processing carried out by the system.

Organizations installing new enterprise information systems usually define new business process models and attempt to derive the information requirements by analyzing these models. However, there have been many anecdotal reports of major problems with such systems; they have often failed to deliver the expected improvements. While there are many different reasons for the problems, one important factor is that too many organizations rely on a simplistic notion of business processes:

1. *Defined business processes are rarely followed by the people doing the work in an organization.* Almost without exception, ethnographic studies of work have revealed that business process models are not a good description of actual operational activities [1,2]. Individuals and teams modify and extend their processes to cope with deficiencies in the systems that they use, to adapt these processes to local circumstances and to simplify exception handling.
2. *Business processes are not an appropriate way of representing some kinds of work.* For professionals, much of their work is knowledge-based and reflects their professional discipline, training and culture. Rather than follow a standard process, individuals decide how to organize and record their work, with different people, doing the same job, working in different ways.
3. *Business process descriptions do not (and cannot) define all possible exceptions and failures that may occur.* Exception handling is left to individuals, who use their detailed knowledge and local information to deal with problems.

Business process modeling are effective in eliciting the minimal set of information required to produce an output. Our own observations, over more than 15 years [3,4,5], suggest that local process changes involve maintaining additional,

sometimes redundant, information to that provided by a computer-based system or using the affordances of the information representation to support collaboration and communication. These support resilience, allowing work to continue in the event of system failure and make it easier for people to help others if staff are busy or unavailable.

We argue then that information requirements cannot just be derived from business process models. They also require a detailed understanding of how work is actually done and how information is acquired, used and generated in its undertaking.

Information requirements, as discussed here, are also profoundly influenced by a wider class of requirements on a socio-technical system. These were first identified by the ORDIT project [6] as ‘organisational requirements’. We now prefer the term ‘enterprise requirements’ so as to avoid any possible confusion with ‘business requirements’, which reflect the functions and goals of the business.

Enterprise requirements are those requirements on a socio-technical system that are derived from the system being placed in a particular social and organisational context rather than from functions to be performed or tasks to be supported. Examples of sources of such requirements are power structures, responsibilities and obligations, control and autonomy, values and ethics. Requirements of this kind are embedded in organisational structure and policies, often in a way that cannot be directly observed or easily articulated. Therefore, these requirements are not so much captured as debated.

To illustrate what we mean here, consider the situation in hospitals where there is perennial (and probably inevitable) tension between the hospital administrators and the senior doctors. Information that is required to support administration is inevitably different from clinical information and providing that information often requires doctors to do extra work. If doctors are in a strong position within the organization, they may simply refuse to provide that information, thus constraining the information system. On the other hand, if the power struggle favours the hospital managers, then the doctors may comply with the demands to change the way they capture patient information. The information requirements depend on the power relationships in the organization rather than prioritizing stakeholder needs.

We believe that for a large class of systems, information requirements are the most important requirements because it is often difficult to devise work arounds to cope with omissions and inadequate information. Problems are particularly acute where the information requirements of one stakeholder group are based on information that should be produced by some other group. If there is a mismatch here, then it may be very difficult or impossible to recreate the information required.

For example, in studies of a patient information system, built using a COTS product, it was found that the information requirements of clinicians were quite distinct from the information requirements of hospital managers. Managers required information about why patients were referred for treatment using standard classifications. However, these were quite different from the classifications

used by the clinicians working in that domain. Reconciling these turned out to be very difficult indeed, partly because of the tensions discussed above [7].

We argue then, that where there are significant pre-existing limitations on system behaviour, the RE process should focus on understanding the information that needs to be provided to stakeholders to allow them to do their job, the ‘enterprise requirements’ that constrain the way that this information is used and the information that is generated. The focus is not ‘what should the system do’ but ‘what do the stakeholders need and produce’.

### 3 Responsibility

For some time, our interests have been in the interaction between humans, organisational structures and software systems which form constituent parts of broader socio-technical systems. In particular, we are interested in the influence which such issues have on the dependability of systems [8].

As part of this work, we have investigated how we can use an understanding of the responsibilities of agents in a socio-technical system to discover potential system vulnerabilities that can lead to system failure. In particular, we have been interested in highlighting the potential for ‘responsibility failure’ where an agent cannot properly discharge its assigned responsibilities, or where responsibilities are inappropriately assigned to agents in the system [9]. By a ‘responsibility’, we mean:

*A duty, held by some agent, to achieve, maintain or avoid some given state, subject to conformance with organisational, social and cultural norms.*

The key points in this definition are that a responsibility is a duty, which implies that the agent holding the responsibility is accountable to some authority for their actions, that responsibilities may be concerned with avoiding undesirable situations and not just with accomplishing some actions and that, in discharging responsibilities, agent behaviour is constrained by laws, regulations and social/cultural conventions and expectations.

The notion of a responsibility includes the specification of objectives (goals) to be achieved in discharging the responsibility. However, there is also an acknowledgment that in complex socio-technical systems, the achievement of an objective is subject to a range of constraints, that are often implicit. For example, doctors discharge responsibilities subject to ethical constraints, companies operate subject to the financial regulations of their host country. The notion of responsibility as a duty includes an embedded assumption that it is how an agent acts and not just what is achieved that is important. This affects the information requirements, particularly those requirements that define the information that must be recorded when a responsibility is discharged. There may also be requirements to record information, in some critical contexts, about options considered and rejected so that the agent holding the responsibility may be held to account.

In some cases, responsibilities may be procedural, where the agent follows a defined process. However, many responsibilities are knowledge-based and the

agent assigned the responsibility decides how to discharge that responsibility, based on their knowledge and experience, local circumstances and other factors. Even for procedural responsibilities, agents may have discretion in how to cope with exceptions, busy periods of work and so on.

One important reason for using responsibilities as an abstraction is that they are a natural notion that people can relate to. Work is often defined in terms of responsibilities and, if you ask people what their responsibilities are, they can usually tell you. People can usually separate their responsibilities at work from their responsibilities in other areas (e.g. as a parent).

Responsibilities serve as an effective basis for focusing the requirements debate, because of their ubiquity and naturalness. Different stakeholders may have quite different views about particular responsibilities and by making these responsibilities explicit, we provide a means to stimulate that debate.

A valid question, of course, is how do you discover and understand the responsibilities in a complex work setting? So far, our understanding of responsibilities has been based on three complementary approaches:

1. *Document analysis.* Documents that describe the work of a business such as plans or process descriptions often, explicitly or implicitly, discuss the responsibilities of the agents involved.
2. *Stakeholder interviews.* As discussed above, it is natural for people to talk about their own and other people's responsibilities.
3. *Field observations.* Ethnographic studies of work settings reveal how responsibilities are played out in practice and, in particular, how responsibilities may be delegated, transferred and shared. We have been involved in such studies in four areas - hospital administration [9], emergency planning [10], election systems [11] and, more recently, university administration.

These techniques elicit both *planned* and *operational* responsibilities. Planned responsibilities are anticipated by an organisation and describe expectations as to how responsibilities are or will be assigned. On the other hand, operational responsibilities refers to the dynamic assignment of responsibility in response to actual events, and may result in rather different responsibility structures than those anticipated by the organisation. Document analysis is effective for eliciting planned responsibilities, but plans will incorporate changes as a result of observations of responsibility assignment and discharge in practice.

## 4 Responsibility Modelling

A responsibility model is a succinct description of the responsibilities in a system, the agents who have been assigned these responsibilities and the resources that should be available to these agents to assist them in discharging their responsibilities. Although we will explain the models illustrated here, a more complete description of the notation can be found in [12].

The domain in which we are currently working is that of civil emergency management. This involves the emergency services and local and national government working together to cope with some emergency, such as flooding, terrorist attack, aircraft accident, etc. We have been investigating how we can use a model of responsibilities as a way of revealing planning vulnerabilities and the requirements for systems support by the emergency coordination centres. The case studies that we have used have been based on emergency plans for dealing with extensive flooding.

The role of the emergency coordination centres in the UK is to coordinate the ‘work on the ground’ of the different agencies involved. This involves prioritizing actions, maintaining an overview of the situation, facilitating information exchange across agencies and communicating with the media. The agencies involved (fire, police, ambulance, etc.) have their own command and control systems and information systems. Communication between agencies is normally managed by liaison officers within the coordination centre.

In this case, the requirements are for an information management system that allows information to be shared across agencies, that facilitates the transmission of information to people dealing with the emergency and that logs decisions taken by coordinators. Logging is critical both as a means of learning from experience after the recovery phase of the emergency has been completed and as an accountable record of actions.

Figure 1 is a model of the responsibilities involved in evacuating an area in the face of an imminent or actual threat (from flooding). Responsibilities are denoted by round-edged rectangles and agents are named in angle brackets. Dependencies between responsibilities (such as decomposition into sub-responsibilities) are indicated by links between the responsibility icons.

Responsibilities have a set of attributes and an associated description that can take several forms, depending on the type of responsibility [13]. For responsibilities that are normally discharged in the same way, the description can be a work flow model. For knowledge-based responsibilities, the description is usually

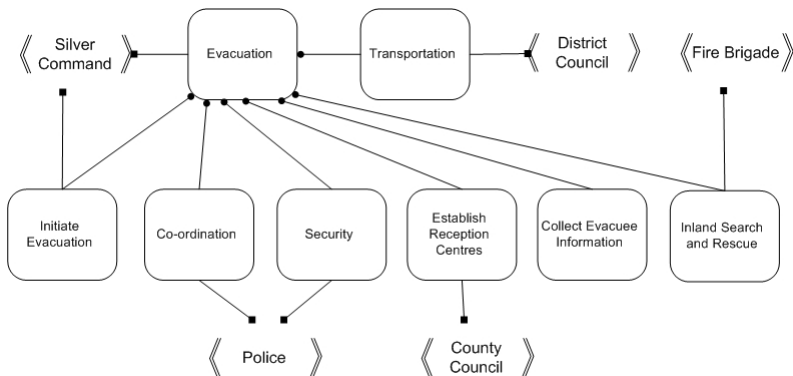
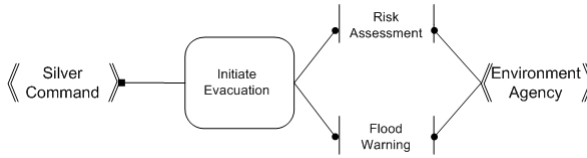


Fig. 1. Responsibility model of evacuation coordination



**Fig. 2.** Resources associated with a responsibility

textual. Attributes may include the goal or goals associated with the responsibility, the context where it is normally discharged, and pre and post-conditions defining assumptions.

To explain Figure 1, Silver Command (the name given to the coordination centre) initiates an evacuation given situation information (flood warnings, weather forecasts, etc.) and a risk analysis. The police coordinate the evacuation process and are responsible for safety and security during the evacuation. If people are trapped, the fire service are responsible for search and rescue but they may not be involved if the evacuation is in advance of a predicted flood. Two levels of local authority are also involved. The district council must provide transport for the evacuees and the county council must provide safe accommodation, food and water, etc. Other agencies, such as the ambulance service, may also be called on if there are ill or infirm people to be evacuated, but we have simplified our model to exclude these.

Notice that the responsibility ‘Collect Evacuee Information’ does not have an agent associated with it. Drawing up the responsibility model revealed this vulnerability in the emergency plan, since it was not explicit which agency should collect this information.

We can also associate resources with responsibilities as shown in Figure 2. A resource can be information or some physical entity such as a flood barrier, vehicle, etc. In this case, the responsibility ‘Initiate Evacuation’ requires information resources (named between vertical bars) in order to discharge that responsibility. The Environment Agency must provide a flood warning (which indicates when local rivers may flood in the near future) and a risk assessment, which shows the areas that would be affected by a flood and so should be evacuated.

## 5 Responsibility and Requirements Engineering

We are currently exploring how responsibilities can be used to support the early stages of the RE process, namely the discovery and analysis of requirements from multiple stakeholders. There are three areas where our work suggests they are useful: understanding enterprise requirements; information requirements discovery; and stakeholder identification. Here, we focus on the use of responsibility models in discovering information requirements.

We believe that for enterprise information systems, the critical requirements are information requirements. In the case of emergency management, different agencies maintain different types of information. The most important system

requirements are to identify the information that has to be shared and how it will be communicated to the people who need that information.

The nature of emergency management is such that the people involved react to the situation on the ground in order to discharge their responsibilities; it is impossible to define an evacuation 'process'. The information they require normally comes through liaison officers based in a coordination centre.

To help us understand the information requirements, we ask a range of questions, prompted by the responsibility model. These are:

1. What information is required to discharge this responsibility?
2. What channels are used to communicate this information?
3. Where does this information come from?
4. What information is recorded in the discharge of this responsibility and why?
5. What channels are used to communicate this recorded information?
6. What are the consequences if the information required is unavailable, inaccurate, incomplete, late, early?

To see how these questions are used, let us focus on the coordination of evacuation of people, which in the UK is the responsibility of the police, working through a local, mobile coordination centre. When we applied the above questions to the emergency flood plan for Cumbria (in the north of England), we derived the following:

1. *What information is required to discharge this coordination responsibility?*
  - A map of the area that is threatened by the emergency.
  - A list of 'priority premises' to be evacuated. These are premises that are evacuated first because they include people who may not be able to evacuate themselves. hospitals, schools, care homes, etc.
  - A list of assembly points where people should gather. The evacuation involves people being directed (or if necessary helped) to these local assembly points from where they are picked up and transported to a safe location.
  - Information about known, unsafe routes in the area.
  - Information about premises that are known to have been evacuated.
  - Information about the emerging threat situation (e.g. predictions of how long until flooding occurs).
  - Information about the capacity and availability of transport from the assembly points.
  - Information about the number of police available to assist with evacuation and the availability of personnel from other emergency services. If the emergency is predicted and has not yet happened, there may be no need for support from other services.
2. *Where does this information come from?*
  - Area map: County council
  - Priority premises: District council
  - Assembly points: District council



- Known, unsafe routes: Police
  - Evacuated premises: Police, Fire service
  - Threat situation: Environment agency
  - Transport information: District council
  - Personnel available: Police, Other services
3. *What channels are used to communicate this information?*
- Area map: Radio data link to printers in local mobile command centre
  - Priority premises: Radio data link to printers in local mobile command centre
  - Assembly points: Radio data link to printers in local mobile command centre
  - Known unsafe routes: Radio from Silver command
  - Evacuated premises: Radio from Silver command
  - Threat situation: Radio from Silver command
  - Transport information: Radio from Silver command
4. *What information is recorded and why?*
- Information about premises evacuated, area evacuation times, units responsible for evacuation. (for accountability)
  - Information about unchecked premises. (in case of future emergency calls)
  - Information about unsafe routes (to assist evacuation)
5. *What channels are used to communicate this information?*
- Radio from ground units to local control centre. Email to Silver command if available, otherwise radio.
  - Word of mouth local reporting
6. *What are the consequences if the information required is unavailable, inaccurate, incomplete, late, early?*
- To assess this, we need to look at each information item in turn. Take, for example, the list of priority premises to be evacuated.
- *Unavailable*: Manual premises check required to see if vulnerable people to be evacuated.
  - *Inaccurate*: Manual premises check may be necessary. Possible delay in evacuation of vulnerable people. People may be left behind.
  - *Incomplete*: Possible delay in evacuation.
  - *Late*: Information has to be communicated to units in the field rather than at local coordination centre.
  - *Early*: No consequence.

The information derived from discovering the information required to discharge a responsibility may then be recorded in a responsibility model, using the approach shown in Figure 2. This may then be used as a basis for debate and discussion about the information.

The final stage in the process is to translate the information requirements into system requirements for supporting information sharing. We do not have space to discuss this in any depth here but simply provide some examples of requirements (with associated rationale) below:

1. The coordination centre system shall be able to import textual information from the District Council planning system, the Police emergency system and the Fire Service emergency system. (Different types of information needs to be shared and this allows for information transfer between agencies).
2. The coordination centre system shall maintain a list of priority premises to be evacuated for each town in the local area. This shall be updated by the local council when the coordination centre is established. (The premises list is maintained by the local government authority but may not be immediately available outside of normal working hours; While a central list may be out of date, it is better than nothing.)
3. The coordination centre system shall maintain a list of premises evacuated along with the time of evacuation and the units involved in the evacuation. (Allows units involved in the evacuation to be coordinated. Maintains an audit trail of who did what and when.)
4. The coordination centre system shall notify all liaison officers of new information about the threat situation as it becomes available. (Different services may respond differently to changes in the threat situation e.g. local government may withdraw from a situation because they are not equipped to deal with search and rescue.)

## 6 Related Work

Models of responsibility were first proposed by Blyth et al. in the ORDIT methodology [6,14], a graphical notation for describing the responsibilities that agents hold with respect to one another. Dobson and Strens have also explored the use of responsibilities in requirements engineering [15,16]. Dewsbury and Dobson have edited a collection of papers that describe much of the research undertaken on responsibility, presenting analyses of inappropriate responsibility allocation in socio-technical systems [17]. In particular, the work includes proposals for graphical responsibility modelling and pattern-based responsibility definition [9,13].

Responsibilities provide a basis for deriving system requirements from different perspectives and, in that respect, have something in common with viewpoint-based approaches to RE. Viewpoints were probably first proposed by Schoman and Ross in SADT under a different name [18] and the first use of the term ‘viewpoint’ was in Mullery’s paper on CORE [19]. CORE is based on information flow so is useful as a means of analyzing information requirements, once they have been discovered. Other researchers took on the general notion of viewpoints as a means to discover, organize and analyse requirements from different perspectives, using different approaches [20,21,22,23].

Responsibility-based approaches are perhaps closest to goal-based modelling approaches, such as  $i^*$  [24] and KAOS [25]. These are intended to expose high level dependencies between the goals associated with agents in a given system. Sub-goals may be derived from higher level objectives and assigned to agents for completion. Goals are achieved through the recursive achievement of some or

all sub-goals. Relationships between sub-goals express (and, or etc.) the possible ways in which the super-goal may be achieved. Analysis of such models can examine, for example, whether a super-goal may fail due to the failure of a single sub-goal (brittleness), or whether a particular agent has been overloaded with too many goals to achieve.

In a review of research on goal-oriented approaches, Lapouchnian [26] rightly states “Identifying goals is not an easy task”. He states that goals are normally derived from other information that is discovered from stakeholders. Responsibilities are, we believe, a more natural abstraction than goals for requirements discovery for a number of reasons:

1. Job specifications are often expressed in terms of responsibilities. Therefore, responsibilities are a common notion across all stakeholders in an enterprise. While managers and system operators (say) may not necessarily interpret responsibilities in the same way, they can both, at least, talk about them.
2. Business goals are often not effectively communicated to the workforce in an organization. Hence they may find it very difficult to distinguish between professional and personal goals. There is often widespread scepticism in organizations about business goals.
3. Goals are not, in our view, a natural way to express what we call ‘avoiding responsibilities’. They imply positive action to achieve the goal. A sentence such as ‘my goal is to ensure that the pressure does not become dangerously high; a much more likely way to express this is, ‘my responsibility is to ensure that the pressure does not become dangerously high’.

Responsibility modelling complements goal-based approaches by providing a higher level of abstraction for the modelling of socio-technical systems. Responsibility modelling could be used as a starting point for more concrete goal or task identification.

## 7 Evaluation

Evaluating any proposals for novel approaches to requirements engineering is difficult. There are practical and methodological problems in comparative evaluation, as the baseline knowledge of stakeholders changes as soon as they become involved in a trial. Even when requirements have been discovered, we have no way of telling, until the system has been implemented, if these requirements reflect the real needs of stakeholders. We may have developed an effective new way of discovering requirements but unless these are the right requirements, then our approach is not much use. We cannot therefore clearly demonstrate that focusing on responsibilities leads to better requirements. We can however say that responsibility modelling is a good way of stimulating requirements discussions and hence reduce the possibility that requirements are inappropriate.

We can consider the effectiveness of our responsibility-based approach from several perspectives:

*Naturalness: Can stakeholders without experience of requirements engineering relate to the approach?* In this respect, our responsibility-based approach scores highly. In emergency management, plans are usually expressed in terms of responsibilities so it is straightforward to understand the concept of a responsibility model. The notion of responsibility and responsible behaviour is widely used in everyday discourse so people can readily discuss their responsibilities in a given situation. The questions used to discover information requirements relate directly to the stakeholder's job and are therefore easy to understand. By contrast, notions such as viewpoints are not particularly easy to explain and, as already discussed, it is not easy to elicit stakeholder goals.

*Applicability: How widely applicable is an approach?* Because of the pervasive notion of responsibilities, we believe that this responsibility-based approach may be used in most socio-technical systems which involve multiple agencies or several units within the same organisation. So far, we have developed responsibility models in the domains of hospital administration, voting systems and emergency planning and are working on models of university administration.

*Scalability: Can the approach be used with real rather than simple example systems?* All RE methods suffer from the danger of information explosion where an immense amount of information is generated. In some cases, this makes it practically impossible to scale these up to practical systems. As our approach has been derived from our experience with large-scale socio-technical systems, we have never started with 'toy' examples; we have only ever used this to help us analyse existing, large-scale systems. Of course, the natural structure of responsibilities helps here - it is everyday practice to decompose and delegate responsibilities.

*User involvement: Have end-users users been involved in the research?* In terms of buy-in from end-users, we are currently working with the Scottish Environmental Protection Agency in developing our approach to responsibility models. They are particularly interested in developing simple support systems for use in the field, which reduce the problems of getting the right information to the right person and at the right time.

*Complementarity: Does the proposed approach complement other methods of requirements engineering?* We believe that a responsibility-based approach can be used alongside goal-based and viewpoint-based approaches. Responsibilities can be identified at an early stage in the process and used as a basis for discovering information requirements. By identifying the agents who hold responsibilities, stakeholders can be identified and involved in the RE process for the discovery of more detailed behavioural system requirements.

## 8 Conclusions

Relatively few complex information systems are now built from scratch but rather are assembled from existing and off-the-shelf systems. This means that

behavioural requirements of the system are constrained and that the key requirements are those that relate to the information provided to and generated by system stakeholders.

We have proposed an approach based on the notion of modelling responsibilities in an enterprise that supports the discovery of these information requirements. We argue that responsibilities are a natural abstraction to use in stakeholder communication and that stakeholders are readily able to articulate the information they require to discharge their responsibilities. Models of responsibility in a system provide a basis for stakeholders, potentially from different organizations, to discuss the information that the need to do their job. Hence, these models can support the discovery of information requirements.

We are in the process of using and evaluating these techniques to support emergency management. Here, a coordination system is required to ensure that information is exchanged between the different agencies involved in a timely way. Our initial discussions with end-users suggest that they can easily relate to a responsibility based approach and appreciate responsibility models as a succinct summary of who does what in emergency situations. They have proved to be an effective way of discovering and analysing information requirements.

## References

1. Suchman, L.A.: *Plans and Situated Actions: The Problem of Human-Machine Communication*. Cambridge University Press, Cambridge (1987)
2. Myers, M.D.: Investigating information systems with ethnographic research. *Communications of the Association for Information Systems* 23(2), 1–20 (1999)
3. Bentley, R., Hughes, J.A., Randall, D., Rodden, T., Sawyer, P., Shapiro, D., Sommerville, I.: Ethnographically-informed systems design for air traffic control. In: *Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW 1992)*, pp. 123–129. ACM Press, New York (1992)
4. Clarke, K., Hartswood, M., Procter, R., Rouncefield, M., Slack, R.: “minus nine beds”: Some practical problems of integrating and interpreting information technology in a hospital trust. In: *Proceedings of the BCS Conference on Healthcare Computing: Current Perspectives on Healthcare Computing*, Harrogate, UK, pp. 205–211 (2002)
5. Hughes, D.R.J.A., O’Brien, J., Rodden, T., Rouncefield, M., Sommerville, I., Tolmie, P.: Banking on the old technology: understanding the organisational context of legacy systems. *Communications of the Association for Information Systems* 2(8), Article 8 (1999)
6. Blyth, A.J., Chudge, J., Dobson, J.E., Strens, M.R.: ORDIT: A new methodology to assist in the process of eliciting and modelling organisational requirements. In: Kaplan, S. (ed.) *Proceedings on the Conference on Organisational Computing Systems*, Milpitas, California, USA, pp. 216–227. ACM Press, New York (1993)
7. Hardstone, G.: d’ Adderio, L., Williams, R.: Standardization, trust and dependability. In: [8], ch., 5, pp. 105–122
8. Clarke, K. (ed.): *Trust in Technology: A Socio-Technical Perspective*. Springer, Heidelberg (2006)
9. Sommerville, I.: Models for responsibility assignment. In: [17] ch., 8

10. Sommerville, I., Storer, T., Lock, R.: Responsibility modelling for contingency planning. In: Workshop on Understanding Why Systems Fail, Contingency Planning and Longer Term Perspectives on Learning from Failure in Safety Critical Systems (June 2007)
11. Lock, R., Storer, T., Harvey, N., Hughes, C., Sommerville, I.: Observations of the Scottish elections. *Transforming Government: People, Process and Policy* 2(2), 104–118 (2007)
12. Storer, T., Lock, R.: Modelling responsibility. Project Working Paper 7, InDeED Project (April 2008)
13. Sommerville, I.: Causal responsibility models. In: [17], ch., 9
14. Dobson, J.E., Blyth, A.J., Chudge, J., Strens, R.: The ORDIT approach to organizational requirements. In: Jirotko, M., Goguen, J.A. (eds.) *Requirements Engineering, Social and Technical Issues*, pp. 87–106. Academic Press, London (1994)
15. Dobson, J.E., Strens, M.R.: Responsibility modelling as a technique for requirements definition. *Intelligent Systems Engineering* 3(1), 20–26 (1994)
16. Dobson, J.E., Strens, M.R.: Organizational requirements definition for information technology systems. In: *Proceedings of the IEEE International Conference on Requirements Engineering (ICRE 1994)*, Colorado Springs, pp. 158–165. IEEE Press, Los Alamitos (1994)
17. Dewsbury, G., Dobson, J. (eds.): *Responsibility and Dependable Systems*. Springer, London (2007)
18. Ross, D.T., Schoman Jr., K.E.: Structured analysis for requirements definition. *IEEE Transactions on Software Engineering* 3(1), 6–15 (1977)
19. Mullery, G.P.: CORE: A method for controlled requirement expression. In: *Proceedings of the 4th International Conference on Software Engineering*, Munich, Germany, pp. 126–135. IEEE Computer Society Press, Los Alamitos (1979)
20. Finkelstein, A., Kramer, J., Nuseibeh, B., Finkelstein, L., Goedicke, M.: Viewpoints: A framework for integrating multiple perspectives in system development. *International Journal of Software Engineering and Knowledge Engineering* 2(1), 31–57 (1992)
21. Dardenne, A., Fickas, S., van Lamsweerde, A.: Goal-directed concept acquisition in requirements elicitation. In: *Proceedings of the Sixth International Workshop on Software Specification and Design*, pp. 14–21. IEEE Computer Society Press, Los Alamitos (1991)
22. Kotonya, G., Sommerville, I.: Requirements engineering with viewpoints. *BCS/IEEE Software Engineering Journal* 11(1), 5–18 (1996)
23. Sommerville, I., Sawyer, P.: Viewpoints: Principles, problems and a practical approach to requirements engineering. *Annals of Software Engineering* 3, 101–130 (1997)
24. Yu, E.S.: Towards modelling and reasoning support for early-phase requirements engineering. In: *3rd IEEE International Symposium on Requirements Engineering (RE 1997)*, pp. 226–235. IEEE Computer Society Press, Los Alamitos (1997)
25. Dardenne, A., van Lamsweerde, A., Fickas, S.: Goal-directed requirements acquisition. *Science of Computer Programming* 20, 3–50 (1993)
26. Lapouchnian, A.: Goal-oriented requirements engineering: An overview of the current research. Depth report, Department of Computer Science, University of Toronto (June 2005)

# Communication Analysis: A Requirements Engineering Method for Information Systems\*

Sergio España<sup>1</sup>, Arturo González<sup>2</sup>, and Óscar Pastor<sup>1</sup>

<sup>1</sup> Centro de Investigación en Métodos de Producción de Software  
Universidad Politécnica de Valencia

{sergio.espana, opastor}@pros.upv.es

<sup>2</sup> Departamento de Sistemas Informáticos y Computación

Universidad Politécnica de Valencia

agdelrio@dsic.upv.es

**Abstract.** Developing Information Systems (ISs) is a hard task for which Requirements Engineering (RE) offers a good starting point. ISs can be viewed as a support for organisational communication. Therefore, we argue in favour of communication-oriented RE methods. This paper presents Communication Analysis, a method for IS development and computerisation. The focus is put on requirements modelling techniques. Two novel techniques are described; namely, Communicative Event Diagram and Communication Structures. These are based on sound theory, they are accompanied by prescriptive guidelines (such as unity criteria) and they are illustrated by means of a practical example.

**Keywords:** Communication Analysis, requirements engineering, communication theory, enterprise information systems, requirements structure.

## 1 Introduction

Information Systems (ISs) development and computerisation is a wicked problem<sup>1</sup>.

To a large extent, this is due to their socio-technical nature [27] and to the intervention of multiple stakeholders with often conflicting needs and world views. To overcome conflicts and to reach agreement, stakeholders' perceptions have to be placed in a knowledge base that is shared with IS developers. Requirements Engineering (RE) facilitates this process by offering techniques for the discovery and description of stakeholders' needs. However, there exist many different explanations of what a requirement is and what it is not (e.g. what vs. how [10], why [35]). The authors' stance in this matter is summarised as follows (for more reasoned arguments see [20]):

- A requirements engineering method should prescribe a requirements structure that fits the problem trying to be solved, it should offer contingent and prescriptive methodological guidance and it should be illustrated with representative examples.

---

\* Research supported by the Spanish Ministry of Science and Innovation (MICINN) project SESAMO (TIN2007-62894), the MICINN FPU grant (AP2006-02323), and FEDER.

<sup>1</sup> Among other characteristics, wicked problems do not have a unique solution and their statement is not clear until they are solved (in part due to stakeholder discrepancies) [34].

- Requirements specifications should offer an external view of the system under development. In case internal details are included, the requirements structure should clearly differentiate the problem space (external) from the solution space (internal).

Attending to academic literature and industrial practice, various conceptions of ISs are found. Some authors consider ISs as a mere representation of reality [40]. Under this perspective, ISs can be described following an ontological approach; that is, focusing on the objects perceived in the universe of discourse (e.g. OO-Method [32]). Other perspectives focus on organisational intentions (e.g. Maps [35]), value object exchanges (e.g. e3-value [23]), etc. The authors view an IS as a support for organisational communications [26, 27]. Therefore, a communicational approach to ISs analysis is necessary; that is, we claim that ISs requirements engineering should take into account users' communicational needs.

We propose Communication Analysis as a method for the development and computerisation of enterprise Information Systems. This method focuses on communicative interactions that occur between the IS and its environment. The method stems from IS foundations academic research [22, 31] and it evolves by means of the collaboration with industry. Communication Analysis is currently being used by important Spanish enterprises and governmental institutions. The communicational perspective of the method has been overviewed in a previous publication [20]. This paper presents with greater detail the modelling techniques that Communication Analysis proposes for requirements specification. The main contributions of this paper are the following:

- Communicative Event Diagram is presented – a business process modelling technique that adopts a communicational perspective and facilitates the development of an IS that will support those business processes.
- Communication Structures are presented – a modelling technique for the specification of messages communicated with (and within) the organisation.
- Both modelling techniques fit well into a requirements structure, and they are both soundly founded on concepts borrowed from diverse disciplines (e.g. Systems Theory, Communication Theory, Information Systems Theory). Methodological guidance is based on sound criteria and illustrative examples are offered.

The rest of the article is structured as follows. Section 2 presents an overview of the approach, highlighting the proposed requirements specification structure and the method workflow. Section 3 describes the case that is used to illustrate the proposal. Section 4 describes Communication Analysis modelling techniques, paying special attention to Communicative Event Diagram and Communication Structures, and illustrating them. Section 5 presents a review of related works. Section 6 presents conclusions and future works.

## 2 Overview of the Approach

From a systemic point of view, the kind of problem that we are confronting involves at least three systems. The Organisational System (OS) is a social system that is interested in observing, controlling and/or influencing a portion of the world [27]. We



refer as Subject System (SS) to the portion of the world in which the OS is interested (a.k.a universe of discourse). An Information System (IS) is a socio-technical system, a set of agents of different nature that collaborate in order to support communication between the OS and its environment (and also within the OS) [27]. We refer as Computerised Information System (CIS) to the part of the IS that is automated.

Therefore, we argue that systemic principles need to be applied to IS requirements engineering. Quite often, the set of requirements are organised as plain enumerated lists. We claim that a requirements structure suited to ISs development is more appropriate than a list. Communication Analysis proposes a requirements structure that allows a stepwise refinements approach to ISs description (by following systemic principles). Also, the proposed method allows tackling with static and dynamic perceptions of reality (by giving support to discovering and describing that duality). Figure 1 shows the first dimension of the proposed requirements structure and the activities that are related to each requirements level. The structure and the method flow of activities have been overviewed in a previous publication [20].

The first dimension concerns several (systemic) requirements levels. *L1.System/subsystems* level refers to an overall description of the organisation and its environment (OS and SS, respectively) and also involves decomposing the problem in order to reduce its complexity. *L2.Process* level refers to business process description both from the dynamic viewpoint (by identifying flows of communicative interactions, a.k.a. communicative events) and the static viewpoint (by identifying business

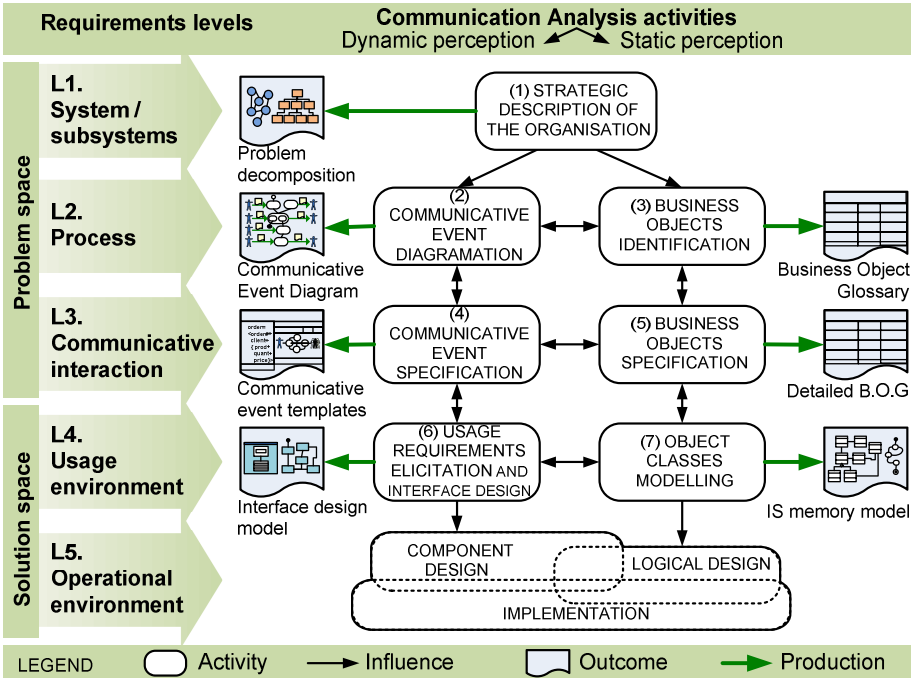


Fig. 1. Communication Analysis requirements levels and workflow

objects). *L3.Communicative interaction* level refers to the detailed description of each communicative event (e.g. the description of its associated message) and each business object. *L4.Usage environment* level refers to capturing requirements related to the usage of the CIS, the design of user interfaces, and the modelling of object classes that will support IS memory. *L5.Operational environment* level refers to the design and implementation of CIS software components and architecture.

Levels L1, L2 and L3 belong to the problem space, since they do not presuppose the computerisation of the IS and they aim to discover and describe the communicational needs of users. Levels L4 and L5 belong to the solution space, since they specify how the communicational needs are going to be supported. This paper focuses on the problem space and it describes in detail some of its modelling techniques.

### 3 Illustration Case Description

In order to exemplify the application of the method, we use an illustration case.

A photography agency manages illustrated reports (a.k.a. reports) and distributes them to publishing houses. Freelance photographers apply to work for the agency. The agency management board decides whether the photographer is accepted or not, and which quality level is assigned to them. Accepted photographers provide reports to the agency. Publishing houses buy reports from the agency catalogue and, sometimes, they request an exclusive report (a.k.a. exclusive) on a particular subject. Each exclusive is assigned to an interested photographer, as long as they do not have any other pending exclusive. Reports and exclusives are sent to publishing houses through a courier company, along with a delivery note. Then the messenger returns to the agency the delivery note, signed by the publishing house. Monthly, the agency issues publishing house invoices and photographers cheques.

## 4 Communication Analysis Modelling Primitives and Guidelines

Before describing Communication Analysis requirements levels, it is worth enumerating three functions of communication defined by Jakobson [25]. These functions allow us to structure requirements and to underpin the concepts underlying the method:

- Phatic: it aims to establish, maintain contact, and ensure operation of the (physical or psychological) communication channel between the addresser and the addressee.
- Referential: the purpose of this function is to convey context-related information.
- Connative: it aims to convey commands, to (attempt to) transform reality or people, to affect the course of events or behaviour of individuals.

### 4.1 L1. System/Subsystems Level

On the first requirements level, the analyst describes the OS from the strategic point of view. On the one hand, when the organisation is complex, it is advisable to decompose the problem into subsystems or organisational areas. On the other hand, the analyst elicits requirements related to strategic-level business indicators.

The photography agency case is of manageable size. Even though, three subsystems can be distinguished: Customer Service Department (it serves publishing houses), Production Department (it deals with photographers and manages reports), Accounting Department. With regard to strategic business indicators, the management board is interested in growth indicators that serve as a scorecard; e.g. increase in the number of photographers, increase in the number of exclusives, cash flow.

## 4.2 L2. Process Level

On this requirements level, Communication Analysis proposes describing business processes from a communicational perspective. The aim is to discover communicative interactions between the IS and its environment, and to describe them taking into account their dynamic and static aspects; that is, creating the Communicative Event Diagram and the Business Objects Glossary, respectively. In the following, a series of definitions clarify the concepts upon which the modelling techniques are built.

We refer as *communicative interaction* to an interaction between actors with the aim of exchanging information. FRISCO report [16] presents a generic model of ISS that considers an IS as a support for communicative interactions. In a previous publication, the authors extend this model in order to deepen the communicative point of view [31]. Depending on the main direction of communication, the following types of communicative interactions can be distinguished:

- *Ingoing communicative interactions* primarily feed the IS memory with new meaningful information. These interactions often appear in the shape of business forms.
- *Outgoing communicative interactions* primarily consult IS memory. These interactions often appear in the shape of business indicators, listings and printouts.

Industrial experience has shown us that ingoing communicative interactions entail more analytical complexity. Therefore, we advise the analyst to focus, first of all, on ingoing communicative interactions<sup>2</sup>.

A *communicative event* is a set of actions related to information (acquisition, storage, processing, retrieval and/or distribution), which are carried out in a complete and uninterrupted way, on the occasion of an external stimulus [22].

Communication Analysis offers *unity criteria* to allow identifying communicative events, also facilitating the determination of their granularity. This way, a communicative event can be seen as an ingoing communicative interaction that fulfils the unity criteria. Each unity criterion is related to a communication function. Table 1 summarises unity criteria and their application, see [21] for detailed information.

Communication Analysis proposes to specify the flow of communicative events by means of the **Communicative Event Diagram** (CED). The primitives of this modelling technique are shown at the bottom of Figure 2 and explained next.

---

<sup>2</sup> Communication Analysis also takes into account outgoing communicative interactions. However, when the OS needs complex indicators for performance management, techniques such as the Balanced Scorecard are recommended.

**Table 1.** Unity criteria to identify and encapsulate communicative events

<b>Criterion (Communication function) Definition</b>	
<b>Trigger unity</b>	(Phatic function) Trigger responsibility is external. The event occurs as a response to an external interaction and, therefore, some actor triggers it. This (primary) actor is the one that provides the information that is conveyed in the event.
<b>Communication unity</b>	(Referential function) Each and every event involves providing new meaningful information. Thus, an interaction needs to provide new facts in order to be considered an event. Input messages are representations of something that happens in the IS environment.
<b>Reaction unity</b>	(Connative function) The event is a composition of synchronous activities; thus, these activities can communicate the information they need from each other. Events are asynchronous among each other; thus, events need a shared IS memory to communicate.
<b>An example of their application.</b> According to the unity criteria, two communicative events are identified with regard to photographer subscriptions: PHO 1 and PHO 3 (see Figure 2). Both events fulfil the three unity criteria: both have an external actor that triggers them (a photographer and the management board, respectively), both result in the provision of new meaningful information (the application and the resolution, respectively), and both are compositions of synchronous activities. Considering them to be only one communicative event would result in violating the trigger criteria (each has a different primary actor) and the reaction criteria (they <i>are</i> asynchronous: PHO 1 can occur at any moment during office hours, PHO 3 occurs Monday mornings).	

Each *communicative event* is represented as a rounded rectangle and is given an identifier and a descriptive name. The *identifier* serves for traceability purposes and it is usually a code composed of a mnemonic (related to the system to which the event is ascribed) and a number (e.g. PHO 3). With regard to the *name*, we recommend to consistently use either an external nomination (primary actor + action + object + qualifier; e.g. “Photographer submits an application”) or an internal nomination (support actor + action + object + qualifier; e.g. “Clerk receives a photographer application”). For instance, in the illustration case we have opted for an external nomination. For each event, involved actors are identified. Communication Analysis distinguishes several roles (see theoretical basis in [31]):

- The *primary actor* triggers the communicative event by establishing contact with the OS and provides the conveyed input information. Therefore, primary actors are modelled as senders of ingoing communicative interactions. For instance, the management board is the primary actor of event PHO 3.
- The *support actor* is in charge of physically interacting with the IS interface in order to encode and edit input messages. Support actors are specified at the bottom of the event rounded rectangle. Sometimes the primary actor and the support actor are different persons (e.g. photographer and clerk, respectively, in event PHO 1). Other times both roles are played by the same person (e.g. the salesman in event PHO 2).

- *Receiver actors* are those who need to be informed of the occurrence on an event. In order to truly understand the meaning of messages in organisations, it is necessary to analyse these actors. They are modelled as receivers of outgoing communicative interactions (e.g. in PHO 3 the photographer is informed of the resolution).
- *Reaction processors* are those in charge of performing the IS reaction to the message. This role is not depicted in the CED.

The messages associated to communicative events are conveyed via *incoming communicative interactions* and *outgoing communicative interactions*. In the CED,

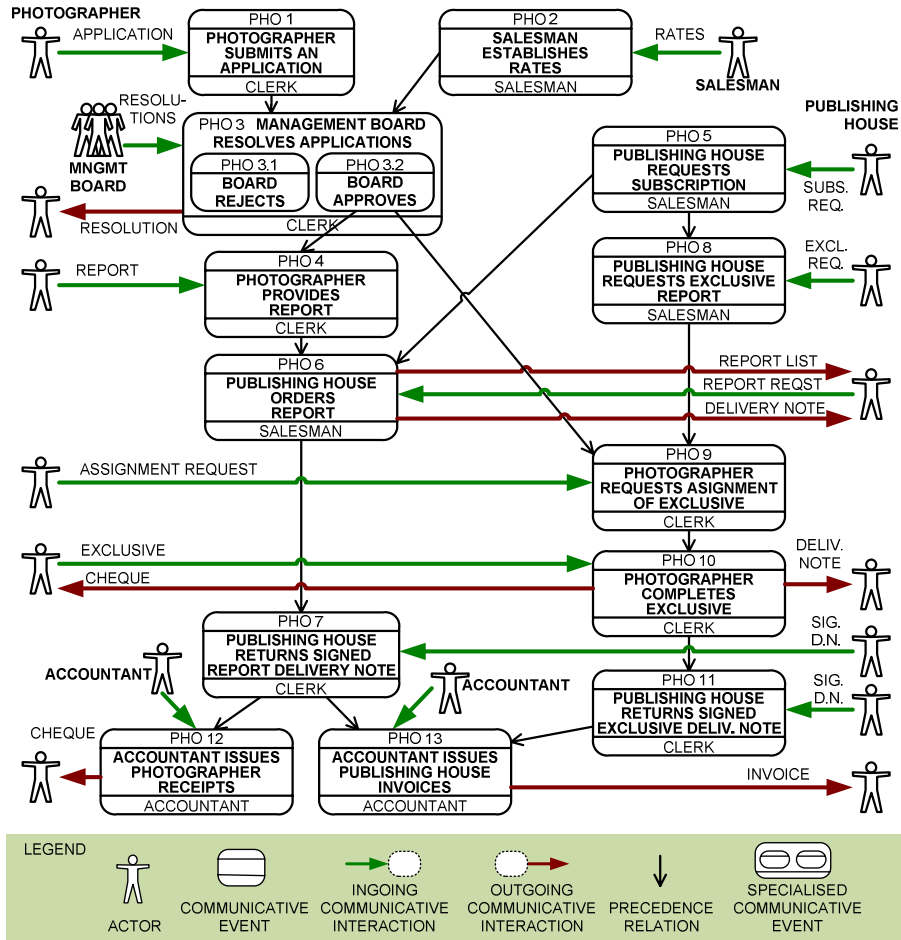


Fig. 2. Communicative Event Diagram of the photography agency<sup>3</sup>

<sup>3</sup> Note that the labels of Photographer actors (to the right) and Publishing house actors (to the left) are omitted for reasons of space. Also, some communicative interaction labels have been abbreviated (e.g. SIG.D.N. stands for SIGNED DELIVERY NOTE).

messages are given a name (by labelling communicative interactions)<sup>4</sup>. Communicative interactions are modelled as arrows placed in the horizontal axis. The vertical axis is reserved for *precedence relations* among communicative events, which are also modelled as arrows<sup>5</sup>. E.g. PHO 3 requires the previous occurrence of PHO 1 and PHO 2.

Communicative events are specialised whenever each specialised variant leads to a different temporal path (i.e. distinct precedence relations). It must be avoided specialising an event as a result of different communication channels, since the message remains the same (e.g. a publishing house can order a report in person or by telephone).

This requirements level also provides a static perspective of business processes, by means of business objects. We refer as *business objects* to the conceptions of those entities of the Subject System in which the OS is interested. Frequently, stakeholders describe business objects as complex aggregates of properties. Business objects are identified and described in a **Business Object Glossary**. Also, users are asked to hand out business forms to the analysts, who catalogue them for later form analysis. For instance, the photography agency manages the following business objects: photographer records, publishing house records and reports<sup>6</sup>. Static analysis also implies eliciting business indicators that are associated to subsystems or processes. For instance, the photography agency is interested in business indicators related to its three subsystems:

- Customer Service Department requires payments and takings indicators that allow them monitoring debts (e.g. publishing house indebtedness).
- Production Department requires productivity and profitability indicators (e.g. delivery performance to customer, photographer productivity).
- Customer Service Department requires client-related indicators that allow them to monitor customer loyalty (e.g. consumption rates).

### 4.3 L3. Communicative Interaction Level

Communicative events that appear in the CED need to be described in detail. Requirements associated to an event are structured by means of an **Event Specification Template**. The template is composed by a header and three categories of requirements: contact, communicational content and reaction requirements. These categories are related to phatic, referential and connative communication functions, respectively.

The *header* contains general information about the communicative event; that is, the event identifier, its name, a narrative description and, optionally, an explanatory diagram. The event identifier and name come from the CED; event identification needs to be kept consistent throughout the entire analysis and design specification in order to enhance requirements traceability. Since requirements specifications is meant, first of all, to facilitate problem understanding, a narrative description of the event is strongly advised. Also, whenever the event is complex, an explanatory diagram illustrating its associated flow of tasks shall be included.

<sup>4</sup> Message structure is specified in detail in a later activity (see Section 4.3).

<sup>5</sup> Complex business processes may require other operators. Start and end symbols can also be used. Besides, loops appear in many business processes.

<sup>6</sup> The description of business objects is omitted for the sake of brevity.

**Table 2.** Primitives and grammar of Communication Structures modelling technique

CSs primitives	EBNF grammar for Communication Structures <sup>7</sup>
<b>Aggregation</b> <b>A = &lt; a + b + c &gt;</b> A is composed of fields <b>a</b> and <b>b</b> and <b>c</b> .	communication structure = structure name, '=', initial substructure; initial substructure = aggregation substructure   iteration substructure; aggregation substructure = '<', substructure list, '>'; iteration substructure = '{', substructure list, '}'; specialisation structure = '[' , substructure list, { ' ', substructure list }, ']' ; substructure list = substructure, { '+', substructure } ; complex substructure = aggregation substructure   iteration substructure   specialisation structure;
<b>Alternative</b> <b>A = [ a   b   c ]</b> A is either composed of field <b>a</b> or <b>b</b> or <b>c</b> , (only one of them).	substructure = substructure name, '=', complex substructure   identifier field   field;
<b>Iteration</b> <b>A = { B }</b> A is composed of several substructures of type <b>B</b> .	substructure = substructure name, '=', complex substructure   identifier field   field;
<b>Identification</b> <b>a( id )</b> Field <b>a</b> identifies an object that is already known by the IS.	substructure = substructure name, '=', complex substructure   identifier field   field;

*Contact requirements* are related to the conditions that are necessary in order to establish communication. For instance<sup>8</sup>, the primary actor, possible communication channels (e.g. fax, email, in person), availability and temporal restrictions (e.g. office hours for order reception), authentication requirements (e.g. in Spain, bureaucratic proceedings often require showing an identity card).

*Communicational content requirements* specify the message conveyed in an event and related restrictions (e.g. reliability: certifying that a diploma provided by a student is not fraudulent). With regard to the message, both metalinguistic aspects (e.g. message field structure, optionality of fields) and linguistic aspects (e.g. field domains, example values) need to be specified. Communication Analysis proposes a message modelling technique. **Communication Structure** (CS) is a modelling technique that is based in structured text and allows specifying the message associated to a communicative event. The structure of message fields lies vertically and many other details of the fields can be arranged horizontally; e.g. the information acquisition operation, the field domain, the link with the business object, an example value provided by users. A communicative event can not be fully understood until its CS is defined in detail. Specifying with precision an event CS forces and helps analysts and users to appropriately mark the event boundary and meaning. Table 2 shows the Communication Structures grammar. On left-hand side column, the primitives are informally

<sup>7</sup> This table summarises the main syntactical rules of the grammar. The elements structure name, substructure name, identifier field and field can be considered character strings.

<sup>8</sup> For reasons of space, not all kinds of requirements in each category are included.





rest of the fields are derived from the IS memory (these data is introduced by a previous event; namely, PHO 1 Photographer submits an application). The business object column links dynamic perspective (communicative interaction description) with the static perspective (Business Object Glossary). Note that only new facts are stored in the IS memory. Example values enhance user-analyst communication.

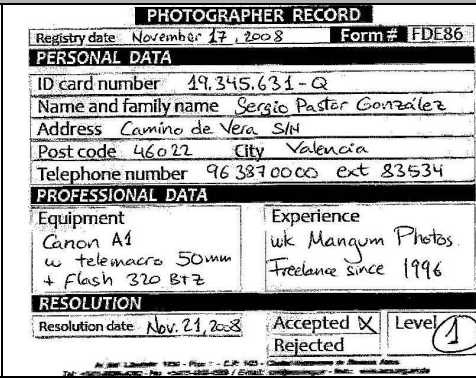
Reaction requirements	
<p><b>Business object:</b> If the application is accepted, the photographer becomes part of the agency. The clerk creates a photographer record that includes photographer’s personal and contact details (See scanned form at the right-hand side).</p> <p><b>Outgoing communicative interaction:</b> After this communicative event, a letter informing of the resolution is sent to the photographer<sup>9</sup>.</p>	 <p>The image shows a scanned form titled "PHOTOGRAPHER RECORD" with the following sections:</p> <ul style="list-style-type: none"> <li><b>Registry date:</b> November 17, 2008</li> <li><b>Form #:</b> FDE86</li> <li><b>PERSONAL DATA</b> <ul style="list-style-type: none"> <li>ID card number: 49.345.631-Q</li> <li>Name and family name: Sergio Pastor Gonzalez</li> <li>Address: Camino de Vera, S/N</li> <li>Post code: 46022 City: Valencia</li> <li>Telephone number: 96 387 0000 ext 83534</li> </ul> </li> <li><b>PROFESSIONAL DATA</b> <ul style="list-style-type: none"> <li>Equipment: Canon A1 w telemacro 50mm + flash 320 Btz</li> <li>Experience: wk Managum Photos Freelance since 1996</li> </ul> </li> <li><b>RESOLUTION</b> <ul style="list-style-type: none"> <li>Resolution date: Nov. 21, 2008</li> <li>Accepted: <input checked="" type="checkbox"/></li> <li>Rejected: <input type="checkbox"/></li> <li>Level: 1 (circled)</li> </ul> </li> </ul>

Fig. 3. Event Specification Template of communicative event PHO 3

## 5 Related Works

There exist distinct orientations with regard to requirements elicitation for IS development. *Goal-oriented approaches* intend to identify stakeholders’ necessities, modelling them as goals, where a “a goal is an objective the system under consideration should achieve” [39]. E.g. Map [35], i\* [43] and KAOS [9]. Among *agent-oriented approaches*, which design the system as a set of autonomous and automatable agents, Tropos includes a goal-oriented requirements stage [6]. *Usage-based approaches* describe the interaction between the user and the software system under development. E.g. Use Cases [30] and Info Cases (an extension of the former) [18]. *Value-oriented approaches* identify and model value object exchanges [41]. E.g. e3-value [23]. *Aspect-oriented approaches* apply the separation of concerns principle [14] to RE. E.g. Early Aspects [33] and Theme/DOC [3]. Some organisational modelling approaches propose modelling and integrating multiple views of the system [5, 36, 11]. There also exist *communicational approaches*. In this field, a widely extended orientation is the Language Action Paradigm (LAP) [17, 42], which is mainly based on the work of Austin [1] and Searle’s speech act classification [37]. Communicative Action Paradigm (CAP) is an evolution of LAP that extends the paradigm to non-verbal communication [13]. Several approaches stem from LAP, such as Action Workflow [28], SAMPO [15], Business Action Theory [19], DEMO [12] and Cronholm and Goldkunhl’s Communication Analysis [8]. SANP [7] adopts a similar approach, but it is based on Ballmer and Brennenstuhl’s speech act classification [2]. *Semiotic approaches* to organisational modelling have also been proposed [38].

<sup>9</sup> This communicative interaction is a printout and it is not described in detail for the sake of brevity.

This paper presents a communicational approach. Communication Analysis does not consider goal modelling, but the industrial projects in which the authors have been involved (which have contributed to consolidate the method) did not require it. Likewise, value network modelling has not been considered, but the method is usually put into practice in existing organisations with well established businesses, not with the intention to support an “innovative e-commerce idea” [23]. In any case, IS development is better tackled with a contingent approach, so we are open to integrating these or other perspectives with Communication Analysis (see Section 6).

Some features give advantage to our proposal over other methods. The actor roles argued in Section 4.2 distinguish Communication Analysis from other approaches. Many business process modelling techniques use support actors and/or reaction processors as criteria for organising processes in swimlanes but primary actors are disregarded (e.g. [11]). However, primary actors are central to our approach<sup>10</sup>. Furthermore, most RE approaches do not specify communicational content of interactions (or message specification is mixed with system usage description, as in Use Cases). Info Cases is an exception, since this technique proposes a structured text specification of messages. However, Communication Structures have greater expressiveness (e.g. alternative, iteration, information acquisition operation)<sup>11</sup>.

With regard to communicational approaches, we share with them the communicational perspective and many foundations borrowed from Communication Theory. However, Communication Analysis does not necessarily preconceive a specific speech acts classification nor assumes conversational patterns, as LAP-based approaches do. An analyst following our approach does not impose patterns on the organisation, but confines to discovering the communication needs of the organisational stakeholders, shedding light on their work practice and identifying possible improvements. Our proposal coincides with the one by Cronholm and Goldkuhl in the communicational perspective. Also, both proposals consider organisational documents (e.g. business forms) invaluable sources of information. However, Cronholm and Goldkuhl choose existing documents as a starting point in RE process and their modelling notation (the Document Activity Diagram) is document-centred; that is, communicative interactions are subordinate to documents. We choose communicative interactions as a starting point and the modelling notation that we propose in this paper (the Communicative Event Diagram) is communicative interaction-centred. We argue that communicative events represent pure work practice and that it is possible to discover and describe them independently of their associated documents. Documents are a specific technological support<sup>12</sup> (solution space) for a communicative event (problem space); that is, documents are the result of a previous IS implementation.

With regard to the conceptual framework for understanding business processes by Melão and Pidd [29], our proposal combines two of the four perspectives; namely the constructivist and mechanistic perspectives.

---

<sup>10</sup> Organisations can replace many support agents with computer interfaces (e.g. clerks vs. web-based forms) and IS reaction processors are typically automated. Primary actors, however, are irreplaceable because they are the ultimate source of information.

<sup>11</sup> Making an in-depth comparison of our proposal with regard to other methods (e.g. feature comparison, performance evaluations) can not be dealt with in one single paper; we are currently working on empirical validation and results will be available as part of future works.

<sup>12</sup> We do not necessarily refer to computer technology; paper is an ancient form of technology.

- In order to discover business processes, a constructivist stance is adopted: business processes are considered a social construct that is agreed among stakeholders, and the requirements engineer acts as facilitator in this agreement.
- In order to describe business processes, a mechanistic stance is adopted: the use of models that are based on actors, events and messages allows creating a requirements specification that serves as a starting point for later software design.

Discovering requirements allows their description and, conversely, IS description provides feedback to discovery by allowing new interactions with stakeholders (e.g. to formulate new questions). Both perspectives are intertwined. The combination of hard (mechanistic) and soft (constructivist) approaches is not new to the ISs scene [4], but Communication Analysis contributes a requirements structure and communication-oriented modelling techniques that do not appear in previous proposals.

One last remark. The proposed business process modelling technique consists of a set of interrelated concepts, criteria plus other methodological guidance, and a notation. We believe that, in general, it is concepts and criteria that matter the most (not notations). We also acknowledge that some practitioners would be more comfortable using other notations for business process modelling (e.g. BPMN, Activity Diagrams, Use Cases). There is no problem with that, as long as the notation is adapted to support the communicational perspective. In fact, this has been done before.

## 6 Conclusions and Future Work

To sum up, Communication Analysis is an Information Systems (ISs) development method that proposes a flow of activities and a requirements structure. It is founded on Systems Theory and Communication Theory, among other scientific fields. An overview of Communication Analysis can be found elsewhere [20]. This paper focuses on the requirements elicitation stage and describes in detail several communication-based modelling techniques. The Communicative Event Diagram specifies business processes from a communicational point of view. In order to guide the analyst in identifying and determining the proper granularity of communicative events, unity criteria are proposed. Each communicative event is later specified by means of a template. Messages associated to communicative events are specified by means of Communication Structures, a notation based on structured text. The approach is exemplified using an illustration case (a photography agency).

Communication Analysis is currently being applied to big projects in industrial environments; e.g. the integration of Anecoop S.Coop (a Spanish major distributor of fruit and vegetables) with its associated cooperatives (>100). We plan to describe our industrial experience by means case study reports. Laboratory experiments have been carried out to test the benefits of the unity criteria; resulting models correction and data analysis is now being undertaken. Future work also involves developing a CASE tool that supports the method, researching how other perspectives (e.g. goal or value orientation) may extend our approach under certain project circumstances, and the integration of Communication Analysis and the OO-Method, an MDA-based method with software generation capabilities.

## References

1. Austin, J.L.: How to do things with words. Oxford University Press, Oxford (1962)
2. Ballmer, T.T., Brennenstuhl, W.: Speech act classification: A study of the lexical analysis of English speech activity verbs. Springer, Berlin (1981)
3. Baniassad, E., Clarke, S.: Theme: an approach for aspect-oriented analysis and design. In: 26th International Conference on Software Engineering (ICSE 2004), pp. 158–167. IEEE Computer Society Press, Los Alamitos (2004)
4. Brown, J., Cooper, C., Pidd, M.: A taxing problem: the complementary use of hard and soft OR in the public sector. *Eur. J. Oper. Res.* 172(2), 666–679 (2006)
5. Bubenko, J.A., Brash, D., Stirma, J.: EKD User Guide. Dept. of Computer and Systems Science tech. report, Stockholm University (1998)
6. Castro, J., Kolp, M., Mylopoulos, J.: Towards requirements-driven information systems engineering: the Tropos project. *Information Systems* 27, 365–389 (2002)
7. Chang, M.K., Woo, C.C.: A speech-act-based negotiation protocol: design, implementation, and test use. *ACM Trans. Inf. Syst.* 12(4), 360–382 (1994)
8. Cronholm, S., Goldkuhl, G.: Communication Analysis as perspective and method for requirements engineering. In: Mate, J.L., Silva, A. (eds.) Requirements engineering for socio-technical systems, pp. 340–358. Idea Group Inc. (2004)
9. Dardenne, A., van Lamsweerde, A., Fickas, S.: Goal-directed requirements acquisition. *Sci. Comput. Program.* 20(1-2), 3–50 (1993)
10. Davis, A.M.: Software Requirements: Analysis and Specification. Prentice-Hall, Englewood Cliffs (1990)
11. de la Vara, J.L., Sánchez, J., Pastor, O.: Business process modelling and purpose analysis for requirements analysis of information systems. In: Bellahsene, Z., Léonard, M. (eds.) CAiSE 2008. LNCS, vol. 5074. Springer, Heidelberg (2008)
12. Dietz, J.L.G.: Understanding and modelling business processes with DEMO. In: Akoka, J., Bouzeghoub, M., Comyn-Wattiau, I., Métais, E. (eds.) ER 1999. LNCS, vol. 1728, pp. 188–202. Springer, Heidelberg (1999)
13. Dietz, J.L.G., Goldkuhl, G., Lind, M., van Reijswoud, V.E.: The Communicative Action Paradigm for business modelling - a research agenda. In: 3rd International Workshop on the Language Action Perspective on Communication Modelling (LAP 1998). Jönköping International Business School (1998)
14. Dijkstra, E.W.: A discipline of programming. Prentice-Hall, Englewood Cliffs (1976)
15. Esa, A., Lehtinen, E., Lyytinen, K.: A speech-act-based office modeling approach. *ACM Trans. Inf. Syst.* 6(2), 126–152 (1988)
16. Falkenberg, E., Hesse, W., Lindgreen, P., Nilsson, B., Oei, J.L.H., Rolland, C., Stamper, R., Van Assche, F., Verrijn-Stuart, A., Voss, K.: FRISCO. A Framework of Information Systems Concepts. IFIP WG 8.1 Task Group Report (1998)
17. Flores, F., Ludlow, J.: Doing and speaking in the office. In: Fick, G., Sprague, R.H. (eds.) Decision Support Systems: issues and challenges, NY, USA, pp. 95–118. Pergamon Press, Oxford (1980)
18. Fortuna, M., Werner, C., Borges, M.: Info Cases: integrating use cases and domain models. In: 16th International Requirements Engineering Conference (RE 2008), Barcelona, Spain, pp. 81–84. IEEE, Los Alamitos (2008)
19. Goldkuhl, G.: Generic business frameworks and action modelling. In: International workshop on the Language Action Perspective on Communication Modelling (LAP 1996). Tilburg, The Netherlands (1996)

20. González, A., España, S., Pastor, O.: Towards a communicational perspective for enterprise Information Systems modelling. In: IFIP WG 8.1 Working Conference on the Practice of Enterprise Modeling (PoEM 2008), Stockholm, Sweden. LNBIP, vol. 15, pp. 63–77. Springer, Heidelberg (2008)
21. González, A., España, S., Pastor, O.: Unity criteria for Business Process Modelling: a theoretical argumentation for a Software Engineering recurrent problem. In: 3rd Intl. Conf. on Research Challenges in Information Science (RCIS 2009), Fes, Morocco. IEEE, Los Alamitos (2009)
22. González, A.: Algunas consideraciones sobre el uso de la abstracción en el análisis de los sistemas de información de gestión. Ph.D thesis (in Spanish). Departamento de Sistemas Informáticos y Computación. Universidad Politécnica de Valencia (2004)
23. Gordijn, J., Wieringa, R.J.: A value-oriented approach to e-business process design. In: Eder, J., Missikoff, M. (eds.) CAiSE 2003. LNCS, vol. 2681, pp. 390–403. Springer, Heidelberg (2003)
24. ISO/IEC 14977: Information technology - Syntactic metalanguage - Extended BNF (1996)
25. Jakobson, R.: The speech event and the functions of language. In: Monville-Burston, M., Waugh, L.R. (eds.) On language, pp. 69–79. Harvard University Press, Cambridge (1990)
26. Langefors, B.: Theoretical analysis of Information Systems, 4th edn. Studentlitteratur, Lund (1977)
27. Lockemann, P.C., Mayr, H.C.: Information System Design: Techniques and Software Support. In: Kugler, H.-J. (ed.) IFIP 1986, North-Holland, Amsterdam (1986)
28. Medina-Mora, R., Winograd, T., Flores, R., Flores, F.: The action workflow approach to workflow management technology. In: ACM conference on Computer-Supported Cooperative Work (CSCW 1992), Toronto, Ontario, Canada, pp. 281–288. ACM, New York (1992)
29. Melão, N., Pidd, M.: A conceptual framework for understanding business processes and business process modelling. *Inform. Syst. J.* 10(2), 105–129 (2000)
30. OMG: Unified Modeling Language: Superstructure version 2.0, <http://www.omg.org/docs/formal/05-07-04.pdf> (accessed 11, 2008) (2005)
31. Pastor, O., González, A., España, S.: 31. Pastor, O., González, A., España, S.: Conceptual alignment of software production methods. In: Krogstie, J., Opdahl, A.L., Brinkkemper, S. (eds.) Conceptual modelling in Information Systems engineering, pp. 209–228. Springer, Berlin (2007)
32. Pastor, O., Molina, J.C.: Model-Driven Architecture in practice: A Software Production Environment Based on Conceptual Modeling. Springer, New York (2007)
33. Rashid, A., Sawyer, P., Moreira, A., Araújo, J.: Early Aspects: a model for aspect-oriented requirements engineering. In: 10th Anniversary IEEE Joint International Conference on Requirements Engineering, pp. 199–202. IEEE Computer Society, Los Alamitos (2002)
34. Rittel, H., Webber, M.: Dilemmas in a general theory of planning. *Policy Sciences* 4, 155–169 (1973)
35. Rolland, C.: Capturing system intentionality with Maps. In: Krogstie, J., Opdahl, A.L., Brinkkemper, S. (eds.) Conceptual modelling in Information Systems engineering, pp. 141–158. Springer, Heidelberg (2007)
36. Scheer, A.-W.: ARIS - Business Process Modeling, 3rd edn. Springer, New York (2000)
37. Searle, J.R., Vanderveken, D.: Foundations of illocutionary logic. Cambridge University Press, Cambridge (1985)
38. Stamper, R.K.: Organizational semiotics. In: Stowell, F., Mingers, J. (eds.) Information Systems: an emerging discipline, London, pp. 267–283. McGraw Hill, New York (1997)

39. van Lamsweerde, A.: Goal-oriented Requirements Engineering: a guided tour. In: 5th IEEE International Symposium on Requirements Engineering (RE 2001), Toronto, Canada, pp. 249–262. IEEE Computer Society Press, Los Alamitos (2001)
40. Wand, Y., Weber, R.: On the deep structure of information systems. *Inf. Syst. J.* 5, 203–223 (1995)
41. Weigand, H., Johannesson, P., Andersson, B., Bergholtz, M., Edirisuriya, A., Ilayperuma, T.: On the notion of value object. In: Dubois, E., Pohl, K. (eds.) CAiSE 2006. LNCS, vol. 4001, pp. 321–335. Springer, Heidelberg (2006)
42. Winograd, T., Flores, F.: *Understanding computers and cognition: A new foundation for design.* Addison-Wesley, Reading (1987)
43. Yu, E., Mylopoulos, J.: From E-R to "A-R" - Modelling strategic actor relationships for business process reengineering. In: Loucopoulos, P. (ed.) ER 1994. LNCS, vol. 881, pp. 548–565. Springer, Heidelberg (1994)

# Spectrum Analysis for Quality Requirements by Using a Term-Characteristics Map

Haruhiko Kaiya<sup>1,2</sup>, Masaaki Tanigawa<sup>1</sup>, Shunichi Suzuki<sup>1</sup>,  
Tomonori Sato<sup>1</sup>, and Kenji Kaijiri<sup>1</sup>

<sup>1</sup> Dept. of Computer Science, Shinshu University, Nagano 380-8553, Japan  
kaiya@cs.shinshu-u.ac.jp

<sup>2</sup> GRACE Center, National Institute of Informatics (NII), Tokyo 101-8430, Japan  
<http://grace-center.jp/>

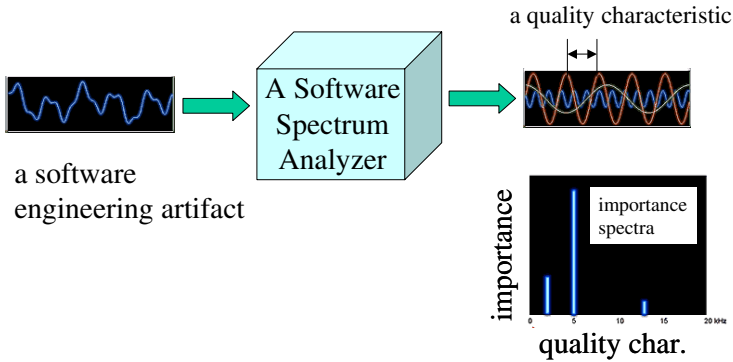
**Abstract.** Quality requirements are scattered over a requirements specification, thus it is hard to measure and trace such quality requirements to validate the specification against stakeholders' needs. We have already proposed a technique called "spectrum analysis for quality requirements" which enables analysts to sort a requirements specification to measure and track quality requirements in the specification. However current spectrum analysis largely depends on expertise of each analyst, thus it takes a lot of efforts to perform the analysis and is hard to reuse experiences for such analysis. We introduce domain knowledge called term-characteristic map (TCM) to improve current spectrum analysis for quality requirements. Through several experiments, we evaluated the improved spectrum analysis.

**Keywords:** Requirements Analysis, Quality Requirements, Non-functional Requirements.

## 1 Introduction

Software quality requirements of a system are specifications for defining how well functions of the system are accomplished. Defining quality requirements has more problems than defining functional ones, and there was a special issue about quality requirements in IEEE Software. In its guest editors' introduction [3], the following three problems are mentioned: implicit understanding of quality requirements by stakeholders, trade-offs among quality requirements and difficulty of measuring and tracking quality requirements.

There are several techniques for resolving one or more problems above, and we proposed a simple and general technique called "spectrum analysis for quality requirements" [15] for measuring and tracking quality requirements. A wave such as sound or light can be decomposed into several regular (or sine) waves each of which has different cycle (or wavelength) and power (or amplitude). Spectrum analysis in optics is based on this fact. In spectrum analysis for quality requirements, a quality characteristic such as suitability, accuracy, and interoperability is regarded as wavelength, and the power of the characteristic as its importance as shown in Figure 1. By using a quality requirements spectrum of a system, stakeholders can identify relative attention to quality requirements in a software engineering artifact such as a requirements specification



**Fig. 1.** Basic idea of spectrum analysis for quality requirements

or a design document. Such relative attention enables stakeholders to validate quality requirements defined in such software engineering artifact. Suppose a power of security is larger than one of usability in a quality spectrum of a requirements document for a system. If a stakeholder regards usability is more important than security, he can easily suspect one of his quality requirements could not be reflected in the document.

There are two systematic comparative analyses for quality spectrum analysis. One is comparison among spectra of similar systems to identify mandatory and optional quality characteristics. There are a lot of similar systems for each application domain, e.g., a lot of web browsers, painting tools, e-learning systems and so on. Systems in the same domain usually have similar quality spectrum, and such similarity shows mandatory quality requirements in such a domain [15]. On the other hand, differences among spectra of the systems in the same domain show optional or specific features of each system. Although different segments or different price ranges of the same domain do not always have similar spectrum, we may regard each segment or each range as a sub-domain and may compare spectra of parts in a segment or spectra systems in a range with each other. Another is comparison among spectra of a system in different development phases, e.g., requirements, design, implementation and so on. Quality requirements should be inherited along the progress of development, but it is not easy to track such inheritance during the progress of such development. Quality requirements spectrum enables developers to track such inheritance.

However, there is a serious problem in the current quality spectrum analysis [15]. As shown in Figure 2, the power of each quality characteristic is calculated based on the number of relationships between an element of an artifact, e.g., a sentence in a requirements document, and each quality characteristics. Making such relationships largely depends on the expertise and subjective decision of an analyst. Therefore, it takes a lot of efforts to perform quality spectrum analysis and is hard to perform the analysis (semi-) automatically. In this paper, we will introduce an improved version of quality spectrum analysis for resolving this problem. In addition, we show a prototype of a CASE tool that supports quality spectrum analysis.

The rest of this paper is organized as follows. In the next section, we briefly explain the original quality spectrum analysis method (called “the old method” in this



paper) [15], and clarify its problem. We then introduce the improved method (called “TCM method” in this paper) by using domain knowledge called term-characteristic map (TCM). In section 3 we evaluate TCM method with respect to the following three points: results of TCM method inherit those of the old method, domain specific spectrum can be found by using TCM method and results of TCM method are objective, i.e., results of TCM method do not depend on analyst’s subjective decision. In section 4 we will show a supporting tool to perform TCM method. Finally, we briefly review related works, summarize our current results and show the future issues.

## 2 A Method for Generating Quality Spectrum

### 2.1 The Old Method and Its Problems

As mentioned in introduction, we call the procedure to generate quality spectrum in our previous paper [15] as “the old method” in this paper. Figure 2 shows a typical application of the old method. Inputs of the old method are a list of requirements and a list (catalog) of quality characteristics. In Figure 2, five requirements are listed in the list and quality factors in ISO9126 [14] are used for the catalog. ISO9129 contains one of the famous catalogs of quality characteristics. Such kind of catalogs helps requirements analysts to find missing quality requirements. A quality model in ISO9126 categorizes software quality attributes into six characteristics (functionality, reliability, usability, efficiency, maintainability and portability), which are further subdivided into subcharacteristics such as resource efficiency, changeability and so on in Figure 2.

An analyst then makes relationships between a requirement and a characteristic subjectively. To make such relationships, the analyst takes into account whether a quality characteristic is mentioned in a requirement. In this figure, four characteristics, resource efficiency, changeability, interoperability and security are related to three, one, one and one requirement(s) respectively. Finally, the number of requirements related to each characteristic is counted respectively, and the numbers are normalized into 0 to 1 based on the total number of requirements. The normalized result is visualized as a bar chart at the bottom right in Figure 2 and this kind of vector value is called “quality spectrum” in our previous paper [15]. As mentioned in the first section, quality spectrum tells us which quality characteristics are more important than the others.

One of the serious problems of the old method is the step to make relationships between requirements and characteristics. The old method largely depends on the expertise of an analyst performing the method. As a result, it takes a lot of hours to perform the old method even if the analyst has enough expertise such as domain knowledge of both the application and the quality characteristics. In addition, it is hard to reuse experiences performing the old method.

### 2.2 TCM Method

To overcome the problem in the old method, we introduce a term-characteristic map (TCM) as domain knowledge for making relationships between documents and characteristics. Figure 3 shows an extended example of Figure 2. TCM is a simple mapping

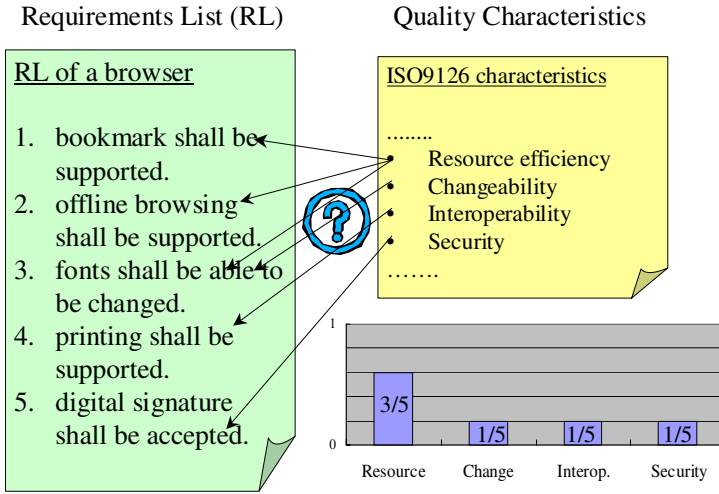


Fig. 2. An Example of the Old Method

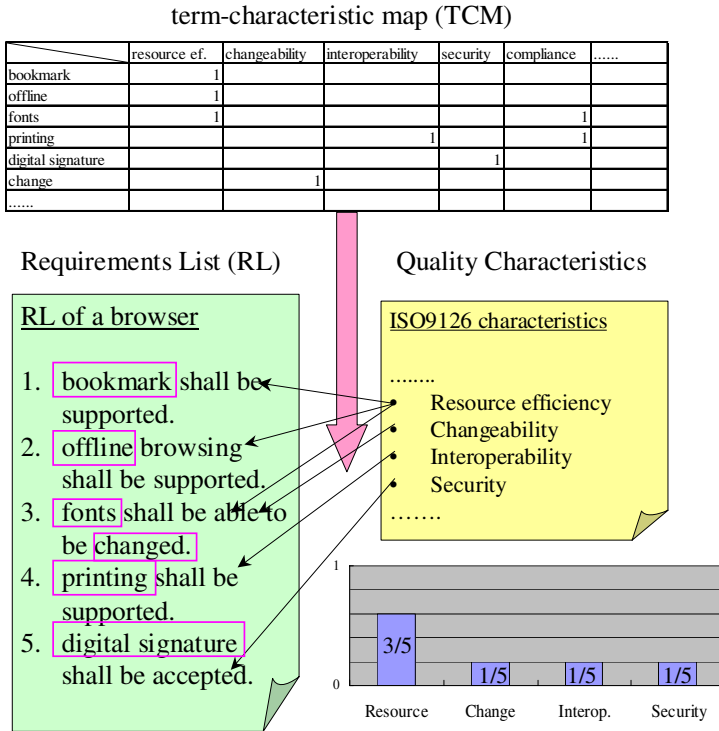


Fig. 3. An Example of TCM Method

between terms and characteristic, that tells potential relationships between them. In Figure 3 terms in “RL of a browser” can be looked up in TCM, and an analyst can easily make relationships between requirements and characteristics. TCM merely tells potential relationships. In addition, the possibility whether a term is related to a characteristic depends on the contexts of the term usage. Therefore, relationships between requirements and characteristics cannot be made automatically and the analyst should make some subjective choice. In this example, the analyst does not use some mappings between several terms and a characteristic “compliance”. Currently, we simply fill 1 or 0 value (blank if the value is 0 in Figure 3) in cells of TCM to show whether there is potential relationship or not, but we would like to introduce some ordinal or ratio values to show the degree of its potential.

Although some domain expert or an analyst himself should develop TCM beforehand, TCM will be able to be reused and be improved among similar systems in the same application domain. We would like to confirm this point in the future.

### 3 Evaluation

We evaluate TCM method mentioned in the last section with respect to the following three points.

- Results of TCM method inherit those of the old method.
- Domain specific spectrum can be found by using TCM method, i.e., spectra of several different systems in the same domain are similar with each other.
- Results of TCM method are objective, i.e., different analysts can generate similar spectra of a system.

#### 3.1 Data Gathering and Evaluation Method

To evaluate TCM method with respect to three points above, we need the following kinds of spectra: a spectrum generated by using the old method, a spectrum generated by using TCM method, a spectrum generated by another analyst using TCM method with the same TCM and a spectrum by the analyst with his own TCM. Figure 4 shows the outline how to gather such spectra data for our evaluation. This figure is written in data flow diagram, where boxes and notes correspond to data and ovals correspond to processes. We had two subjects called subject A and B, and we asked them perform spectrum analysis to documents of three browsers, Internet Explorer (IE), Fire Fox (FF) and Opera (OP), respectively. Subject A well knew this application domain as a user, and subject B was an average user. At the left side of Figure 4, subject A generated a spectrum without TCM. Note that this spectrum was generated before TCM method was proposed. Subjects A and B developed TCM of web browsers’ domain respectively by using documents of web browsers. Subjects A and B then performed TCM method respectively by using the same TCM developed by subject A as shown in the middle of Figure 4. Subject B also developed another spectrum by using his own TCM as shown in the right side of Figure 4. Subject A developed spectra of another types of systems mentioned in the next sub section. Both subjects used general spreadsheet to

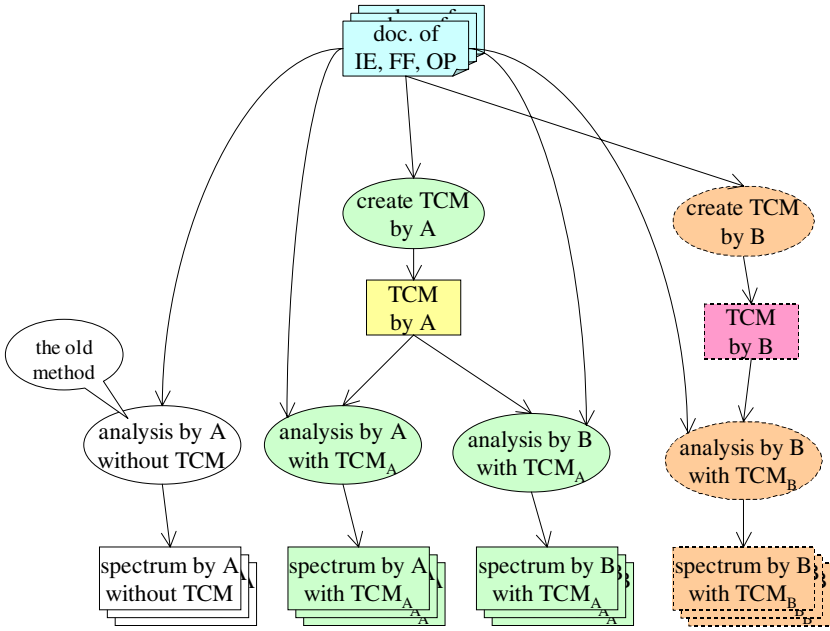


Fig. 4. Data Gathering

perform old or TCM method. Especially, subjects performing TCM method did not use a supporting tool mentioned in the next section because we decided to develop the tool based on the results of this evaluation.

Because a quality spectrum is a kind of vector, we use cosine similarity (cossim) to decide whether two spectra are similar with each other. The definition of cosine similarity between  $\mathbf{a}$  and  $\mathbf{b}$  is as follows.

$$cossim(\mathbf{a}, \mathbf{b}) = \frac{a_1 * b_1 + a_2 * b_2 + \dots + a_n * b_n}{\sqrt{a_1^2 + a_2^2 + \dots + a_n^2} * \sqrt{b_1^2 + b_2^2 + \dots + b_n^2}}$$

When two vectors are completely the same, the value is one. Because quality spectrum never has negative value in its vector, cosine similarity between two quality spectra varies from 0 to 1. Therefore, we may regard two quality spectra are similar if their cosine similarity is close to 1. For example,  $cossim(\mathbf{a}, \mathbf{b})$  is 0.99 when  $\mathbf{a}$  is ( 0.00 0.08, 0.15, 0.25, 0.00, 0.00, 0.01, 0.15, 0.20, 0.85, 0.04, 0.09, 0.01, 0.25, 0.00, 0.00, 0.01, 0.00, 0.01, 0.01) and  $\mathbf{b}$  is ( 0.00, 0.05, 0.18, 0.21, 0.00, 0.00, 0.01, 0.18, 0.12, 0.86, 0.02, 0.08, 0.01, 0.18, 0.00, 0.00, 0.03, 0.00, 0.00, 0.02). Note that  $\mathbf{a}$  corresponds to a spectrum by A with TCM A in Figure 5 and  $\mathbf{b}$  corresponds to a spectrum by B with TCM A in the same figure.

### 3.2 Inheritance from the old method

Because TCM method is one of the improved version of the old method in our previous paper [15], a quality spectrum generated by the method should be similar to one by the

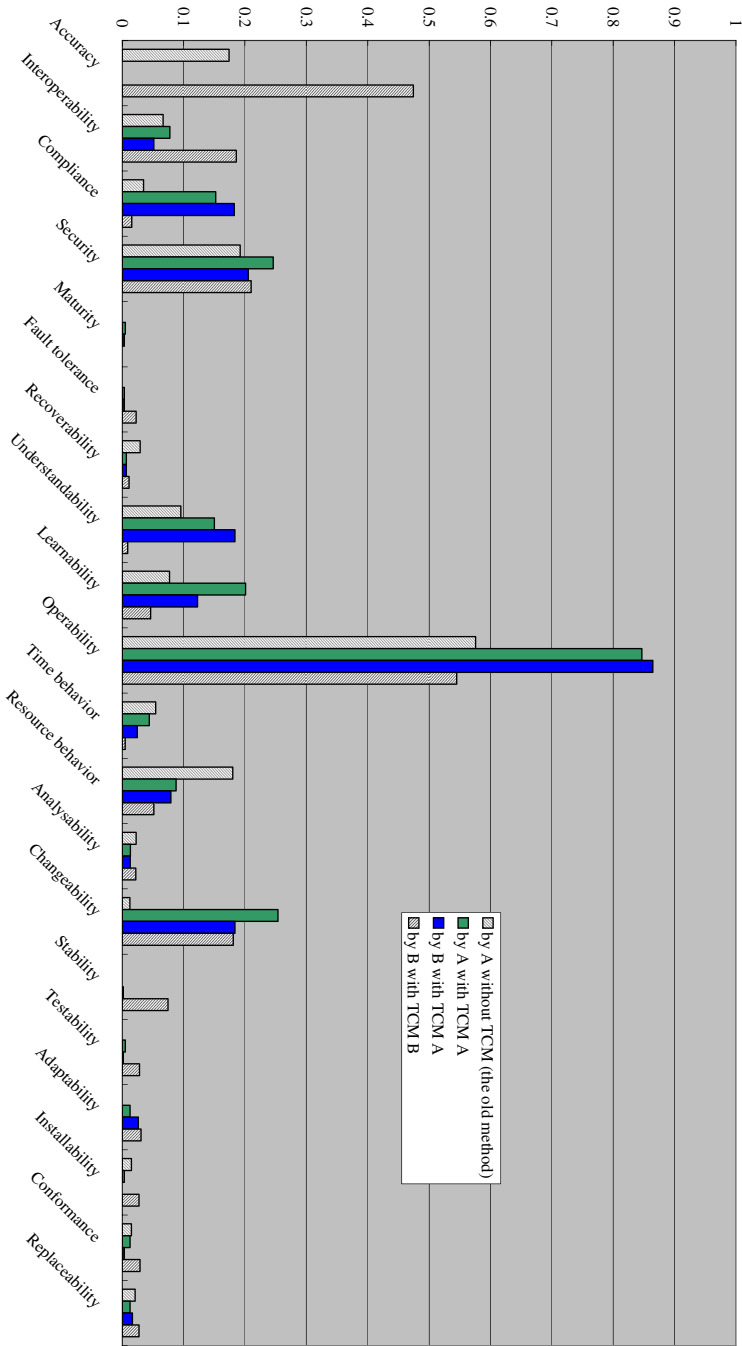


Fig. 5. Four Spectra from the old method, TCM method by subject A and B

old method. Figure 5 shows four quality spectra corresponding to the outputs in Figure 4. Note that each spectrum in Figure 5 is the average of spectra of three web browsers; IE, FF and OP. Horizontal axis of this figure shows quality characteristics used in our quality spectrum analysis. Because we have no explicit users of web browsers, we cannot identify objectives of such users. Therefore, we do not use a quality characteristic “suitability” in ISO9126 during this evaluation. Vertical axis shows the values of spectrum for each quality characteristic. Because documents of web browsers are analyzed and browsers are highly interactive system, “operability” has the highest value in a spectrum. “Security” has also higher value because of a lot of threats over the Internet. According to the definition of cosine similarity, similarity value between first and second spectra is 0.91, and the value between first and third spectra is 0.92. Therefore, we may regard TCM method inherits analytic ability from the old method.

### 3.3 Different Systems in The Same Domain

Quality spectrum is used to identify mandatory and optional quality requirements in an application domain, and this usage is based on the fact that different systems in the same domain have similar quality spectrum [15]. We confirm this fact by using TCM method.

Figure 6 shows three spectra for each browser by subject A (“spectrum by A with TCM<sub>A</sub>” in Figure 4). Figure 7 shows three spectra by subject B (“spectrum by B with TCM<sub>A</sub>” in Figure 4). As mentioned in last subsection, A and B used the same TCM developed by subject A. As shown in these figures, the spectra for each subject are

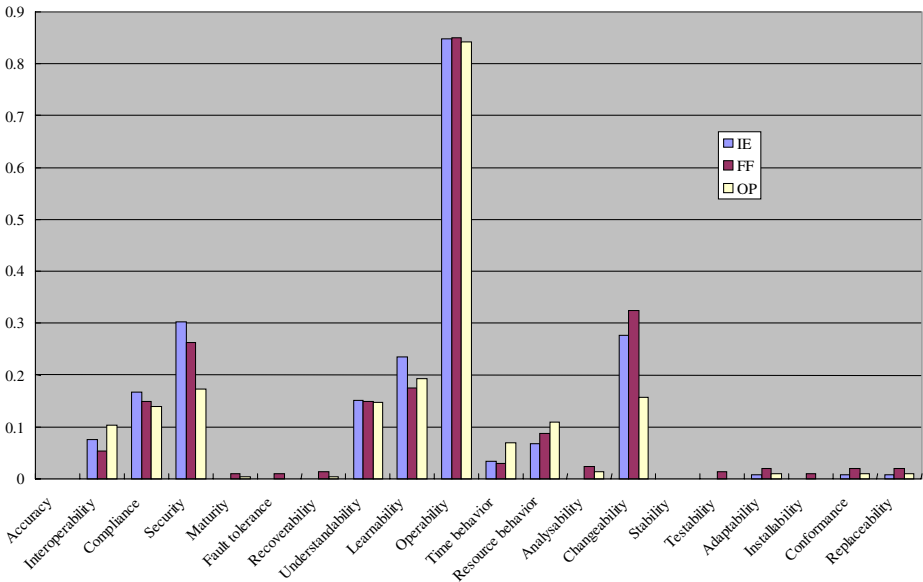


Fig. 6. Three Quality Spectra for each browser by Subject A

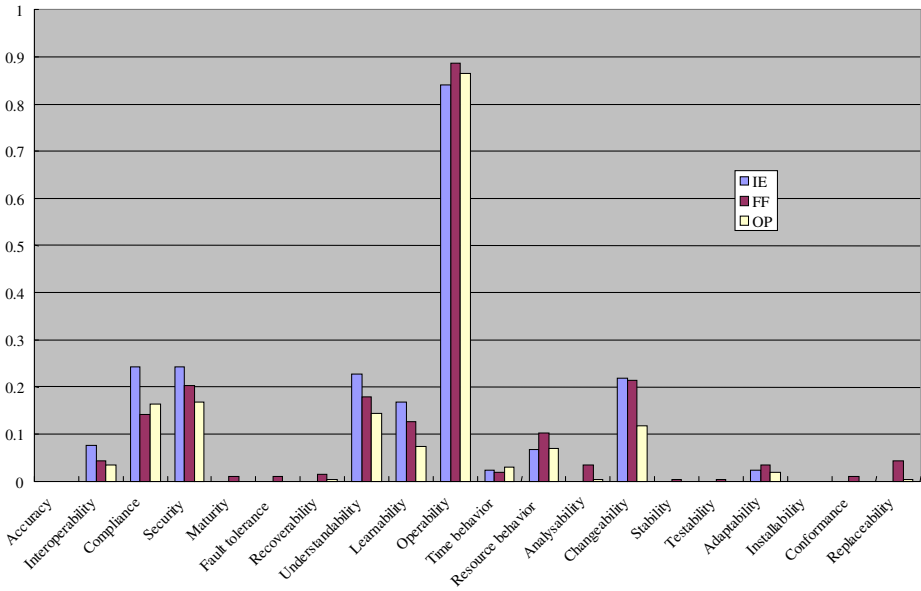


Fig. 7. Three Quality Spectra for each browser by Subject B

similar. In the case of subject A in Figure 6, cosine similarity between two out of three browsers are 0.99, 0.98 and 0.98. In the case of subject B in Figure 7, cosine similarity between two out of three browsers are 0.98, 0.97 and 0.99.

We also have quality spectra of software systems other than browsers, and we show bar charts of both browsers and other types of systems in Figure 8. A system labeled “NEWS” is a kind of a proxy system for feeding news articles to a specific intranet, and another system labeled “DB” is a kind of a document management system. Documents for both systems were published by our government [18]. Both systems are neither browsers nor interactive systems. Therefore, the spectra of NEWS and DB should be different from spectra of browsers. As shown in Figure 8, spectra of NEWS and DB are clearly different from spectra of browsers. Cosine similarity between NEWS or DB and each browser is almost 0.5. Therefore, we may regard different systems in the same domain have similar quality spectrum generated by TCM method. In addition, different types of systems have different quality spectrum.

As shown in figures 6 and 7, the power (amplitude) of some characteristics is different within three browsers. For example, powers of “interoperability” and “changeability” are different with each other. Because these three browsers are different with respect to its license (open source software or not), its platform (multi-platform including mobile devices or not) and so on, quality characteristics such as “interoperability” and “changeability” will be differently focused. On the other hand, powers of “operability” are similar because this characteristic is important for interactive systems such as browsers in general.

### 3.4 Different Analysts

Because one of the expected advantages of TCM method is that the result is more objective than the old method. To confirm this advantage, we compare a spectrum by subject A with  $TCM_A$  and another by B with  $TCM_A$  in Figure 5. Because cosine similarity between these two spectra is 0.99, we may regard results by using TCM method is almost the same. On the other hands, cosine similarity between a spectrum by B with  $TCM_A$  and another spectrum by B with  $TCM_B$  is 0.75. We may also regard these two spectra are slightly different with each other. As a result, sharing TCM seems to help analysts to analyze requirements documents objectively. As mentioned in 3.2, subjects performing TCM method only use general spreadsheet. Therefore, they have to achieve tedious tasks that can be performed automatically because no supporting tools existed.

## 4 A Supporting Tool for TCM Method

Through evaluation in the last section, we can confirm TCM method seems to work well. To improve the efficiency of the task using TCM method, we are developing a supporting tool as shown in Figure 9. In an example in this figure, NEWS system mentioned in Figure 8 is analyzed. As stated in subsection 2.2, TCM method includes both subjective and automatic tasks. Therefore, its supporting tool should be interactive one.

Before performing quality spectrum analysis, someone especially domain expert has to perform the following task.

1. To create TCM of a domain.

An analyst then analyzes requirements with the help of following automatic tasks.

2. To look up terms appearing in each requirement in TCM, and to look up characteristics related to each term.
3. To generate quality spectrum by counting the number of requirements related to each quality characteristic.

The analyst finally performs the following tasks.

4. To choose quality characteristics actually related to each requirement based on TCM, contexts of each term and his expertise.

The tool in Figure 9 supports its users to perform four tasks above in the following ways.

1. The domain analyst can manually generate TCM by using “Term Characteristic Map (TCM)” tab in this figure (the contents of the tab are not shown in this figure). He first enumerates terms usually appearing in an application domain, and fill the checkboxes corresponding to quality characteristics for each term. By using this tool, candidates of terms can be automatically extracted and enumerated in “Term Characteristic Map (TCM)” tab from text documents. Therefore, the analyst can pick terms out from such candidates.



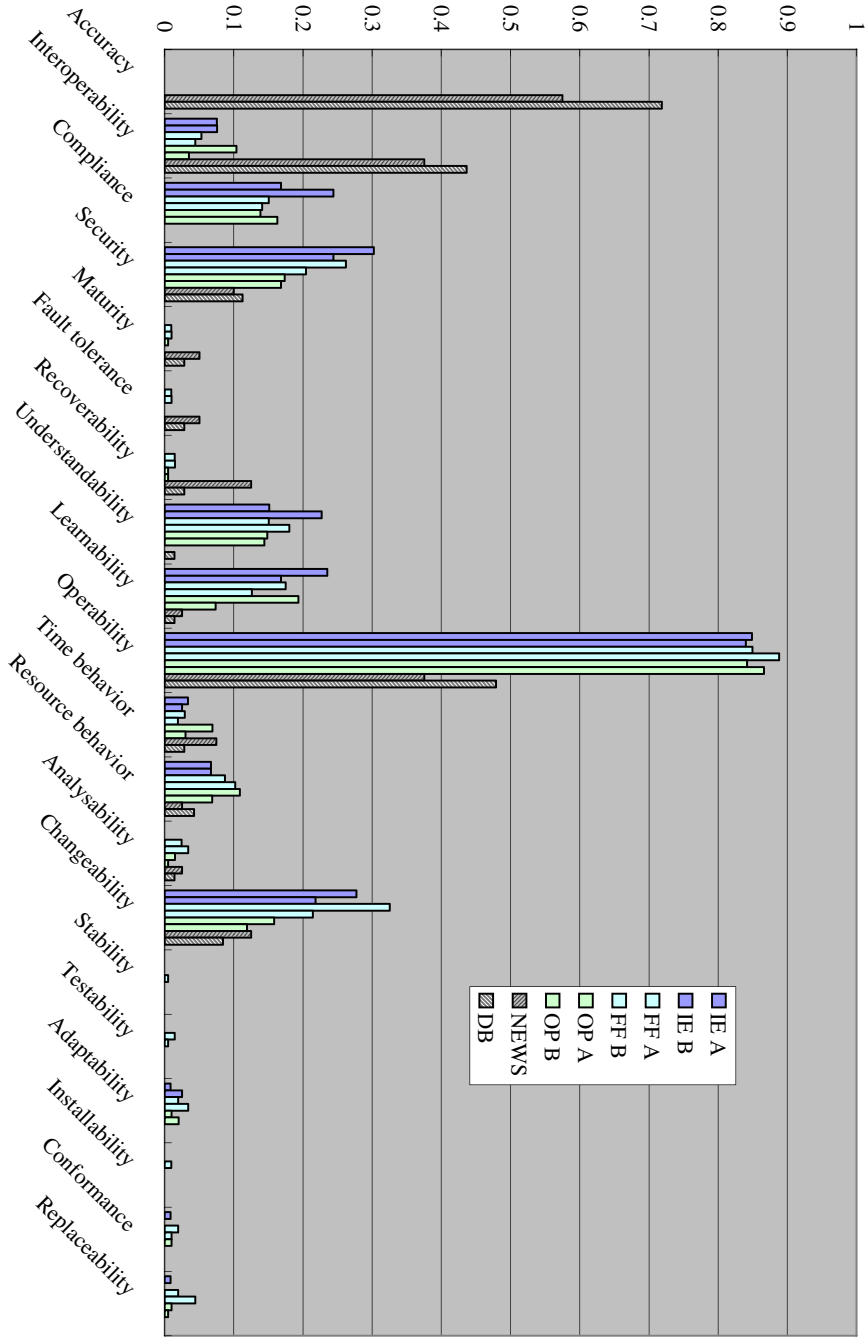


Fig. 8. Quality Spectra of browsers and other types of software

Nice tool

File Tool Window Help

Make Load Save Evaluate Spec Sheet Generate TCM

Spec Sheet & Evaluation (\*) Term Characteristic Map (TCM) (\*)

Spec Sheet Evaluations

No	Sentence	Su...	Ac...	Int...	Co...	Se...	Ma...	Fa...	Re...	Le...	Uh...	O...	T...	Re...	An...	Ch...	St...	Te...	Ad...	In...	Co...	Re...	
26	The system shall be co...	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
27	The system shall greve ...	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
28	The system shall work c...	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
29	The system shall comm...	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
30	The system shall accept...	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
31	The system shall be acc...	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Detail of Selected Spec

The system shall work correctly without any modification when the amount of data increases 50 %

Edit detail Edit Comp

Related TCM

No	Term	Su...	Ac...	Int...	Co...	Se...	Ma...	Fa...	Re...	Le...	Uh...	Op...	T...	Re...	An...	Ch...	St...	Te...	Ad...	In...	Co...	Re...	
126	data																						
139	correctly																						
142	modification																						
143	without																						
145	amount																						

Add Delete

Reading Word List is completed.

synchronized

Fig. 9. A Snapshot of A Supporting Tool

2. As shown in the center area “Detail of Selected Spec” of the figure, terms are automatically identified according to the terms in pre-loaded TCM. A subset of the TCM is then shown at the bottom table “Related TCM” in the figure. In this example, five terms “correctly”, “without”, “modification”, “amount” and “data” are looked up.
3. According to the checks on the table “Related TCM” in the figure, related quality characteristics are automatically accumulated at the top table of this figure “Spec Sheet Evaluations”. Based on the checks of characteristics for each requirement (wrote “sentence” in this tool), the tool will visualize or output quality spectrum.
4. An analyst may freely change the value of checkboxes on the table “Related TCM” in this figure. Because this table is a copy of original TCM (generated in the first task), the original TCM gets no effects according to the changes.

## 5 Related Works

There are several studies how to define each quality requirement. In ISO25021 [13], concrete examples how to measure quality requirements are shown, and these examples help analysts to make quality requirements measurable. Donald Firesmith gives some format to specify quality requirements rigorously [10]. In ATAM (Architecture Tradeoff Analysis Method) [16,2], a template for quality requirements called “quality attribute scenario” is provided to support stakeholders writing quality requirements.

Studies mentioned above focus on the *micro-view* of quality requirements because they focus on each requirement. Quality spectrum analysis [15] and its extension in this paper rather focus on the *macro-view* because it focuses on distribution of quality requirements in an artifact such as a requirements specification. We think both views are important to improve the quality requirements analysis, but there are few studies with macro-view. Studies (e.g., [4]) about the quality of requirements documents, e.g., completeness, correctness, and so on mentioned in IEEE 830 [1], also focus on the macro-view, but this kind of studies is not directly related with studies of quality requirements.

In an article by Ozkayad et al. [19], an empirical data of the most common quality attributes was shown based on the ATAM. The idea to analyze this kind of data is similar to quality spectrum analysis [15], but comparative analysis mentioned in introduction is not proposed in the article [19]. In DDP (defect detection prevention) [7,9], the relationships among requirements, risks and their mitigations are visualized. Although this visualization shows a macro-view of quality requirements, trade-offs between risks and their mitigation costs are mainly focused.

Basic idea about relationships between requirements and quality characteristics seems to be imported from QFD (Quality Function Deployment) [12]. TCM as domain knowledge is imported from a probabilistic model among terms, documents and queries in a paper by Cleland-Huang et al. [6].

## 6 Conclusion

In this paper, we improved quality requirements spectrum analysis proposed in our previous paper [15] by introducing term-characteristic map (TCM) as domain knowledge.

Quality requirements spectrum analysis is a technique for measuring and tracking quality requirements over a requirements document written in natural language. TCM helps analysts to derive amplitude of each characteristic in a quality spectrum because TCM plays a role of domain knowledge for finding quality characteristics related to each requirements statement. Through several experiments, we evaluate quality requirements analysis method with TCM, and confirmed TCM method inherited some features of the old method in our previous paper [15] and results by TCM method became more objective.

Even if TCM for a domain can be reused, it is still hard to develop and maintain TCM for each domain. A technique called LSA (Latent Semantic Analysis) [8] seems to be used for developing and maintaining TCM because terms with the similar meaning can be found automatically by using LSA. In addition, LSA seems to be used to look up quality characteristics in TCM because the semantic similarity of a requirement sentence and a quality characteristic with its description can be calculated based on term occurrences in such sentences and descriptions.

Requirements documents written in natural languages are only analyzed now, but we would like to apply quality spectrum analysis to other types of artifacts. In a paper by Zhang et al. [20], UML notation is extended for representing quality attributes. In a paper by Chowdhury et al. [5], detailed characteristics about security in source codes are identified. These studies can be used to develop methods for quality spectrum analysis for design and source codes.

As mentioned in 2.2, we do not specify the degree of relationships between terms and characteristics. In the same way as shown in a paper by Cleland-Huang [6], frequency of relationships among requirements documents in the same domain can be used for specifying such a degree. We do not also specify the degree of relationships between requirements and characteristics. Frequency of terms in each requirement can be used for specifying such a degree. In addition, types of requirements representation can be used. In an article by Glinz [11], several different types of representations, e.g., qualitative, by example, quantitative and so on, are proposed. By using such types, we can give higher degree to a requirement sentence if a quality requirement is represented not qualitatively but quantitatively, for instance.

Currently, we only focus on coarse-grained quality as characteristics in a system, but we may use another kind of characteristics scattered over the system, e.g., fine-grained quality or an attention for each type of stakeholders. In an article by Ozkayad et al. [19], fine-grained quality characteristics about security can be found, and they can be used for quality spectrum analysis. In a book by Macaulay [17], types of stakeholders are shown such as designer, financial person, maintainer and users, and each stakeholder is interested in different part of requirements. We can perform “stakeholder spectrum analysis” over a document based on such types. That is the reason why we regard spectrum analysis over software artifacts is general.

## References

1. IEEE Recommended Practice for Software Requirements Specification (October 1998) IEEE Std 830-1998 (ISBN 0-7381-0332-2 SH94654) (Print)
2. Bass, L., Clements, P., Kazman, R.: *Software Architecture in Practice*, 2nd edn. Addison-Wesley, Reading (2003)

3. David Blaine, J., Cleland-Huang, J.: Software Quality Requirements: How to Balance Competing Priorities. *IEEE Software* 25(2), 22–24 (2008)
4. Bucchiarone, A., Gnesi, S., Pierini, P.: Quality Analysis of NL Requirements: An Industrial Case Study. In: *IEEE International Requirements Engineering Conference (RE 2005)*, pp. 390–394 (2005)
5. Chowdhury, I., Chan, B., Zulkernine, M.: Security Metrics for Source Code Structures. In: *International Workshop on Software Engineering for Secure Systems (SESS 2008)*, pp. 57–64 (2008)
6. Cleland-Huang, J., Settimi, R., BenKhadra, O., Berezhanskaya, E., Christina, S.: Goal-Centric Traceability for Managing Non-Functional Requirements. In: *International Conference on Software Engineering (ICSE) (2005)*
7. Cornford, S.L., Feather, M.S., Kelly, J.C., Larson, T.W., Sigal, B., Kiper, J.D.: Design and Development Assessment. In: *Proceedings of the Tenth International Workshop on Software Specification and Design (IWSSD 2000)*, pp. 105–114 (2000)
8. Deerwester, S., Dumais, S.T., Furnas, G.W., Landauer, T.K., Harshman, R.: Indexing by Latent Semantic Analysis. *Journal of the Society for Information Science* 41(6), 391–407 (1990)
9. Feather, M.S., Cornford, S.L., Hicks, K.A., Kiper, J.D., Menzies, T.: A Broad, Quantitative Model for Making Early Requirements Decisions. *IEEE Software* 25(2), 49–56 (2008)
10. Firesmith, D.: Quality Requirements Checklist. *Journal of Object Technology* 4(9), 31–38 (2005)
11. Glinz, M.: A Risk-Based, Value-Oriented Approach to Quality Requirements. *IEEE Software* 25(2), 35–41 (2008)
12. Herzwurm, G., Schockert, S., Pietsch, W.: QFD for Customer-Focused Requirements Engineering. In: *Proceedings of 11th IEEE International Requirements Engineering Conference*, pp. 330–338 (September 2003)
13. International Standard ISO/IEC 25021. Software engineering - Software product Quality Requirements and Evaluation (SQuaRE) - Quality measure elements (October 2007)
14. International Standard ISO/IEC 9126-1. Software engineering - Product quality - Part 1: Quality model (2001)
15. Kaiya, H., Sato, T., Osada, A., Kitazawa, N., Kaijiri, K.: Toward Quality Requirements Analysis based on Domain Specific Quality Spectrum. In: *Proc. of the 23rd Annual ACM Symposium on Applied Computing 2008, Fortaleza, Ceara, Brazil, vol. 1(3)*, pp. 596–601. ACM, New York (2008) (Track on Requirements Engineering)
16. Kazman, R., Klein, M., Barbacci, M., Longstaff, T., Lipson, H., Carriere, J.: The Architecture Tradeoff Analysis Method. In: *IEEE International Conference on Engineering of Complex Computer Systems (ICECCS)*, p. 68 (1998)
17. Macaulay, L.A.: *Requirements Engineering*. In: *Applied Computing*. Springer, Heidelberg (1996)
18. Minister of Economy, Trade and Industry, Japan (in Japanese), <http://www.meti.go.jp/feedback/data/i30728aj.html>
19. Ozkayad, I., Bass, L., Sangwan, R.S., Nord, R.L.: Making Practical Use of Quality Attribute Information. *IEEE Software* 25(2), 25–33 (2008)
20. Zhang, Y., Liu, Y., Zhang, L., Ma, Z., Mei, H.: Modeling and Checking for Non-Functional Attributes in Extended UML Class Diagram. In: *Annual IEEE International Computer Software and Applications Conference (COMPSAC 2008)*, pp. 100–107 (2008)

# Author Index

- Aalst, Wil M.P. van der 2, 425, 486  
Andersson, Birger 410  
Andrikopoulos, Vasilios 290  
Antonellis, Valeria De 334
- Barone, Daniele 171  
Belhajjame, Khalid 79  
Benbernou, Salima 290  
Bergholtz, Maria 410  
Bianchini, Devis 334  
Blanc, Xavier 32  
Borgida, Alex 171
- Cabot, Jordi 125  
Cao, Jinli 305  
Cappiello, Cinzia 334
- Dalpiaz, Fabiano 246  
de Spindler, Alexandre 275  
Dobson, John 515  
Dubois, Eric 319
- Easterbrook, Steve 141  
Eder, Johann 349  
España, Sergio 530
- Fernandes, Alvaro A.A. 79  
Franch, Xavier 201
- Gailly, Frederik 395  
Ghazarian, Arbi 156  
Giachetti, Giovanni 110  
Giorgini, Paolo 246  
Godart, Claude 364  
Gómez, Cristina 125  
González, Arturo 530  
Gottschalk, Florian 486  
Green, Peter 501  
Grossniklaus, Michael 275  
Guermouche, Nawal 364  
Guerriero, Annie 319  
Guizzardardi, Giancarlo 94
- Halin, Gilles 319  
Haller, Klaus 63
- Hao, Yanan 305  
Hornung, Thomas 440  
Huang, Tao 455
- Indulska, Marta 501
- Jansen-Vullers, Monique H. 486  
Jiang, Lei 171  
Johannesson, Paul 410  
Jurjens, Jan 231
- Kajiri, Kenji 546  
Kaiya, Haruhiko 546  
Kubicki, Sylvain 319
- La Rosa, Marcello 486  
Lausen, Georg 440  
Liang, Senlin 455  
Liu, Jie 455  
Liu, Lin 216  
Lock, Russell 515
- Ma, Wenting 216  
Mao, Lu 79  
Marín, Beatriz 110  
May, Wolfgang 440  
Mens, Tom 32  
Metzger, Andreas 11  
Mougenot, Alix 32  
Mounier, Isabelle 32  
Mouratidis, Haralambos 231  
Mutschler, Bela 379  
Mylopoulos, John 141, 171, 186, 246
- Noda, Kazuhide 17  
Norrie, Moira C. 275
- Otsubo, Genya 17
- Paalvast, Edwin 8  
Papazoglou, Mike P. 290  
Pastor, Óscar 110, 530  
Paton, Norman W. 79  
Pernici, Barbara 334  
Pichler, Horst 349  
Planas, Elena 125  
Poels, Geert 395  
Pohl, Klaus 11

- Queralt, Anna 47
- Recker, Jan 501
- Reichert, Manfred 379
- Reiff-Marganiec, Stephan 261
- Reijers, Hajo A. 470
- Rosemann, Michael 501
- Salay, Rick 141, 186
- Sato, Tomonori 546
- Shadbolt, Nigel 1
- Sidorova, Natalia 425
- Simone, Mark de 6
- Sommerville, Ian 515
- Storer, Tim 515
- Sunyaev, Ali 231
- Suzuki, Shunichi 546
- Tanigawa, Masaaki 546
- Teniente, Ernest 47
- Trčka, Nikola 425
- Ubayashi, Naoyasu 17
- Wagemakers, Teun A.C. 486
- Weber, Barbara 470
- Wei, Jun 455
- Weigand, Hans 410
- Wenger, Michaela 349
- Wiederhold, Gio 9
- Wild, Werner 470
- Xie, Haihua 216
- Ye, Dan 455
- Yin, Jinglei 216
- Yoshida, Jun 17
- Yu, Hong Qing 261
- Zhang, Hongyu 216
- Zhang, Yanchun 305
- Zugal, Stefan 470