

CONFERENCES IN RESEARCH AND PRACTICE IN  
INFORMATION TECHNOLOGY

VOLUME 96

# CONCEPTUAL MODELLING 2009

AUSTRALIAN COMPUTER SCIENCE COMMUNICATIONS, VOLUME 31, NUMBER 6



AUSTRALIAN  
COMPUTER  
SOCIETY



CO<sub>mputing</sub>  
R<sub>esearch</sub>  
& E<sub>ducation</sub>



# CONCEPTUAL MODELLING 2009

Proceedings of the  
Sixth Asia-Pacific Conference on Conceptual Modelling  
(APCCM 2009), Wellington, New Zealand,  
January 2009

Markus Kirchberg and Sebastian Link, Eds.

Volume 96 in the Conferences in Research and Practice in Information Technology Series.  
Published by the Australian Computer Society Inc.



Published in association with the ACM Digital Library.

**Conceptual Modelling 2009.** Proceedings of the Sixth Asia-Pacific Conference on Conceptual Modelling (APCCM 2009), Wellington, New Zealand, January 2009

**Conferences in Research and Practice in Information Technology, Volume 96.**

Copyright ©2009, Australian Computer Society. Reproduction for academic, not-for-profit purposes permitted provided the copyright text at the foot of the first page of each paper is included.

Editors:

**Markus Kirchberg**

Data Mining Department

Institute for Infocomm Research (I<sup>2</sup>R)

Agency for Science, Technology and Research (A\*STAR)

1 Fusionopolis Way

#21-01 Connexis (South Tower)

Singapore 138632

Singapore

Email: [MKirchberg@i2r.a-star.edu.sg](mailto:MKirchberg@i2r.a-star.edu.sg)

**Sebastian Link**

School of Information Management

Victoria University of Wellington

PO Box 600

Wellington

New Zealand

Email: [sebastian.link@vuw.ac.nz](mailto:sebastian.link@vuw.ac.nz)

Series Editors:

Vladimir Estivill-Castro, Griffith University, Queensland

John F. Roddick, Flinders University, South Australia

Simeon Simoff, University of Western Sydney, NSW

[crpit@infoeng.flinders.edu.au](mailto:crpit@infoeng.flinders.edu.au)

Publisher: Australian Computer Society Inc.

PO Box Q534, QVB Post Office

Sydney 1230

New South Wales

Australia.

Conferences in Research and Practice in Information Technology, Volume 96.

ISSN 1445-1336.

ISBN 978-1-920682-77-4.

Printed, January 2009 by Flinders Press, PO Box 2100, Bedford Park, SA 5042, South Australia.

Cover Design by Modern Planet Design, (08) 8340 1361.

The *Conferences in Research and Practice in Information Technology* series aims to disseminate the results of peer-reviewed research in all areas of Information Technology. Further details can be found at <http://crpit.com/>.

## Table of Contents

### Proceedings of the Sixth Asia-Pacific Conference on Conceptual Modelling (APCCM 2009), Wellington, New Zealand, January 2009

Preface .....	vii
Programme Committee .....	viii
Organising Committee .....	ix
Welcome from the Organising Committee .....	x
CORE - Computing Research & Education .....	xi
ACSW Conferences and the Australian Computer Science Communications .....	xii
ACSW and APCCM 2009 Sponsors .....	xiv

### Keynote

Finite Model Theory and its Origins .....	3
<i>Ronald Fagin</i>	

### Invited Papers

A Semantic Associative Computation Method for Automatic Decorative-Multimedia Creation with “Kansei” Information .....	7
<i>Yasushi Kiyoki and Xing Chen</i>	
Modeling Natural Language Communication in Database Semantics .....	17
<i>Roland Hausser</i>	

### Contributed Papers

Business Process Integration: Method and Analysis .....	29
<i>Evan D. Morrison, Alex Menzies, George Koliadis and Aditya K. Ghose</i>	
Conceptional Modeling and Analysis of Spatio-Temporal Processes in Biomolecular Systems .....	39
<i>Andreas Schäfer and Mathias John</i>	
Conceptual Application Domain Modelling .....	49
<i>Bernhard Thalheim, Klaus-Dieter Schewe and Hui Ma</i>	
Conceptual Business Document Modeling using UN/CEFACT’s Core Components .....	59
<i>Philipp Liegl</i>	
Contrasting Classification with Generalisation .....	71
<i>Thomas Kühne</i>	

Extracting and Modeling the Semantic Information Content of Web Documents to Support Semantic Document Retrieval .....	79
<i>Shahrul Azman Noah, Lailatulqadri Zakaria and Arifah Che Alhadi</i>	
Extracting Conceptual Graphs from Japanese Documents for Software Requirements Modeling .....	87
<i>Ryo Hasegawa, Motohiro Kitamura, Haruhiko Kaiya and Motoshi Saeki</i>	
Modelling Web-Oriented Architectures .....	97
<i>Gunnar Thies and Gottfried Vossen</i>	
Multi-Level Domain Modeling with M-Objects and M-Relationships .....	107
<i>Bernd Neumayr, Katharina Grün and Michael Schrefl</i>	
Reverse Engineering of XML Schemas to Conceptual Diagrams .....	117
<i>Martin Nečaský</i>	
Synthesis of Orchestrators from Service Choreographies .....	129
<i>Stephen McIlvenna, Marlon Dumas and Moe Thandar Wynn</i>	
Towards Accurate Conflict Detection in a VCS for Model Artifacts: A Comparison of Two Semantically Enhanced Approaches .....	139
<i>Kerstin Altmanninger and Gabriele Kotsis</i>	
<b>Author Index</b> .....	147

## Preface

This volume contains the proceedings of the *Sixth Asia-Pacific Conference on Conceptual Modelling (APCCM 2009)*, held at the Museum of New Zealand Te Papa Tongarewa in Wellington, New Zealand from January 20 to 23, 2009 as part of the Australasian Computer Science Week (ACSW 2009).

The APCCM series focuses on disseminating the results of innovative research in conceptual modelling and related areas, and provides an annual forum for experts from all areas of computer science and information systems with a common interest in the subject. The scope of APCCM 2009 includes areas such as:

- Business, enterprise, process and services modelling;
- Concepts, concept theories and ontologies;
- Conceptual modelling and user participation;
- Conceptual modelling for decision support and expert systems; digital libraries; e-business, e-commerce and e-banking systems; health care systems; knowledge management systems; mobile information systems; user interfaces; and Web-based systems;
- Conceptual modelling of semi-structured data and XML;
- Conceptual modelling of spatial, temporal and biological data;
- Conceptual modelling quality;
- Conceptual models in management science;
- Design patterns and object-oriented design;
- Evolution and change in conceptual models;
- Implementations of information systems;
- Information and schema integration;
- Information customisation and user profiles;
- Information recognition and information modelling;
- Information retrieval, analysis, visualisation and prediction;
- Information systems design methodologies;
- Knowledge discovery, knowledge representation and knowledge management;
- Methods for developing, validating and communicating conceptual models;
- Philosophical, mathematical and linguistic foundations of conceptual models;
- Reuse, reverse engineering and reengineering;
- Semantic Web; and
- Software engineering and tools for information systems development.

The program committee has selected the contributed papers from 38 submissions. All submitted papers have been refereed by at least two international experts, and have been discussed thoroughly. The twelve papers judged best by the program committee members have been accepted and are included in this volume.

The program committee invited Dr. Ronald Fagin, to present a plenary keynote on *Finite Model Theory and its Origins*. Dr. Fagin is the Manager of the Foundations of Computer Science group at the Computer Science department of the IBM Almaden Research Centre in San Jose, California, USA. The committee also invited Professor Roland Hausser from the University of Erlangen, Germany, and Professor Yasushi Kiyoki from Keio University, Japan, to give invited talks. Professor Hausser's talk was entitled *Modeling Natural Language Communication in Database Semantics*, and Professor Kiyoki's talk was entitled *A Semantic Associative Computation Method for Automatic Decorative Multimedia Creation with "Kansei" Information*.

The program committee selected the paper *Reverse Engineering of XML Schemas to Conceptual Diagrams* by Martin Nečaský for the APCCM 2009 Best Paper Award. The award includes a prize of NZ\$ 500.-; which has been sponsored by the School of Information Management, Victoria University of Wellington, New Zealand. Warmest congratulations to the author.

We wish to thank all authors who submitted papers and all the conference participants for the fruitful discussions. We are grateful to the members of the program committee and the additional reviewers for their timely expertise in carefully reviewing the papers. We also like to acknowledge the excellent work of Dagong Dong who programmed and supported the MuCoMS conference management system (<http://www.mucoms.org/>). Finally, we wish to express our appreciation to the local organisers at the Victoria University of Wellington for the wonderful days in Wellington.

**Markus Kirchberg**

Institute for Infocomm Research (I<sup>2</sup>R), A\*STAR, Singapore

**Sebastian Link**

Victoria University of Wellington, New Zealand

APCCM 2009 Programme Chairs

January 2009

# Programme Committee

## Chairs

Markus Kirchberg, Institute for Infocomm Research (I<sup>2</sup>R), A\*STAR (Singapore)  
Sebastian Link, Victoria University of Wellington (New Zealand)

## Members

Boualem Benatallah, University of New South Wales (Australia)  
Byron Koon Kau Choi, Hong Kong Baptist University (Hong Kong SAR, China)  
Denise de Vries, Flinders University (Australia)  
Gillian Dobbie, University of Auckland (New Zealand)  
Aditya K. Ghose, University of Wollongong (Australia)  
Angela Eck Soong Goh, Nanyang Technological University (Singapore)  
Sven Hartmann, Clausthal University of Technology (Germany)  
Annika Hinze, University of Waikato (New Zealand)  
Yasushi Kiyoki, Keio University (Japan)  
Dirk Labudde, Dresden University of Technology (Germany)  
Chiang Lee, National Cheng-Kung University (Taiwan)  
Qing Li, City University of Hong Kong, (Hong Kong SAR, China)  
Jixue Liu, University of South Australia (Australia)  
Pavle Mogin, Victoria University of Wellington (New Zealand)  
James Noble, Victoria University of Wellington (New Zealand)  
Sudha Ram, University of Arizona (USA)  
Michael Rosemann, Queensland University of Technology (Australia)  
Motoshi Saeki, Tokyo Institute of Technology (Japan)  
Klaus-Dieter Schewe, Information Science Research Centre (New Zealand)  
Il-Yeol Song, Drexel University (USA)  
Stefano Spaccapietra, EPFL (Switzerland)  
Nigel Stanger, University of Otago (New Zealand)  
Markus Stumptner, University of South Australia (Australia)  
Bernhard Thalheim, Christian-Albrechts-University Kiel (Germany)  
Yanchun Zhang, Victoria University (Australia)

## Additional Reviewers

George Koliadis, University of Wollongong (Australia)  
Thomas Kühne, Victoria University of Wellington (New Zealand)  
Duy Ngan Le, Nanyang Technological University (Singapore)  
Ki Jung Lee, Drexel University (USA)  
Evan Morrison, University of Wollongong (Australia)  
Wee Siong Ng, Institute for Infocomm Research (I<sup>2</sup>R), A\*STAR, (Singapore)  
Rajesh Thiagarajan, University of South Australia (Australia)  
Ornsiri Thonggoom, Drexel University (USA)



# Organising Committee

## Co-Chairs

Dr Alex Potanin  
Professor John Hine

## Venues

Dr David Pearce

## Operations

Dr Peter Komisarczuk  
Mrs Suzan Hall  
Mr Craig Anslow

## Finance and Program

Dr Stuart Marshall

## Communications

Dr Ian Welch  
Mr Craig Anslow

## Events

Professor John Hine

# Welcome from the Organising Committee

We would like to welcome you to ACSW2009 hosted by Victoria University of Wellington, New Zealand.

Wellington is set on the edge of a stunning harbour and surrounded by rolling hills. The earliest name for Wellington, from Maori legend, is Te Upoko o te Ika a Maui. In Maori it means the head of Maui's fish. Caught and pulled to the surface by the Polynesian navigator Maui, the fish became the North Island. Wellington is the capital city of New Zealand and home to the seat of parliament. But this vibrant and dynamic city also has many other capital claims including Culture capital, Creative capital and Events capital. It is a compact, walkable city waiting to be explored. The conference venue is less than fifteen minutes walk to accommodation, Courtenay Place with its wide range of bars, and the harbour with its restaurants and activities such as sea kayaking. The conference venue itself is in the Museum of New Zealand Te Papa Tongarewa, offering visitors a unique and authentic experience of this country's treasures and stories. Over five floors, you can explore the nation's nature, art, history, and heritage - from the shaping of its land to the spirit of its diverse peoples, from its unique wildlife to its distinctive art and visual culture.

Victoria University of Wellington - Te Whare Wānanga o te Ūpoko o te Ika a Māui - is over a century old. Victoria College was founded through an Act of Parliament in 1897, the year of Queen Victoria's Diamond Jubilee celebrations, and named in her honour. Victoria is a thriving community of almost 25,000 people. Situated in the capital city across four campuses, Victoria can take advantage of connections and values its relationships with iwi, business, government, the judiciary, public and private research organisations, cultural organisations and resources, other universities and tertiary providers and the international community through the diplomatic corps. ACSW2009 coincides with the opening of the new School of Engineering and Computer Science as part of the Faculty of Engineering at Victoria University of Wellington - combining a long history of research and teaching of the software engineering and network engineering in the Computer Science department and computer system engineering and electronic engineering in the Physics department. Professor John Hine, co-chairing ACSW2009, is the current Dean of Engineering and the inaugural Head of School of Engineering and Computer Science.

ACSW2009 consists of the following conferences:

- Australasian Computer Science Conference (ACSC) (Chaired by Bernard Mans),
- Australasian Computing Education Conference (ACE) (Chaired by Margaret Hamilton and Tony Clear),
- Australasian Database Conference (ADC) (Chaired by Athman Bouguettaya and Xuemin Lin),
- Australasian Symposium on Grid Computing and e-Research (AUSGRID) (Chaired by Wayne Kelly and Paul Roe),
- Computing: The Australasian Theory Symposium (CATS) (Chaired by Prabhu Manyem and Rod Downey),
- Asia-Pacific Conference on Conceptual Modelling (APCCM) (Chaired by Markus Kirchberg and Sebastian Link),
- Australasian Information Security Conference (AISC) (Chaired by Ljiljana Brankovic and Willy Susilo),
- Australasian Workshop on Health Informatics and Knowledge Management (HIKM) (Chaired by Jim Warren),
- Australasian User Interface Conference (AUIC) (Chaired by Gerald Weber and Paul Calder),
- Australasian Computing Doctoral Consortium (ACDC) (Chaired by David Pearce and Vladimir Estivill-Castro).

The nature of ACSW requires the co-operation of numerous people. We would like to thank all those who have worked to ensure the success of ACSW2009 including the Organising Committee, the Conference Chairs and Programme Committees, our sponsors, the keynote speakers and the delegates.

**Dr Alex Potanin and Professor John Hine**

ACSW2009 Co-Chairs

Victoria University of Wellington

January, 2009

## CORE - Computing Research & Education

CORE welcomes all delegates to ACSW2009 in Wellington. CORE, the peak body representing academic computer science in Australia and New Zealand, is responsible for the annual ACSW series of meetings, which are a unique opportunity for our community to network and to discuss research and topics of mutual interest. The original component conferences – ACSC, ADC, and CATS, which formed the basis of ACSW in the mid 1990s – now share the week with seven other events, which build on the diversity of the Australasian CS community.

This year, we have chosen to feature a small number of plenary speakers chosen from across the discipline, Ronald Fagin, Ian Foster, Mark Guzdial, and Andy Hopper. I thank them for their contributions to ACSW'09. The efforts of the conference chairs and their program committees have led to strong programs in all the conferences – again, thanks. And thanks are particularly due to Alex Potanin, John Hine, and their colleagues for organising what promises to be a memorable ACSW.

In Australia, 2008 has been a busy year for academia, with the incoming Labor government instituting major reviews in areas such as the higher education sector, research funding, postgraduate study, and national curricula. However, while the reviews have exposed severe shortcomings in the funding of higher education and research, they have not as yet been translated into definite action, and the sector as a whole is shrinking. Although there is a widespread perception of a shortage of IT staff, and graduate salaries remain strong, student interest in ICT continues to be low. Moreover, per-place funding for computer science students has dropped relative to that of other physical and mathematical sciences. Several forums and initiatives involving industry, government, and academia have attempted to address the issue of the ongoing difficulties of attracting students to the discipline, but with little perceptible effect. New initiatives that seek to address the issues of students and funding will be a CORE priority in 2009.

During 2008, CORE continued to work on journal and conference rankings, with much of the activity driven by requests for information from the government. A key aim is now to maintain the rankings, which are widely used overseas as well as in Australia, a challenging process that needs to balance competing special interests as well as addressing the interests of the community as a whole. A new activity in 2008 was a review of computing curriculum, which is still ongoing, with the intention that a CORE curriculum statement be used for accreditation of degrees in computer science, software engineering, and information technology. ACSW'09 includes a forum on computing curriculum to discuss this process.

CORE's existence is due to the support of the member departments in Australia and New Zealand, and I thank them for their ongoing contributions, in commitment and in financial support. Finally, I am grateful to all those who gave their time to CORE in 2008; in particular, I thank Jenny Edwards, Alan Fekete, Tom Gedeon, Leon Sterling, Vanessa Teague, and the members of the executive and of the curriculum and ranking committees.

**Justin Zobel**  
President, CORE  
January, 2009

# ACSW Conferences and the Australian Computer Science Communications

The Australasian Computer Science Week of conferences has been running in some form continuously since 1978. This makes it one of the longest running conferences in computer science. The proceedings of the week have been published as the *Australian Computer Science Communications* since 1979 (with the 1978 proceedings often referred to as *Volume 0*). Thus the sequence number of the Australasian Computer Science Conference is always one greater than the volume of the Communications. Below is a list of the conferences, their locations and hosts.

**2010.** Volume 32. Host and Venue - Queensland University of Technology, Brisbane, QLD.

**2009. Volume 31. Host and Venue - Victoria University, Wellington, New Zealand.**

**2008.** Volume 30. Host and Venue - University of Wollongong, NSW.

**2007.** Volume 29. Host and Venue - University of Ballarat, VIC. First running of HDKM.

**2006.** Volume 28. Host and Venue - University of Tasmania, TAS.

**2005.** Volume 27. Host - University of Newcastle, NSW. APBC held separately from 2005.

**2004.** Volume 26. Host and Venue - University of Otago, Dunedin, New Zealand. First running of APCCM.

**2003.** Volume 25. Hosts - Flinders University, University of Adelaide and University of South Australia. Venue - Adelaide Convention Centre, Adelaide, SA. First running of APBC. Incorporation of ACE. ACSAC held separately from 2003.

**2002.** Volume 24. Host and Venue - Monash University, Melbourne, VIC.

**2001.** Volume 23. Hosts - Bond University and Griffith University (Gold Coast). Venue - Gold Coast, QLD.

**2000.** Volume 22. Hosts - Australian National University and University of Canberra. Venue - ANU, Canberra, ACT. First running of AUC.

**1999.** Volume 21. Host and Venue - University of Auckland, New Zealand.

**1998.** Volume 20. Hosts - University of Western Australia, Murdoch University, Edith Cowan University and Curtin University. Venue - Perth, WA.

**1997.** Volume 19. Hosts - Macquarie University and University of Technology, Sydney. Venue - Sydney, NSW. ADC held with DASFAA (rather than ACSW) in 1997.

**1996.** Volume 18. Host - University of Melbourne and RMIT University. Venue - Melbourne, Australia. CATS joins ACSW.

**1995.** Volume 17. Hosts - Flinders University, University of Adelaide and University of South Australia. Venue - Glenelg, SA.

**1994.** Volume 16. Host and Venue - University of Canterbury, Christchurch, New Zealand. CATS run for the first time separately in Sydney.

**1993.** Volume 15. Hosts - Griffith University and Queensland University of Technology. Venue - Nathan, QLD.

**1992.** Volume 14. Host and Venue - University of Tasmania, TAS. (ADC held separately at La Trobe University).

**1991.** Volume 13. Host and Venue - University of New South Wales, NSW.

**1990.** Volume 12. Host and Venue - Monash University, Melbourne, VIC. Joined by Database and Information Systems Conference which in 1992 became ADC (which stayed with ACSW) and ACIS (which now operates independently).

**1989.** Volume 11. Host and Venue - University of Wollongong, NSW.

**1988.** Volume 10. Host and Venue - University of Queensland, QLD.

**1987.** Volume 9. Host and Venue - Deakin University, VIC.

**1986.** Volume 8. Host and Venue - Australian National University, Canberra, ACT.

**1985.** Volume 7. Hosts - University of Melbourne and Monash University. Venue - Melbourne, VIC.

**1984.** Volume 6. Host and Venue - University of Adelaide, SA.

**1983.** Volume 5. Host and Venue - University of Sydney, NSW.

**1982.** Volume 4. Host and Venue - University of Western Australia, WA.

**1981.** Volume 3. Host and Venue - University of Queensland, QLD.

**1980.** Volume 2. Host and Venue - Australian National University, Canberra, ACT.

**1979.** Volume 1. Host and Venue - University of Tasmania, TAS.

**1978.** Volume 0. Host and Venue - University of New South Wales, NSW.

## Conference Acronyms

**ACE.** Australian/Australasian Computing Education Conference.  
**ACSAC.** Asia-Pacific Computer Systems Architecture Conference (previously Australian Computer Architecture Conference (ACAC)).  
**ACSC.** Australian/Australasian Computer Science Conference.  
**ACSW.** Australian/Australasian Computer Science Week.  
**ADC.** Australian/Australasian Database Conference.  
**AISW.** Australasian Information Security Workshop.  
**APBC.** Asia-Pacific Bioinformatics Conference.  
**APCCM.** Asia-Pacific Conference on Conceptual Modelling.  
**AUIC.** Australian/Australasian User Interface Conference.  
**AusGrid.** Australasian Workshop on Grid Computing and e-Research.  
**CATS.** Computing - The Australian/Australasian Theory Symposium.  
**HDKM.** Australasian Workshop on Health Data and Knowledge Management.  
**HIKM.** Australasian Workshop on Health Informatics and Knowledge Management (former HDKM).

Note that various name changes have occurred, most notably the change of the names of conferences to reflect a wider geographical area.

## ACSW and APCCM 2009 Sponsors

We wish to thank the following sponsors for their contribution towards this conference. For an up-to-date overview of sponsors of ACSW 2009 and APCCM 2009, please see <http://www.mcs.vuw.ac.nz/Events/ACSW2009/Sponsors>.



CityLink, New Zealand,  
[www.citylink.co.nz](http://www.citylink.co.nz)



New Zealand Computer Society,  
[www.nzcs.org.nz](http://www.nzcs.org.nz)



Victoria University of Wellington,  
[www.victoria.ac.nz](http://www.victoria.ac.nz)



AUSTRALIAN  
COMPUTER  
SOCIETY

Australian Computer Society,  
[www.acs.org.au](http://www.acs.org.au)



CORE - Computing Research and Education,  
[www.core.edu.au](http://www.core.edu.au)



Xero,  
[www.xero.com](http://www.xero.com)



Security Assessment, New Zealand,  
[www.security-assessment.com](http://www.security-assessment.com)



Catalyst, New Zealand,  
[www.catalyst.net.nz](http://www.catalyst.net.nz)



Helium, New Zealand,  
[www.heliumnz.co.nz](http://www.heliumnz.co.nz)



Institute for  
Infocomm Research

Institute for Infocomm Research (I<sup>2</sup>R),  
Agency for Science, Technology and Research  
(A\*STAR), Singapore  
[www.i2r.a-star.edu.sg](http://www.i2r.a-star.edu.sg)

# KEYNOTE





# Finite Model Theory and its Origins

Ronald Fagin

IBM Almaden Research Center, Dept. K53/B2  
650 Harry Road, San Jose, California 95120-6099, USA  
Email: [fagin@almaden.ibm.com](mailto:fagin@almaden.ibm.com)

## Extended Abstract

Finite model theory is a study of the logical properties of finite mathematical structures. This talk gives an overview of how finite model theory arose, and of some work that sprang from that. This includes:

1. Differences between the model theory of finite structures and infinite structures. Most of the classical theorems of logic fail for finite structures, which gives us a challenge to develop new concepts and tools, appropriate for finite structures.
2. The relationship between finite model theory and complexity theory. Surprisingly enough, it turns out that in some cases, we can characterize complexity classes (such as NP) in terms of logic, without using any notion of machine, computation, or time.
3. Zero-one laws. There is a remarkable phenomenon, which says that certain properties (such as those expressible in first-order logic) are either almost surely true or almost surely false.
4. Descriptive complexity. Here we consider how complex a formula must be to express a given property.

The goal of this talk is to introduce the audience to the fascinating area of finite model theory.



# INVITED PAPERS



# A Semantic Associative Computation Method for Automatic Decorative-Multimedia Creation with “Kansei” Information

Yasushi Kiyoki\* and Xing Chen\*\*

\*Faculty of Environmental Information

Keio University

Fujisawa, Kanagawa 252-8520, Japan

\*\*Department of Information & Computer Sciences,

Kanagawa Institute of Technology,

Atsugi, Kanagawa 243-0292, Japan

\*kiyoki@sfc.keio.ac.jp, www.mdbl.sfc.keio.ac.jp

\*\*chen@ic.kanagawa-it.ac.jp

## Abstract

In the design of multimedia systems, one of the important issues is how to deal with “Kansei” of human beings. The concept of “Kansei” in Japanese includes several meanings on sensitive recognition, such as “impression,” “human senses,” “feelings,” “sensitivity,” “psychological reaction” and “physiological reaction.”

This paper presents a new concept of “automatic decorative-multimedia creation” and a semantic associative computation method. This method realizes automatic main-media decoration with dynamic sub-media data selection for representing main-media as decorative multimedia. The aim of this method is to create a new field of “automatic decorative-media art” with “semantic associative computing.”

This paper defines an “automatic media decoration model” with semantic spaces and media-decoration functions. Automatic media decoration is realized by applying the Mathematical Model of Meaning (MMM) to a media-transmission space for computing semantic correlations between main-media objects and sub-media.

The process of this dynamic media decoration method consists of the following functions:

- (1) Extraction of semantic “Kansei” features of “main-media object,” such as music, image and video.
- (2) Mapping of the main-media object onto the media-transmission space between main-media and sub-media.
- (3) Semantic associative computation of correlations between the main-media object and the features of the sub-media space by MMM, and creating a vector of the main-media object with the features of the sub-media space.
- (4) Mapping of the vector of the main-media object to the sub-media space, and semantic associative computing between the main-media object and sub-media data.

- (5) Semantic ranking of sub-media objects as the result of the semantic associative computation, and selects one of the sub-media objects with high correlation values to the target main-media object.

- (6) Automatic rendering of the target main-media object with the selected sub-media object for decorating the main-media presentation.

This paper shows several significant applications of the semantic associative computation method for “automatic decorative-media creation.”

## 1 Introduction

In the field of multimedia systems, the concept of “Kansei” is related to data definition and data retrieval with “Kansei” information for multimedia data, such as images, music and video. The important subject is to retrieve images, music and stories dynamically according to the user's impression given as “Kansei” information.

The field of “Kansei” information was originally introduced as the word “aesthetics” by Baumgarten in 1750. The aesthetics of Baumgarten had been established and succeeded by Kant with his ideological aesthetics [5]. In the research field of multimedia database systems, it is becoming important to deal with “Kansei” information for defining and extracting media data according to impressions and senses of individual users. In the field of “Kansei” database systems, the essential functions for dealing with “Kansei” in database systems can be summarized as follows:

- (1) Defining “Kansei” information to media data (metadata definition for media data).
- (2) Defining “Kansei” information for user's requests (metadata definition for user's requests (user's keywords) with “Kansei” information).
- (3) Computing semantic correlations between “Kansei” information of media data and a user's request
- (4) Adapting retrieval results according individual variation and improving accuracy of the retrieval results by applying a learning mechanism to metadata of media data (learning mechanism for metadata).

There are several research projects to realize these functions. In the design of the “Kansei” information for media data, the important issues are how to define and

represent the metadata of media data and how to search media data dynamically, according to user's impression and media data contents. Creation and manipulation methods of metadata for media data have been summarized in [5], [20].

As a semantic associative search method for multimedia database systems dealing with "Kansei" information, we have proposed the Mathematical Model of Meaning (MMM) [10], [11], [13].

The MMM is a basic model for realizing a semantic associative search method with context recognition mechanisms for computing semantic distances and correlations between different media data, information resources and words. One of the important applications, we have presented a semantic associative search system for images [11], [14].

The important feature of this model is that the data objects in databases are mapped into an orthogonal semantic space and extracted by a semantic associative search mechanism [11], [14]. This method realizes the computational machinery for recognizing the meaning of a keyword according to a context (context words) and obtaining the related data objects to the keyword in the given context.

The MMM is applied to a semantic image and music search, as a fundamental framework for representing the metadata and searching images and music. The main feature of this model is that the semantic associative search is performed unambiguously and dynamically in the orthogonal semantic space. This space is created for computing semantic equivalence or similarity between user's impression and image's metadata items which represent the features of image data.

We point out that context recognition is essentially needed for multimedia information retrieval. The meaning of information is determined by the relation between contents and the context. The machinery for realizing dynamic context recognition is essentially important for multimedia information acquisition.

The advantages and original points of the MMM are as follows:

- (1) The semantic associative media search based on semantic computation for words is realized by a mathematical approach. This media search method surpasses the search methods which use pattern matching for associative search. Users can use their own words for representing impression and data contents for media retrieval, and do not need to know how the metadata of media data of retrieval candidates are characterized in databases.
- (2) Dynamic context recognition is realized using a mathematical foundation. The context recognition can be used for obtaining multimedia information by giving the user's impression and the contents of the information as a context. A semantic space is created as a space for representing various contexts which correspond to its subspaces. A context is recognized by the computation for selecting a subspace.

Several information retrieval methods, which use the orthogonal space created by mathematical procedures like SVD (Singular Value Decomposition), have been proposed. The MMM is essentially different from those

methods using the SVD (e.g. the Latent Semantic Indexing (LSI)) method [3]. The essential difference is that our model provides the important function for semantic projections which realizes the dynamic recognition of the context. That is, the context-dependent interpretation is dynamically performed by computing the distance between different media data, information resources and words. The context-dependency is realized by dynamically selecting a subspace from the entire orthogonal semantic space, according to a context. In MMM, the number of phases of contexts is almost infinite (currently  $2^{2000}$  in the general English word space and  $2^{180}$  in the color-image space, approximately). For semantic associative computations of "Kansei" information in MMM, we have constructed several actual semantic spaces, such as the general English-word space in 2115 dimensions, the color-image space in 183 dimensions, and music space in 8 dimensions in the current implementations.

We have applied this method to several multimedia database applications, such as image and music database search by impressionistic classification. We have introduced these research results in [11], [14] and [15]. Through these studies, we have created a new meta-level knowledge base environment by applying those methods to data retrieval, data integration and data mining [12], [16].

A learning mechanism is very important for database systems dealing with "Kansei" information to adapt search results according to individual variation and to improve accuracy of the search results. Generally, multimedia database systems dealing with "Kansei" information might not always select accurate and appropriate media data from databases, because the judgment of accuracy for the search results is highly related to individual variation. In the learning process, if inappropriate search results for a request are extracted by the system, accurate data items which must be the search results are specified as suggestions. Then, the learning mechanism is applied to the system to extract the appropriate retrieval results in subsequent requests. We have designed several database systems dealing with "Kansei" information for searching and extracting media data according to the user's impression and the image's contents. Those systems provide learning mechanisms for supporting adaptability to individual variations in "Kansei" [11], [14].

In this paper, we present a new concept of "automatic decorative-media creation" and a semantic associative computation method realizing automatic sub-media data selection and main-media decoration for representing a main-media object as decorative multimedia, as shown in **Figure 1**.

The important features of this method are summarized as follows:

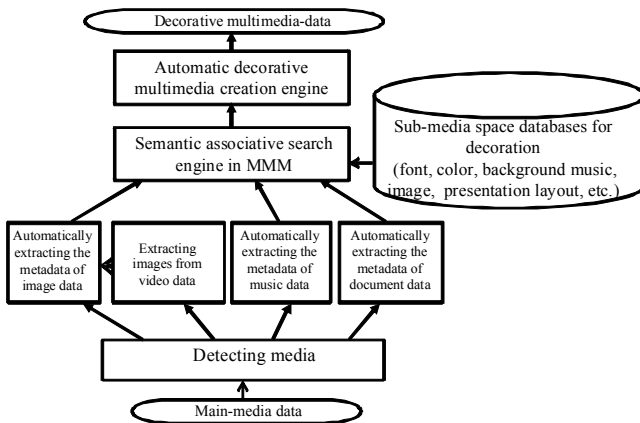
- (1) Main-media representation is automatically decorated by sub-media data with the semantic associative computation between a main-media object and sub-media data. The "Kansei" information of the main-media object is used as a context to compute semantic association and selection to the sub-media data in MMM.
- (2) Each main-media object is mapped in the media-specific semantic space corresponding to the

main-media, and then, it is mapped onto the related sub-media space by using semantic associative computation in MMM, through a media-transmission space. Two semantic spaces and a media-transmission space are used for computing semantic correlations between different media data in our automatic decorative-multimedia creation.

This method will lead to various applications for decorative-multimedia creation as follows:

- (A-1) “decorative music rendering with images, according to the impression and “Kansei” of the music,
- (A-2) “decorative video rendering with color visualization according to impression transition of a video story”,
- (A-3) “decorative text (novel) representation with the appropriate fonts, according to the impression of a story.
- (A-4) “decorative image presentation with music according to the impression and “Kansei” of the image,
- (A-5) “automatic decoration of a room with appropriate room lighting, according to the situation and context of the room,
- (A-6) “automatic Web-page decoration with appropriate fonts and colors according to the impression of the page content.

In this paper we show several significant applications of the semantic associative computation method for “automatic decorative-media creation” related to (A-1) and (A-2).



**Figure 1: The overview of automatic decorative-multimedia creation**

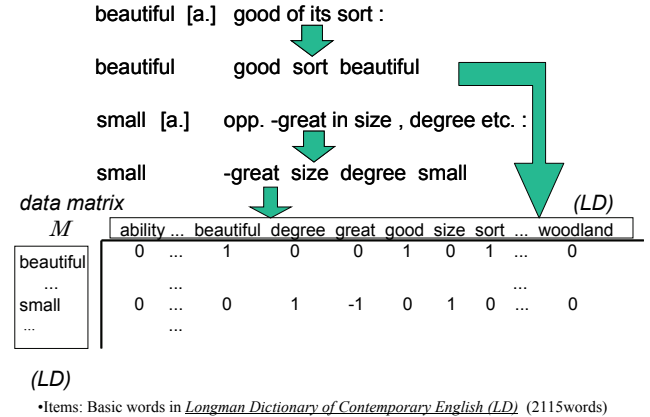
## 2 The Semantic Associative Search Method

In this section, the outline of the Mathematical Model of Meaning (MMM) is briefly reviewed. This model has been presented in [10], [11] and [13] in detail.

In the Mathematical Model of Meaning, an orthogonal semantic space is created for realizing the semantic associative search. Retrieval candidates and queries are mapped onto the semantic space. The semantic associative search is performed by calculating correlations of the retrieval candidates and queries on the semantic space.

### 2.1 Creation of the semantic space

To create the semantic space, a dictionary is selected and utilized. We refer to the basic words that are used to explain all the vocabulary entries in the dictionary as features [1]. For example, in a dictionary, an entry term “beautiful” is explained by the features “good,” “sort,” “beautiful” etc. as shown in Figure 2.



**Figure 2: The matrix  $M$  for semantic space creation in MMM (The matrix is constructed based on a dictionary.)**

When  $m$  terms are given as the vocabulary entries in the dictionary and  $n$  features are utilized to explain all these terms, an  $m$  by  $n$  matrix  $M$  is constructed for the space creation. Each term in the matrix  $M$  is characterized by the  $n$  features.

For example, the term “beautiful” is characterized by the features “good,” “sort,” “beautiful,” etc. When a feature, for example, “good,” is used for explaining the term “small,” the value of the entity at the row of “beautiful” and the column “good” is set to “1” as shown in Figure 2.

In the case of “small,” for example, in Figure 2, the term is characterized by the features “great,” “size,” “degree,” etc. When a feature, for example, “degree,” is used for explaining the term “small,” the value of the entity at the row of “small” and the column “degree” is set to “1”. If a feature is used as the negative meaning, for example, the feature “great,” the column corresponding to this feature is set to the value “-1.” If features are not used to explain terms, the columns corresponding to those features are set to “0.” As the features “ability” and “beautiful” are not used to explain the term “small,” the characterized value of these two features is “0.”

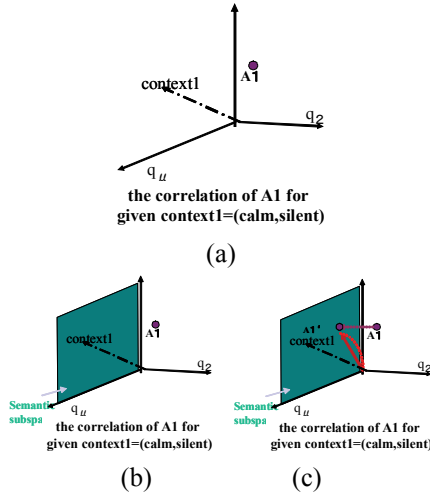
By using this matrix  $M$ , the orthogonal space is created as the semantic space based on a mathematical method.

### 2.2 An overview of the Mathematical Model of Meaning

In the Mathematical Model of Meaning, an orthogonal semantic space is created for semantic associative search. Retrieval candidates and queries are mapped onto the semantic space. The semantic associative search is performed by calculating the correlation of the retrieval candidates and the queries on the semantic space in the following steps:

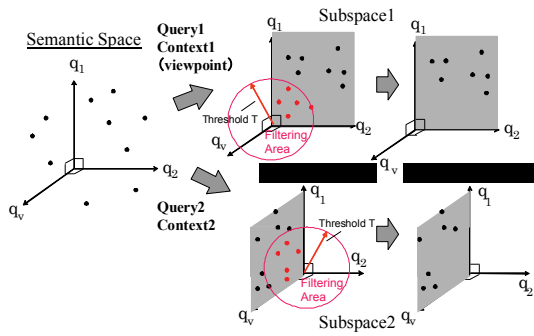
- (1) A context represented as a set of impression words is given by a user, as shown in Figure 3 (a)..

- (2) A subspace is selected according to the given context as shown in **Figure 3 (b)**.
- (3) Each information resource is mapped onto the subspace and the norm of  $A_1$  is calculated as the correlation value between the context and the information resource, as shown in **Figure 3 (c)**.



**Figure 3: Semantic associative search in MMM**

In MMM, the semantic interpretation is performed as projections of the semantic space dynamically, according to contexts, as shown in **Figure 4**.



**Figure 4: Semantic interpretation according to contexts in MMM**

### 2.3 Metadata structures in MMM

In the MMM, an orthogonal semantic space is defined and information resources are mapped onto the space for computing semantic correlations between associated information resources according to various contexts.

To compute semantic correlations, context words that represent the user's impression and data contents are given as a context. According to these context words, a "semantic subspace" is selected dynamically. Then, the most related information resource to the context is extracted by computing semantic correlations in the selected semantic subspace.

Metadata items are classified into three different kinds. The first kind of metadata items is used for the creation of the orthogonal semantic space, which is used as a search space for semantic associative search. These data items are referred to as "data-item for space creation."

The second kind of metadata items is used as the metadata items of the information resources, which are the candidates for semantic associative search. These

metadata items are referred to as "metadata for information resources."

The third kind of metadata items is used as context words, which represent user's impression and data contents in semantic associative search. These metadata items are referred to as "metadata for contexts."

The metadata structures in the MMM are summarized as follows:

- (1) Creation of the semantic space:

To provide the function of semantic associative search, basic information on  $m$  data items ("data-items for space creation") is given in the form of a matrix. Each data item is provided as fragmentary metadata which are independently represented one another. No relationship between data items is needed to be described. The information of each data item is represented by  $n$  features as  $n$ -dimensional vector. The  $m$  basic data items are given in the form of an  $m$  by  $n$  matrix  $M$ . For given  $m$  basic data items, each data item is characterized by  $n$  features. Then, each column of the matrix is normalized by the 2-norm in order to create the matrix  $M$ .

The eigenvalue decomposition of  $M^T M$  is computed.

$$M^T M = Q \begin{bmatrix} \lambda_1 & & & \\ & \ddots & & \\ & & \lambda_v & \\ & & & 0 \\ & & & & \ddots \\ & & & & & 0 \end{bmatrix} Q^T, 0 \leq v \leq n.$$

The orthogonal matrix  $Q$  is defined by

$$Q = (q_1, q_2, \dots, q_n)^T.$$

We call the eigenvectors "semantic elements." Here, all the eigenvalues are real and all the eigenvectors of  $\{q_1, q_2, \dots, q_v\}$  are mutually orthogonal because the matrix  $M^T M$  is symmetric.

The orthogonal semantic space  $MDS$  is created as a linear space generated by linear combinations of  $\{q_1, q_2, \dots, q_v\}$ . We note that  $\{q_1, q_2, \dots, q_v\}$  is an orthogonal basis of  $MDS$ .

The number of the semantic elements is  $2^v$ , and accordingly it implies that  $2^v$  different phases of meaning can be expressed by this formulation.

(In this space, a set of all the projections from the orthogonal semantic space to the invariant subspaces (eigen spaces) is defined. Each subspace represents a phase of meaning, and it corresponds to a context.)

- (2) Representation of information resources and contexts in  $n$ -dimensional vectors:

Each of the information resources is represented in the  $n$ -dimensional vector whose elements correspond to  $n$  features used in (1). These vectors are used as "metadata for information resources". The information resources are the candidates for semantic associate search in this model. Furthermore, each of context words, which are used to represent the user's impression and data contents in semantic associative search, is also represented in the  $n$ -dimensional vector. These vectors are used as "metadata for contexts."



## 2.4 The outline of semantic associative search in MMM

The outline of the MMM is expressed as follows:

- (1) A set of  $m$  words is given, and each word is characterized by  $n$  features. That is, an  $m$  by  $n$  matrix  $M$  is given as the data matrix.
- (2) The correlation matrix  $M^T M$  with respect to the  $n$  features is constructed from the matrix  $M$ . Then, the eigenvalue decomposition of the correlation matrix is computed and the eigenvectors are normalized. The orthogonal semantic space  $MDS$  is created as the span of the eigenvectors which correspond to nonzero eigenvalues.
- (3) Context words are characterized by using the  $n$  features and representing them as  $n$ -dimensional vectors.
- (4) The context words are mapped into the orthogonal semantic space by computing the Fourier expansion for the  $n$ -dimensional vectors.
- (5) A set of all the projections from the orthogonal semantic space to the invariant subspaces (eigen spaces) is defined. Each subspace represents a phase of meaning, and it corresponds to a context or situation.
- (6) A subspace of the orthogonal semantic space is selected according to the user's impression expressed in  $n$ -dimensional vectors as context words, which are given as a context represented by a sequence of words.
- (7) The most correlated information resources to the given context are extracted in the selected subspace by applying the metric defined in the semantic space.

## 3 Semantic Associative Search for Image Media

The MMM can be used to realize a semantic associative search system for image media.

The basic function of semantic associative search is provided for context dependent interpretation. This function performs the selection of the semantic subspace from the semantic space. When a sequence  $s'$  of context words for determining a context is given to the system, the selection of the semantic subspace is performed. This selection corresponds to the recognition of the context, which is defined by the given context words. The selected semantic subspace corresponds to a given context. The metadata item for the most correlated image to the context in the selected semantic subspace is extracted from the specified image data item set. Semantic associative search is performed by the following procedure:

1. When a sequence of the context words for determining a context (the user's impression and the image's contents) are given, the Fourier expansion is computed for each context word, and the Fourier coefficients of each context word with respect to each semantic element are obtained. This corresponds to seeking the correlation between each context word and each semantic element.
2. The values of the Fourier coefficients for each semantic element are summed up to find the

correlation between the given context words and each semantic element.

3. If the sum obtained in the step 2 in terms of each semantic element is greater than a given threshold, the semantic element is employed to form the semantic subspace. This corresponds to recognizing the context which is determined by the given context words.
4. By using the norm calculation as a metric in the semantic subspace, the metadata item for the image with the maximum norm is selected among the candidate metadata items for images in the selected semantic subspace. This corresponds to finding the image with the greatest association to the given context.

## 4 An automatic media-decoration model

In this section, we present a new concept of "automatic decorative-multimedia creation" by applying the MMM to automatic sub-media data selection for decorating a main-media object. To realize this concept, we define an "automatic media decoration model" with semantic spaces and media-decoration functions. The overview of this model is shown in Figure 5.

### 4.1 Basic semantic spaces and a media-transmission space

We define two semantic spaces and a media-transmission space (matrix) for computing semantic correlations between main-media objects and sub-media.

- (1) M-Space (Main-media semantic space)

Each main-media object or each impression word expressing impression of a main-media object is defined as an M-Space vector with  $m$  main-media-features.

- (2) S-Space (Sub-media semantic space):

Each sub-media object or each impression word expressing impressions in a sub-media object is defined as an S-Space vector with  $n$  sub-media-features. In this space, various sub-media objects are mapped in advance, as retrieval candidates, in S-Space for decorating the main-media object.

- (3) MS-Space (Main-media and Sub-media transmission space):

Each of  $m$  features of Main-media is expressed in the  $n$  features of Sub-media in the MS space. The MS-space is defined as a  $(m, n)$  matrix for transmitting an M-space vector into its corresponding S-space vector.

### 4.2 Basic functions for media decoration:

In this method, basic functions for decoration of a main-media object with sub-media are defined:

Step-1: maps a target main-media object onto the M-Space as the M-space vector for the decoration target, by expressing the object as an  $m$ -dimensional vector with the  $m$  features.

Step-2: computes correlation values between the M-space vector and each sub-media feature in the MS-Space (Main-media and Sub-media transmission space) by the Mathematical Model of Meaning (MMM), and creates an

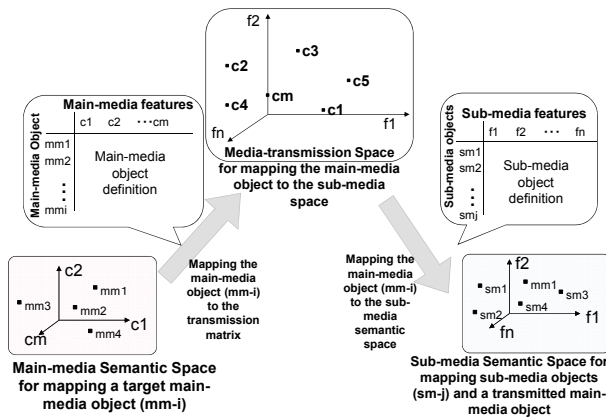
S-Space vector (target-S-Space vector) as the transmitted vector of the main-media object.

**Step-3:** maps the target S-Space vector expressing the main-media object onto the S-Space.

**Step-4:** executes the semantic associative search processes by the MMM between the target S-Space vector and the candidate sub-media objects which have been mapped onto the S-Space. (In MMM, the target S-Space vector is mapped as a context vector in S-Space, and candidate sub-media objects are mapped as retrieval candidates in S-Space.)

**Step-5:** outputs semantic ranking of sub-media objects as the result of our semantic associative search processes, and selects one of the sub-media objects with high correlation values to the target main-media object.

**Step-6:** renders the target main-media object with the selected sub-media object for decorating the main-media presentation with the selected sub-media data.



**Figure 5: Automatic decorative-multimedia creation by the semantic associative computation method**

## 5. Applications of “automatic decorative multimedia creation”

In this section, we present several applications of the automatic decorative-multimedia creation by our semantic associative computation method.

### (1) Music decoration with images:

This application is automatic “music decoration with images,” as shown in **Figures 6 and 7**. This decoration process consists of 6 steps.

**Step-1:** generates the metadata (in a form of a vector in the “music semantic space”) of music-media object, as a main-media object.

Our research project has proposed several automatic impression metadata generation methods for music [8], [21]. In this step, we use the metadata generation method to create impression metadata of music data [8]. This method applies the music psychology by K. Hevner [6], [7] to automatic extraction of music impression.

**Step-2:** generates the metadata (in a form of a vector in the “image semantic space”) of each image-media object in the image collection, as the collection of sub-media object.

Our research project has also proposed several automatic impression metadata generation methods for

images. In this step, we apply one of the metadata generation methods to create metadata of image data [11], [14].

**Step-3:** maps the metadata of the music-media object to the metadata in the image-media space, by using the MS-space (“Main-media and Sub-media transmission space”).

In this step, we apply a main-media and sub-media transformation space by using the relationships between the features of music and those of images in colors, which are created by artists and psychologists. This space consists of the correlations between the impression words of music and images.

**Step-4:** calculates semantic correlations between the music-media object and image-media objects in the “image semantic space.”

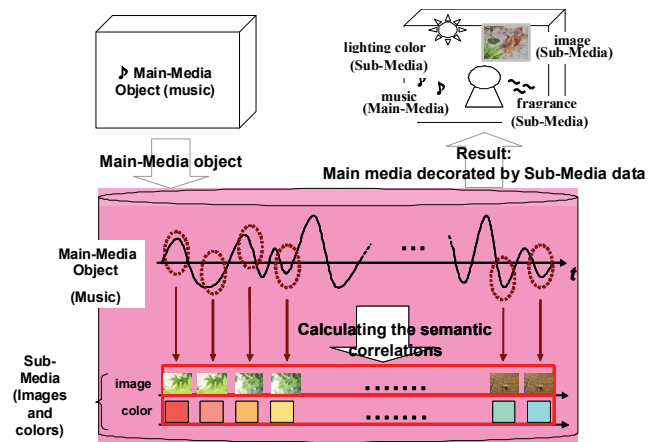
In this step, we apply the MMM to calculate the impression correlation between music and images.

**Step-5:** outputs the semantic ranking of image-media objects as the result of our semantic associative computation process, and selects image-media objects with high correlation values to the target music-media object.

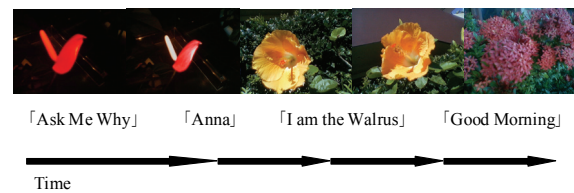
This step outputs the correlation values of image data in ascending order.

**Step-6:** renders the music-media object with the selected images as the music-media decorated with images.

This step is a rendering process for music decorated with images.



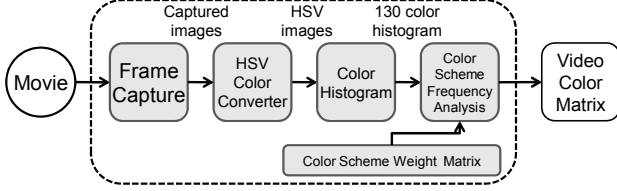
**Figure 6: Music decoration with images by automatic decorative-multimedia creation**



**Figure 7: Decoration for music-collections (the Beatles music collection) with images**

### (2) Color-based impression analysis for video and decoration with “Kansei” information

We have designed an experimental system for video analysis in terms of “Kansei” information expressed by the color-impression, as shown in **Figure 8**. This system realizes a color-based impression analysis for video-media. This system creates a story-line in impressions by analysing colors in each frame composing a video stream [19].



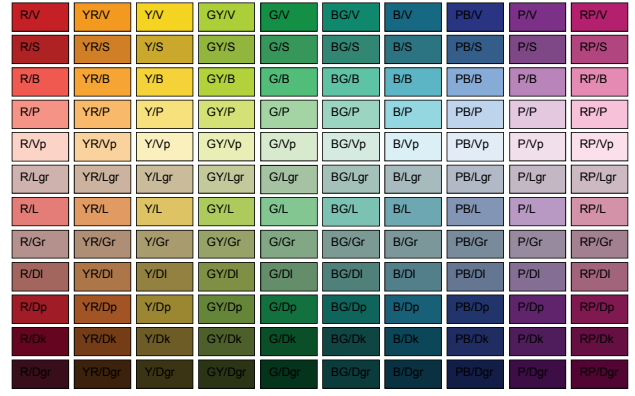
**Figure 8: Video analysis in colors for decorating video with “Kansei” information**

In this system, we have created the color-impression space using 120 chromatic colors and 10 monochrome colors defined in the “Theory of Colours” [4] and “Color Image Scale” [17] based on the Munsell color system. We used 183 words, which are also defined as cognitive scales of colors in the Color Image Scale, as impression words in the process of construction of color-impression space, shown in **Figure 9**. To generate a color histogram of each frame composing video, we used 130 colors in **Figure 10**, the same number of colors used in the color-impression space. This system converts RGB values to HSV values per pixel of each image, clusters them into the closest color of 130 colors, and calculates the percentage of each color to all pixels of the image [18]. The 183 color schemas, which correspond to 183 impression word sets, are defined as color-impression variations by using the 130 basic color features, as shown in **Figure 9**. By correlation calculations between 183 color schemas (183 impression word sets) and 130 basic colors, this system extracts the color-impression for each frame composing a video, and creates a sequence of color-impressions of the video along the timeline, as shown in **Figures 11, 12, 13** and **14**.

In the color-impression space used as the main-media space, this system creates a sequence of color-impressions of the video and applies it to decorate the video with sub-media, such as music or images, by using the semantic associative computation method.

		130 Basic Colors					
183 Color-Schemas		R/V	R/S	R/B	R/P	...	N/9.5
	cs1	1	0	0	0	...	0
	cs2	0.4	0	0	0	...	0
	cs3	0	0	0	0	...	0
	...	...	...	...	...	...	...
	cs183	0	0	0	0.6	...	0

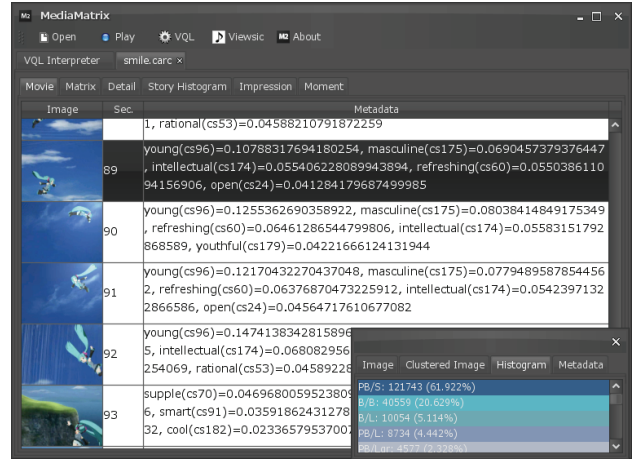
**Figure 9: Image-media features in 183 color schemas (183 impression word sets) related to 130 color variations**



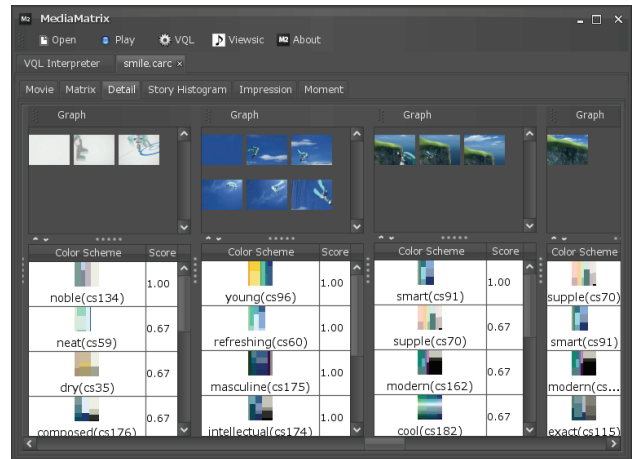
**Figure 10: Munsell 130 Basic Colors for extracting color schemas in impressions**

		183 Color-Schemas				
Time		cs1	cs2	cs3	...	cs <sub>m</sub>
	t1	0.2	0.4	0.2	...	0.1
	t2	0.1	0.1	0.0	...	0.2
	t3	0.1	0.3	0.25	...	0.4
	...	...	...	...	...	...
	t <sub>n</sub>	0.43	0.33	0.11	...	0.04

**Figure 11: The video-media story expressed in 183 color schemas (183 impression word sets) along the timeline**



**Figure 12: Video analysis for expressing 183 color schemas (183 impression word sets) in each frame composing the video**



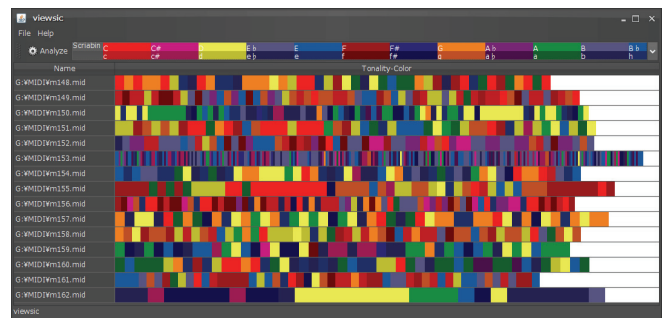
**Figure 13: Video analysis with 183 color schemas (183 impression word sets) in each scene**



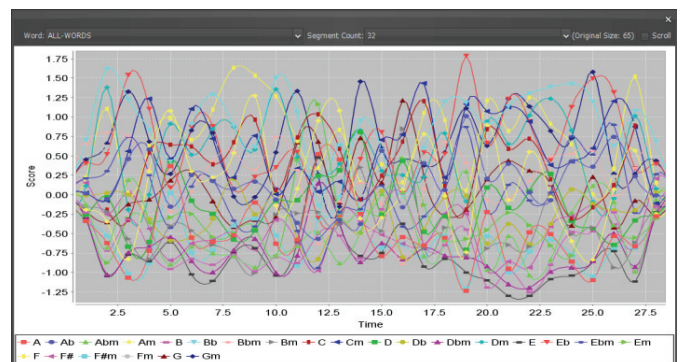
**Figure 14: Video-media decoration with impression-transition in color schemas (183 impression word sets) along the timeline**

### (3) Music-media decoration with tonality-transition in colors

The experimental system realizes music-media decoration with colors along tonality-transition in music. This system renders music with the visualization of tonality-transition in colors [9]. **Figure 15** shows a music-media decoration with tonality-transition in colors for 15 music compositions, J.S.Bach's Invention No.1—No.15. The system analyses a MIDI file as a data source for each music composition and generates music features extracted by tonality-analysis, musical segment analysis and tempo analysis. For analysing tonality transition of music, we have applied the Krumhansl-Schmuckler algorithm to MIDI files as a tonality-finding method. **Figure 16** represents the tonality transition of music "Sarabande" in colors. Those music features of tonality are automatically extracted and mapped to the music-media space along the timeline. Then, our semantic associative computation method is applied to music rendering for decorating it with colors.



**Figure 15: Music-media decoration for J.S.Bach's Invention No.1—No.15 with tonality-transition in colors along the timeline**



**Figure 16: Music-media decoration with tonality transition of "Sarabande" in colors along the timeline**

## 5 Conclusion

In this paper, we have presented a new concept of "automatic decorative-multimedia creation" and a semantic associative computation method. This method realizes automatic main-media decoration with sub-media data selection for representing a main-media object as decorative multimedia.

We have reviewed the Mathematical Model of Meaning (MMM) applied to automatic decorative-multimedia creation as a basic model for semantic associative computing for multimedia creation. This model is used as a basic computational model for computing semantic correlations between different media data with context computation mechanisms.

This paper has also defined an "automatic media decoration model" with semantic spaces and media-decoration functions. This model defines several functions for realizing automatic decorative-multimedia creation by semantic associative computations for images, music and video.

We have implemented several experimental decorative-multimedia creation systems for music, images and video media to clarify the feasibility of "automatic decorative-multimedia creation" and a semantic associative computation method. As our future work, we realize automatic decorative-multimedia creation environments for various application fields. We also create an on-the-fly automatic decorative-multimedia creation system for dynamic video representation decorated with various media data.



## Acknowledgments

We would like to thank Prof. Takashi Kitagawa (University of Tsukuba) as a co-designer of the Mathematical Model of Meaning (MMM). We would also like to thank our MDBL project members in Keio University for valuable experiments in multimedia database systems.

## 6 References

- [1] "Longman Dictionary of Contemporary English", Prentice Hall College Div, 2006.
- [2] Chen, X. and Kiyoki, Y., "A Visual and Semantic Image Retrieval Method Based on Similarity Computing with Query-Context Recognition," Information Modelling and Knowledge Bases, Vol. XVIII, pp.245-252, May 2007.
- [3] Deerwester, S., Dumais, S. T., Landauer, T. K., Furnas, G. W. and Harshman, R. A., "Indexing by latent semantic analysis," Journal of the American Society for Information Science, Vol.41, No.6, pp.391-407, 1990.
- [4] Goethe, W. J., "Theory of Colours," trans. Charles Lock Eastlake, Cambridge, Massachusetts: The M.I.T. Press, 1982.
- [5] Harada, A. (eds.), "Report of modeling the evaluation structure of KANSEP", Univ. of Tsukuba, 1997.
- [6] Hevner, K., "Expression in Music: A Discussion of Experimental Studies and Theories," Psychological Review, Vol.42, pp.186-204, 1935.
- [7] Hevner, K., "Experimental Studies of the Elements of Expression in Music," American Journal of Psychology, Vol.48, pp.246-268, 1936.
- [8] Ijichi, A. and Kiyoki, Y., "A Kansei Metadata Generation Method for Music Data Dealing with Dramatic Interpretation," Information Modelling and Knowledge Bases, Vol.XVI, IOS Press, pp. 170--182, (May, 2005).
- [9] Imai, S., Kurabayashi, S. and Kiyoki, Y., "A Music Retrieval System Supporting Intuitive Visualization by the Color Sense of Tonality," Proceedings of the 24th IASTED International Multi-Conference DATABASES AND APPLICATIONS (DBA2006), pp.153-159, February 2006.
- [10] Kitagawa, T. and Kiyoki, Y., "A mathematical model of meaning and its application to multidatabase systems," Proc. 3rd IEEE International Workshop on Research Issues on Data Engineering: Interoperability in Multidatabase Systems, pp.130-135, April 1993.
- [11] Kiyoki, Y., Kitagawa, T. and Hayama, T., "A metadatabase system for semantic image search by a mathematical model of meaning," ACM SIGMOD Record, Vol.23, No. 4, pp.34-41, Dec. 1994.
- [12] Kiyoki, Y. and Kitagawa, T., "A semantic associative search method for knowledge acquisition," Information Modelling and Knowledge Bases (IOS Press), Vol. VI, pp.121-130, 1995.
- [13] Kiyoki, Y., Kitagawa, T. and Hitomi, Y., "A fundamental framework for realizing semantic interoperability in a multidatabase environment," International Journal of Integrated Computer-Aided Engineering, Vol.2, No.1(Special Issue on Multidatabase and Interoperable Systems), pp.3-20, John Wiley & Sons, Jan. 1995.
- [14] Kiyoki, Y., Kitagawa, T. and Hayama, T., "A Metadatabase System for Semantic Image Search by a Mathematical Model of Meaning," Multimedia Data Management -- using metadata to integrate and apply digital media --, McGrawHill(book), A. Sheth and W. Klas(editors), Chapter 7, March 1998.
- [15] Kiyoki, Y., "A Semantic Associative Search Method for WWW Information Resources," Proceedings of 1ST International Conference on Web Information Systems Engineering, (invited paper), 2000.
- [16] Kiyoki, Y. and Ishihara, S., "A Semantic Search Space Integration Method for Meta-level Knowledge Acquisition from Heterogeneous Databases," Information Modelling and Knowledge Bases (IOS Press), Vol. 14, pp.86-103, May 2002.
- [17] Kobayashi, S., "Color Image Scale" (The Nippon Color & Design Research Institute ed., translated by Louella Matsunaga, Kodansha International, 1992).
- [18] Sasaki, S., Itabashi, Y., Kiyoki, Y. and Chen, X., "An Image-Query Creation Method for Representing Impression by Color-based Combination of Multiple Images," Proceedings of the 18th European-Japanese Conference on Information Modelling and Knowledge Bases, pp. 105-112, June 2008.
- [19] Sato, Y. and Kiyoki, Y., "A semantic associative search method for media data with a story," Proceedings of the 18th IASTED International Conference on Applied Informatics, pp., Feb., 2000.
- [20] Sheth, A. and Klas, W. (eds.) "Multimedia Data Management - Using Metadata to Integrate and Apply Digital Media," MacGraw-hill, March 1998.
- [21] Yara, F., Yoshida, N., Sasaki, S. and Kiyoki, Y., "A Continuous Media Data Rendering System for Visualizing Psychological Impression-Transition," The 10th IASTED International Conference on Internet and Multimedia Systems and Applications (IMSA2006), pp. 32 - 40, Aug. 2006.



# Modeling Natural Language Communication in Database Semantics

Roland Hausser<sup>1</sup>

<sup>1</sup> Abteilung Computerlinguistik  
Universität Erlangen-Nürnberg (CLUE)  
Bismarckstr. 6 & 12, D-91054 Erlangen, Germany  
Email: rrrh@linguistik.uni-erlangen.de

## Abstract

Database Semantics (DBS) is a computational model of how communicating with natural language works. Defined at a level of abstraction which may be applied to natural and artificial agents alike, such a model is a precondition for achieving free human-machine communication in natural language, and thus has great potential for a wide range of practical applications.

The basic functionality of DBS is a cognitive agent's turn taking between the *speaker mode* (mapping content stored in memory into language surfaces) and the *hearer mode* (mapping surfaces into content for storage in memory). DBS is defined in terms of (i) the data structure of flat feature structures, (ii) the algorithm of time-linear LA-grammar, and (iii) the database schema of a classic network database.

**Keywords:** language production; language interpretation; turn taking; data structure; algorithm; database schema; parts of speech; semantic relations; functor-argument structure; coordination; elementary, phrasal, and clausal level; hearer mode; speaker mode; proplets; LA-grammar; Word Bank

## 1 Linguistic Background

The analysis of natural languages and of natural language communication has long been an interdisciplinary enterprise involving linguistics, philosophy, psychology, physiology, neurology, sociology, mathematics, and computer science. As a consequence, we are faced today with a vast patchwork of different theories, topics, and empirical analyses. Rather than attempting an overview of current work, which by necessity would have to be selective, let us begin with those aspects of traditional<sup>1</sup> language analysis which are (i) uncontroversial and (ii) important for understanding the remainder of this paper.

### 1.1 Compositionality

The sentences of natural language are built from word forms. For example, the sentence *Julia knows John* is built from the word forms *Julia*, *knows*, *John* and the full stop. Each word form has a surface and a lexical meaning.<sup>2</sup> By ordering word form surfaces of

a given natural language into a grammatically well-formed sequence, the associated lexical meanings are combined into a sentence meaning.

Lexical meanings are analyzed in the linguistic field of *lexical semantics*, while the derivation of sentence meanings is analyzed in *compositional semantics*. The relation between lexical semantics and compositional semantics is described by the Fregean Principle, named after Gottlob Frege (1848–1925):

#### 1.1.1 FREGEAN PRINCIPLE

The meaning of a complex expression is a function of the meaning of its parts and their mode of composition.

For example, the two sentences

The dog bites the man.  
The man bites the dog.

consist of exactly the same parts (English word form surfaces), but differ in their word order (mode of composition) – which is one reason why two sentences may turn out to have different sentence meanings.

Furthermore, the two sentences

The dog is chasing a cat.  
The dog is chasing a squirrel.

have the same word order (mode of composition), but differ in one of their parts (namely the respective last word form) – which is the other reason why two sentences may turn out to have different sentence meanings.<sup>3</sup>

### 1.2 Parts of Speech

The words of a natural language are divided into *content words*, for example, *table*, *write*, or *beautiful*, and *function words*, for example, *the*, *and*, or *before*. The number of content words in a natural language is several ten thousands, while the number of function words is only a few hundred. However, while a function word like *the* is used very often, comprising 6% of the running word forms in a corpus, a content word like *table* has a comparatively much lower frequency.

The most basic way of classifying a word is according to its *part of speech*. The basic parts of speech are the *nouns*, the *verbs* and the *adjectives*.<sup>4</sup>

Consider the following examples:

of meaning in an agent-oriented approach see FoCL'99, Sects. 3.2–3.4, 4.2, 4.3, 6.1, 6.2, 19.1; NLC'06, Sects. 4.2–4.4, 5.4–5.6.

<sup>3</sup>In order for the Fregean Principle to hold it is important to distinguish (i) between the literal meaning of language expressions (meaning<sub>1</sub>) and the speaker meaning of utterances (meaning<sub>2</sub>) and (ii) between the unanalyzed and the analyzed surface. As shown in FoCL'99, Sects. 4.2–4.5, the Fregean Principle applies solely to the meaning<sub>1</sub> of analyzed surfaces.

<sup>4</sup>Additional parts of speech used in the classic grammar of Latin are *adverb*, *pronoun*, *preposition*, *conjunction*, and *interjection*.

Copyright ©2009, Australian Computer Society, Inc. This paper appeared at the Sixth Asia-Pacific Conference on Conceptual Modelling (APCCM 2009), Wellington, New Zealand, January 2009. Conferences in Research and Practice in Information Technology (CRPIT), Vol. 96, Markus Kirchberg and Sebastian Link, Ed. Reproduction for academic, not-for profit purposes permitted provided this text is included.

<sup>1</sup>For a representative contemporary example see Benner 2008.

<sup>2</sup>For example, the meaning of the English surface *tree* has the surface *arbre* in French and *Baum* in German. For a detailed analysis

### 1.2.1 PARTS OF SPEECH AT ELEMENTARY LEVEL

1. *noun*: Examples are *Julia* (proper name) and *table* (common noun).
2. *verb*: Examples are *sleep* (one-place), *write* (two-place), and *give* (three-place).
3. *adjective*: Examples are *brief* (adnominal) and *briefly* (adverbial).

In the linguistic school of structuralism, the parts of speech are justified by means of substitution and movement tests.<sup>5</sup> For example, *Julia* and *John* have the same part of speech because substituting *John* for *Julia* in the sentence *Julia slept briefly*, results in another well-formed sentence with a similar meaning. In contrast, substituting *Julia* with *wrote* would result in an ungrammatical expression. However, substituting *wrote* in *John wrote a letter*, with *typed* renders another well-formed sentence, indicating that *wrote* and *typed* are of the same part of speech.

In addition to the classification of a word in terms of its part of speech, there is the distinction between the different *word forms* of a word. For example, *book*, *book's*, *books*, and *books'* are different forms of the noun *book*. Similarly, *learn*, *learns*, *learned*, and *learning* are different forms of the verb *learn*. And *quick*, *quickly*, *quicker*, and *quickest* are different forms of the adjective *quick*. Strictly speaking, sentences, texts, and dialogues of natural language consist of word forms rather than words.<sup>6</sup>

Forms of elementary content and function words may be combined into complex parts of speech at the *phrasal* and the *clausal* level:

### 1.2.2 PARTS OF SPEECH AT THE PHRASAL LEVEL

1. *noun*: An example is *the pretty young girl*, which consists of a determiner (function word), two elementary adjectives, and an elementary noun, the latter being content words.
2. *verb*: An example is *could have been sleeping*, which consists of (forms of) a modal verb, two auxiliaries, and a main verb.
3. *adjective*: An example is *for five minutes*, which can be used adnominally as well as adverbially and consists of a preposition, a determiner, and a noun, the latter being a content word.

Again, the functional equivalence between elementary and clausal parts of speech may be shown by substituting one for the other. For example, the sentence *Julia slept briefly*, consists of an elementary noun, an elementary verb, and an elementary adjective. Replacing *Julia* with *The pretty young girl* results in a similar sentence with a phrasal noun, namely *the pretty young girl slept briefly*. By replacing the elementary verb and adjective with phrasal counterparts as well, we might get *the pretty young girl could have been sleeping for five minutes*. – which has the same grammatical structure at the phrasal level as *Julia slept briefly*, has at the elementary level.

At the highest level, elementary and phrasal parts of speech are combined into clausal parts of speech, which serve as nouns and as adjectives. The part of speech *verb* at the clausal level is equivalent to a complete clause or sentence.

Adverbs are treated here as the adverbial use of adjectives (in contradistinction to the adnominal use); pronouns like *you* or *he* are treated as nouns; prepositions are treated as function words which make phrasal adjectives, conjunctions are treated as function words which make coordinations (parataxis) and clausal nouns or adjectives (hypotaxis). A treatment of interjections is omitted.

<sup>5</sup>Cf. FoCL'99, Sect. 8.4.

<sup>6</sup>For a more detailed discussion see FoCL'99, Chapt. 13.

### 1.2.3 PARTS OF SPEECH AT THE CLAUSAL LEVEL

1. *noun*: An example is that *Julia read a book*, as in *That Julia read a book pleased her mother*. – which has a similar grammatical structure as *Julia pleased her mother*. In other words, *that Julia read a book* may serve the same grammatical function as the elementary noun *Julia* or the phrasal noun *the pretty young girl*.
2. *adnominal adjective*: An example is *which Mary saw*, as in *The dog which Mary saw barked*. – which has a similar grammatical structure as *The black dog barked*. In other words, the adnominal clause *which Mary saw* has the same grammatical function as the elementary adnominal *black* or the phrasal adnominal *with the black fur*.
3. *adverbial adjective*: An example is *When Fido barked*, as in *When Fido barked Mary smiled* – which has a similar grammatical structure as *Recently Mary smiled*. In other words, the adverbial clause *When Fido barked* has the same grammatical function as the elementary adverb *recently* or the phrasal adverb *after her nap*.

## 1.3 Semantic Relations

In natural language, word forms are assembled into well-formed, meaningful expressions by means of only two kinds of semantic relations, namely (i) *functor-argument* and (ii) *coordination* structure. These semantic relations are based to a large extent on the parts of speech of the words and expressions, and are the topic of the linguistic disciplines of syntax and compositional semantics.

Functor-argument structure is used to build the different sentential moods, such as declarative, interrogative, and imperative in English. In a proposition (sentence), there is the obligatory relation between the functor (verb) and its arguments (nouns), and the optional relation between a verb or noun and its modifiers (adjectives).

For example, the functor-argument structure of *Julia knows John* is based on the lexical analysis of the verb form *knows* as a *two-place functor* and of the nouns *Julia* and *John* as *arguments*. Similarly, the functor-argument structure of *Julia slept* is based on the lexical analysis of the verb form *slept* as a *one-place functor* and *Julia* as the argument. And the functor-argument structure of *John gave Julia a flower* is based on the lexical analysis of *gave* as a *three-place functor* and the arguments *John*, *Julia*, and a *flower*.

Depending on the lexical valency structure of the verb, its arguments are distinguished with respect to their grammatical role. For example, in the sentence *Julia slept*, the argument *Julia* serves as the *subject*; in *Julia knows John*, *Julia* serves as the subject and *John* as the *object*; and in *John gave Julia a flower*, *John* serves as the subject, *Julia* as the *indirect object*, and a *flower* as the *direct object*. These distinctions apply also to arguments at the phrasal and the clausal level.

In order for a sentence to be complete, it must have as many arguments as required by its verb. It is in this sense that the arguments are *obligatory*. The modifiers, in contrast, are *optional*. For example, the sentence *Julia slept briefly*, is still complete without the adverbial modifier *briefly*. Similarly, the sentence *The black dog barked*, is still complete without the adnominal modifier *black*. The optional nature of modifiers applies also at the phrasal and clausal level.

While functor-argument structure assembles expressions of different parts of speech, *coordination* assembles word forms, phrases, and clauses of the same part of speech. For example, *Julia, Susanne,*



and John is an elementary noun coordination, bought, cooked, and ate is an elementary verb coordination, and quickly, tastefully, and professionally is an elementary adjective coordination in adverbial use.

Clausal coordination may be *intrasentential*, as in Julia slept, John sang, and Susanne dreamed. or *extrasentential*, as in Julia slept. John sang. Susanne dreamed. An intrasentential form are gapping constructions, such as Bob ate an apple, walked his dog, and read the paper (subject gap), Bob ate an apple, Jim a pear, and Bill a peach (verb gap), and Bob bought, Jim peeled, and Bill ate the peach. (object gap).<sup>7</sup>

The combination of the two semantic relations at the elementary, the phrasal and the clausal levels may result in constructions of arbitrary complexity and length. Intrasententially, functor-argument structure provides for subclause constructions of arbitrary complexity in the different sentential moods. Extrasententially, coordination provides for text or dialogue of arbitrary length.

#### 1.4 Word Order, Word Forms, Agreement

While the functor-argument and coordination structure are based semantically on the parts of speech, their coding depends on *word order*, *word forms*, and *agreement*. The role of these structural surface properties varies widely between different natural languages and is thus highly language-dependent.

For example, in English the grammatical role of the arguments is coded by means of word order: in *Julia knows John.*, the first argument is the subject and the second argument the object. Changing the order into *John knows Julia.* changes the subject and the object. Furthermore, in declarative sentences of English, the verb (functor) must be in post-nominative position.

In Russian, in contrast, the word order is free and the case role of the arguments is coded morphologically in the endings (suffixes) of the word forms. Thus the content of English *Julia knows John* has six surfaces in Russian, corresponding in word order to *Julia<sub>N</sub> knows John<sub>A</sub>*, *Julia<sub>N</sub> John<sub>A</sub> knows*, *Knows Julia<sub>N</sub> John<sub>A</sub>*, *Knows John<sub>A</sub> Julia<sub>N</sub>*, *John<sub>A</sub> knows Julia<sub>N</sub>*, and *John<sub>A</sub> Julia<sub>N</sub> knows.*, while the content of English *John gave Julia a flower* has a total of twenty-four corresponding surfaces in Russian.

Because of the fixed word order of English, the number of different word forms may be comparatively small. In common nouns, there is the distinction between singular and plural, e.g., *book* vs. *books* and between unmarked and genitive case, e.g., *book* vs. *book's* and *books* vs. *books'*. In finite verbs, there is the distinction between 3. person singular present tense, and the other persons and numbers (numeri) in present tense, e.g., *sings* vs. *sing*, and the distinction between present tense and past tense, e.g., *sing* and *sang*. In adjectives, there is the distinction between adnominal and adverbial use, e.g., *beautiful* and *beautifully*, and the distinction between the positive, comparative, and superlative, as in *fast*, *faster*, *fastest*.

The choice of a particular form of a word is partially semantic, as between *Julia knows John.* and *Julia knew John.*, and partially syntactic, as in *every girl* (singular) and *all girls* (plural). In the latter case, the correct choice of word forms is a matter of grammatical *agreement*. For example, in English there is agreement between the subject and the finite verb in the present tense, and between determiners like *a(n)*, *every*, *all*, *one*, *two*, *three*, etc., and their nouns.<sup>8</sup>

<sup>7</sup>Even though gapping constructions turn out to be of very low frequency in corpora, we know of no natural language in which they (i) don't occur and (ii) don't have strong grammatical intuitions with the native speakers. For a detailed analysis of coordination including gapping constructions see NLC'06, Chaps. 8 and 9.

<sup>8</sup>For a DBS analysis of "quantifiers" see NLC'06, Sect. 6.2.

#### 1.5 Automatic Word Form Recognition

For purposes of computational linguistics, the traditional distinctions of grammar at the level of elementary word forms may be represented as flat feature structures. A feature structure is defined as a set of attribute value pairs or avp. In DBS, a flat<sup>9</sup> feature structure representing word forms is called a *proplet*: just as a droplet is the smallest element in a sea of water, a proplet is the smallest element in a sea of propositions. Consider the following examples of lexical proplets, which illustrate the basic parts of speech with forms of content words:

##### 1.5.1 EXAMPLES OF LEXICAL PROPLETS

[sur: books noun: <i>book</i> cat: pl sem: count fnc: mdr: nc: pc: idy: prn:	[sur: swam verb: <i>swim</i> cat: s3' v sem: past arg: mdr: nc: pc: prn:	[sur: slowly adj: <i>slow</i> cat: adv sem: pos mdd: nc: pc: prn:
---	--	--

The first attribute *sur* (surface) has the language-dependent surface of the word as its value. The second attribute is called the *core attribute* of the proplet. It indicates the part of speech and has a language-independent meaning as its value, called the *core value*. Core values may be of the different kinds of sign *symbol*, *indexical*, and *name*.<sup>10</sup> For simplicity, the meanings are represented by corresponding words of English written in italics,<sup>11</sup> used here as place holders or names of the meaning in question.

The third attribute *cat* specifies grammatical properties relevant for the combinatorics of the word form, such as singular vs. plural in nouns, the valency positions of verbs, and the adnominal versus adverbial use in adjectives. The fourth attribute *sem* specifies morphologically coded aspects of meaning which are not directly relevant for the combinatorics of the word form, such as tense.

The remaining attributes are non-lexical and therefore have no values yet. The attributes *fnc* (functor), *mdr* (modifier), *nc* (next conjunct), and *pc* (previous conjunct) in nouns are called the *continuation attributes* which code the functor-argument and coordination relations of the proplet. Verbs and adjectives differ from nouns in that verbs have *fnc* (functor) and adjectives *mdd* (modified) as their obligatory continuation attribute.<sup>12</sup>

In addition to the continuation attributes there are the *bookkeeping attributes* *idy* (identity) in nouns and *prn* (proposition number) in nouns, verbs, and adjectives. Other bookkeeping attributes used only in the software are *trc* (transition counter) and *wrn* (word number). While lexical attributes have their values defined in the on-line lexicon and continuation attributes receive their values by the rules of syntactic-semantic parsing, bookkeeping attributes have numerical values assigned by the parsing engine.

The on-line definition of lexical proplets is the basis of *automatic word form recognition*. It is needed because for the computer an unanalyzed surface like *books* is merely a sequence of letters, with no essential difference between *books* and *skoob*, for example. By matching the unanalyzed surface with the *sur* value of

<sup>9</sup>A feature structure is called flat or non-recursive if it does not allow that values may themselves be feature structures.

<sup>10</sup>Cf. FoCL'99, Sect. 6.1.

<sup>11</sup>Basic meanings originate as the patterns (types) needed for the agent's recognition and action procedures, cf. FoCL'99, Sect. 3.2; NLC'06, Sects. 4.2–4.4.

<sup>12</sup>For a more detailed discussion see NLC'06, Sect. 4.1.

a lexical proplet, all the information contained in the proplet definition becomes available to the parser.

The main tasks of automatic word form recognition are *categorization* and *lemmatization*. Categorization is provided by the *cat* value of a proplet, and needed for controlling the combinatorial interaction within the functor-argument and the coordination structure. Lemmatization is provided by the *core* value, i.e., the value of the *noun*, *verb*, or *adj* attribute, and needed for access to the words' core meaning.<sup>13</sup>

## 2 Formal Systems of Grammatical Analysis

Based on automatic word form recognition, complex expressions of natural language may be grammatically analyzed by a computer in accordance with the Fregean Principle (cf. 1.1.1). For computational linguistics, this procedure has two aspects: (i) the formal grammatical analysis of complex expressions and (ii) the construction of an automatic analysis system, called parser, which uses the formal grammatical analysis as a *declarative specification*.

A declarative specification is important for software writing in general because it formulates the abstract solution to the task at hand, specifying the *necessary* properties of the software, in contrast to accidental properties such as the choice of programming language or idiosyncrasies of the programmer.<sup>14</sup> However, while using a formal grammar as a declarative specification is a necessary condition for implementing a transparent natural language parser, it is not sufficient to guarantee longterm success.

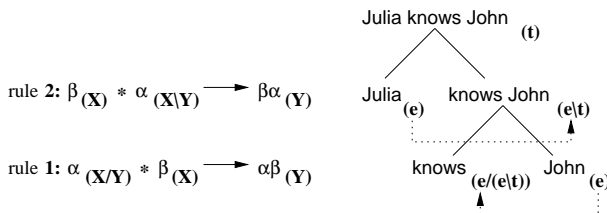
In addition to being formally explicit, the grammar must be of *low mathematical complexity* to run in real time. Furthermore, the grammar must turn out to be empirically adequate by supporting *fast upscaling* from a small system to one with an ever wider data coverage. Finally, the formal system must be *functionally complete* to support different applications of human-computer communication, whereby building a freely talking robot is the ultimate challenge.

### 2.1 Categorical Grammar

Designing a formal grammar raises the questions of (i) how exactly the word forms in a sentence should be combined and (ii) what the purpose of the result should be. To get an idea of how these questions have been approached in the past, let us briefly consider the most distinctive approaches proposed so far.

The historically first approach to a formal grammatical analysis of natural language is *Categorical grammar* as proposed by Bar Hillel (1953), based on a formal system by Leśniewski (1929) and Ajdukiewicz (1935). Consider the following example:

#### 2.1.1 BOTTOM-UP CATEGORIAL ANALYSIS OF Julia knows John.



The grammatical analysis is shown as the tree on the right, while the combination rules applying at the

input levels are shown on the left. The derivation starts with (i) categorized words like  $\text{knows}_{(e/(e,t))}$  and  $\text{John}_{(e)}$ . The rules consist of the variables  $\alpha$  and  $\beta$ , which match the surface of a categorized word, e.g.,  $\text{John}$ , and the variables  $X$  and  $Y$ , which match a category or part of a category, e.g.,  $(e \backslash t)$ . Categorized words or expressions may be combined if their categories can be matched by one of the two rules. The purpose is a semantic interpretation of functor-argument structure using set theory.

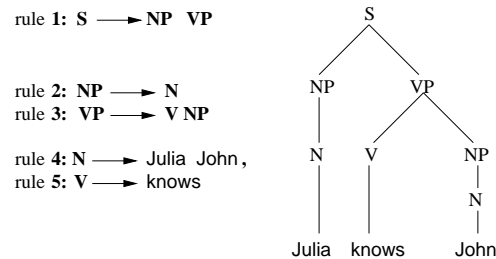
For example,  $\alpha_{X/Y}$  matches the categorized word  $\text{knows}_{(e/(e \backslash t))}$ , while  $\beta_X$  matches the categorized word  $\text{John}_{(e)}$ . Therefore, according to rule 1,  $\text{knows}_{(e/(e \backslash t))} * \text{John}_{(e)}$  may be combined into the expression  $\text{knows John}_{(e \backslash t)}$ . The category  $(e \backslash t)$  results from the category  $(e/(e \backslash t))$  of  $\text{know}$  by canceling the first  $e$  with the category  $(e)$  of  $\text{John}$  (see dotted arrow on the right of 2.1.1).

Next,  $\text{Julia}_{(e)}$  and  $\text{knows John}_{(e \backslash t)}$  are combined by rule 2 into  $\text{Julia knows John}_{(t)}$ , this time canceling the  $e$  in the category  $(e \backslash t)$  of  $\text{knows John}$ . The two rules differ only in that the expression with the complex category (the functor) precedes the expression with the canceling category (the argument) in rule 1, while in rule 2 the order is reversed.

### 2.2 Phrase Structure Grammar

The historically second approach to a formal grammatical analysis is *Phrase Structure grammar*, as proposed by Chomsky (1957, 1965), based on a formal system by Post (1936). Its purpose is to characterize the native speakers' innate language capability and to explain language acquisition by the child.

#### 2.2.1 TOP-DOWN PHRASE-STRUCTURE ANALYSIS OF Julia knows John.



The analysis uses rewrite rules of the form  $A \rightarrow B C$ . There are non-terminal symbols like  $S$  (for start or sentence),  $NP$  (for noun phrase),  $VP$  (for verb phrase),  $N$  (for noun), and  $V$  (for verb), and terminal symbols like  $\text{Julia}$ ,  $\text{John}$ , and  $\text{knows}$ . The derivation begins with the  $S$ , which rule 1 substitutes with  $NP VP$ . Then rule 2 substitutes  $NP$  with  $N$ , and rule 3 substitutes  $VP$  with  $V NP$ . Then rule 2 applies again, substituting the second  $NP$  with  $N$ . Finally the terminal rules 4 and 5 apply, replacing the nodes  $N$  and  $V$  with terminal symbols, i.e., with word surfaces.

The Categorical analysis 2.1.1 and the Phrase Structure analysis 2.2.1 have in common that they express the grammatical relations as a hierarchy such that a complex constituent (i.e., a phrasal or clausal part of speech) dominates its smaller parts. This hierarchy, called constituent structure, is based on the principle of *possible substitutions* and is reflected by the associated tree in terms of the *dominance* and *precedence* relations between the nodes. The assumption of constituent structure affects the grammatical derivation order: the phrasal and clausal parts of speech must be assembled first before they can be combined with each other like elementary parts of speech.

<sup>13</sup>An overview of different methods of automatic word form recognition is provided in FoCL'99, Chapt. 14.

<sup>14</sup>For a more detailed discussion see NLC'06, Sect. 1.2.

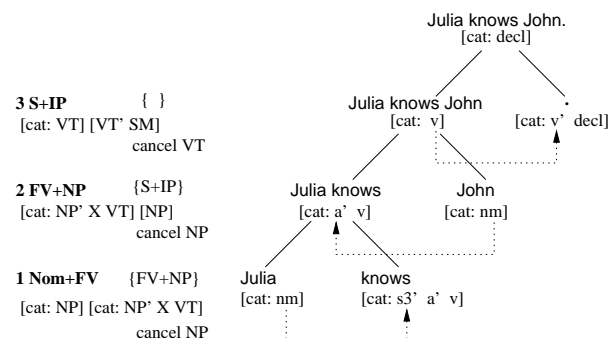
## 2.3 Left-Associative Grammar

According to constituent structure, it would not be correct to postulate a Phrase Structure rule like  $S \rightarrow NV$  because  $NV$  or the corresponding *Julia knows* is neither an elementary nor a phrasal nor a clausal part of speech. For the speaker, however, a sentence like *Julia knows John* begins by combining *Julia* and *knows* into *Julia knows*. Then *Julia knows* is combined with *John* into *Julia knows John*. Similarly for the hearer, who doesn't have to wait for the end of the sentence before interpretation can begin.

In other words, the derivation order used by the speaker and the hearer is *time-linear*, i.e., linear like time and in the direction of time, word by word, from beginning to end. This raises the question of whether there isn't a better way to express the grammatical relations than dominance and precedence. How could functor-argument and coordination structure at the elementary, phrasal, and clausal levels be integrated into a strictly time-linear derivation order?

The time-linear derivation order corresponds to the *left-associative* bracketing structure  $((((a\ b)\ c)\ d)\ e)$  known from logic. It is used by Left-Associative grammar (LA-grammar), which analyzes the combination of elementary, phrasal, and clausal parts of speech by computing *possible continuations* rather than possible substitutions. LA-grammar was first developed in the context of the NEWCAT'86 parser.

### 2.3.1 TIME-LINEAR NEWCAT DERIVATION OF *Julia knows John*.



The bottom-up left-associative derivation always combines a sentence start with a next word into a new sentence start, until no next word is available. The completely regular structure of the tree is used to express the derivation order, not the grammatical relations – which are coded instead into the categories shown in the tree on the right and into the categorial operations of the rules on the left.

The combination of *Julia* and *knows* is based on canceling the subject valency position  $s3'$  in the category of the next word. The combination of *Julia knows* and *John* is based on canceling the object valency position  $a'$  in the sentence start. And similarly for the third combination (as indicated by the dotted arrows).

Each rule consists of (i) a rule name, e.g., *Nom+FV* (for nominative plus finite verb), (ii) a rule package, e.g.  $\{FV+NP\}$ , containing the set of rules to be tried after a successful application of the current rule, (iii) a pattern for the sentence start, e.g.,  $[cat: NP]$ , (iv) a pattern for the next word, e.g.,  $[cat: NP' X VT]$ , and a set of operations, e.g., *cancel NP*. The patterns are based on variables like  $NP$  and  $NP'$ , where  $NP$  is restricted to noun valency fillers and  $NP'$  is restricted to noun valency positions.<sup>15</sup>

<sup>15</sup>In addition to the definition of the rules, an LA-grammar requires the definition of a set of start states, a set of final states, a set of variables and their restrictions, and a set of lexical proplets. See FoCL'99, 10.2.1, for the algebraic definition of LA-grammar.

LA-grammar differs from Categorical grammar as shown in 2.1.1 in that (i) an LA-grammar is not restricted to two rules, (ii) the LA-grammar rules have rule packages, (iii) the rule patterns indicate valency structures as lists rather than binary bracketings, (iv) there is an open number of variables with suitable restrictions and agreement conditions, and (v) the derivation has a clearly defined starting point, namely the first word. LA-grammar differs from Phrase Structure grammar in that it has a complexity hierarchy which is orthogonal to the Chomsky hierarchy (cf. TCS'92), parsing many context-free and context-sensitive languages in linear time.

The NEWCAT approach illustrated in 2.3.1 has been applied to all major functor-argument and coordination constructions of English and German. Due to its time-linear derivation structure, it proved to be well-suited for rapid upscaling. The algebraic definition of LA-grammar in CoL'89 was distilled from the Lisp code of the English parser published in NEWCAT'86.

## 2.4 Database Semantics: Hearer Mode

Despite their differences, C-grammar 2.1.1, PS-grammar 2.2.1, and NEWCAT 2.3.1 have in common that they are *sign-oriented*. Sign-oriented approaches analyze expressions of natural language as isolated grammatical structures, but leave unanswered the central question of *What to do with them?*

For human speaker-hearers the answer is obvious: expressions of natural language are used for communication. To arrive at a similar answer in linguistics, the analysis of language expressions (theory of grammar) must be embedded into a functional model of how communicating with natural language works (theory of language).<sup>16</sup> Today, the most straightforward scientific way to do this is the construction of an artificial cognitive agent, i.e., a talking robot with a real body<sup>17</sup> acting in the real world (AIJ'01).

This leads from a sign-oriented to an *agent-oriented* approach, which constrains the grammatical analysis with much needed functional requirements. The most basic of these is the computational reconstruction of turn taking,<sup>18</sup> i.e., the agent's ability to switch between the *hearer mode* and the *speaker mode*. In the hearer mode signs of natural language are interpreted as content which is stored in the agent's memory. In the speaker mode, content in the agent's memory is selectively activated and realized as corresponding surfaces in the natural language of choice.

Leaving important matters<sup>19</sup> like the external interfaces for recognition and action, the auto- and the service channel of artificial cognitive agents, and the language and the context level aside, let us turn to three theoretical issues which any computational agent-oriented approach has to address from the start:

### 2.4.1 COMPUTATIONAL DESIGN DECISIONS

1. What should be the *data structure* (abstract data type) of the content items stored in memory?
2. What should be the *algorithm* to read items of content into (hearer mode) and out of (speaker mode) the agent's memory?
3. What should be the *data base schema* according to which the content items are stored in and retrieved from memory?

<sup>16</sup>Cf. NLC'06, Sect. 2.4.

<sup>17</sup>The importance of agents with a real body (instead of virtual agents) has been emphasized by emergentism (MacWhinney 2008).

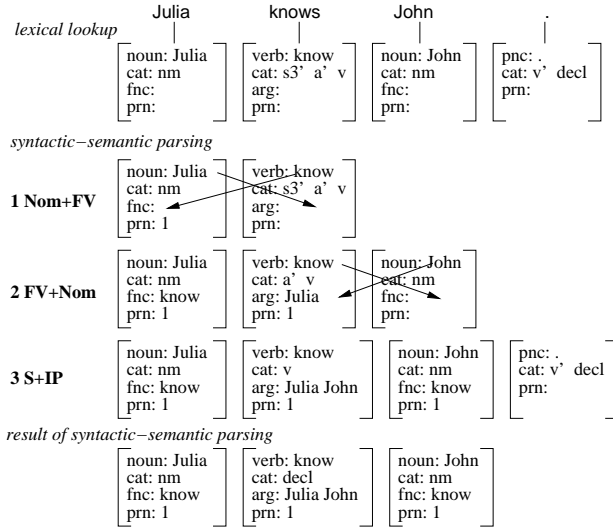
<sup>18</sup>Cf. NLC'06, Sect. 1.1, Schegloff (2007).

<sup>19</sup>For a detailed discussion see NLC', Chaps. 1 and 2.

These three must be co-designed for maximal support of the functional flow in the hearer mode, the think mode, and the speaker mode.

In Database Semantics, the data structure are proplets, i.e., flat (non-recursive) feature structures representing word forms; the algorithm is time-linear LA-grammar; and the data base schema is that of a classic network database (Elmasri and Navate 1989). As an example of the hearer mode, consider the following derivation, using the same sentence as before:

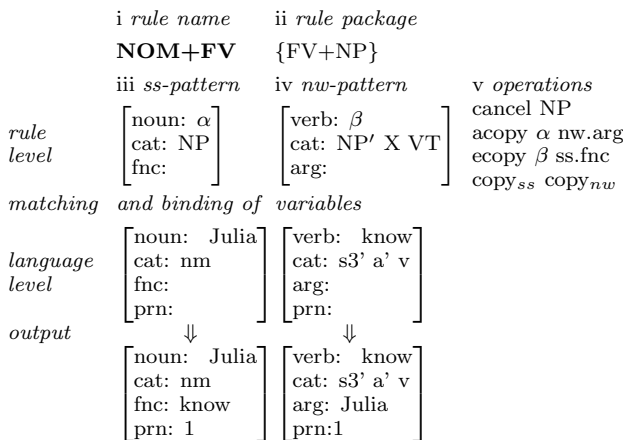
#### 2.4.2 HEARER-MODE DERIVATION IN DBS



The derivation begins with lexical lookup of the incoming unanalyzed surfaces. The functor-argument structure is coded by copying values between proplets (indicated by the diagonal arrows). The derivation is strictly time-linear, as indicated by the stair-like addition of next word proplets.<sup>20</sup> The result of the derivation is an order-free set of proplets ready to be sorted into the database. The case assigned to the arguments is indicated in the resulting verb proplet by the order of the *arg* values (here *arg*: Julia John).

Connecting the lexical proplets in accordance with the grammatical functor-argument (and, if given, coordination) structure of an input is provided by syntactic-semantic parsing, based on a variant of LA-grammar, called *LA-hear*. Consider, for example, the first rule application in 2.4.2 (explanations in *italics*):

#### 2.4.3 EXAMPLE OF LA-HEAR RULE APPLICATION



The LA-hear rule resembles the corresponding NEWCAT rule (cf. 2.3.1) in that it consists of (i) a rule

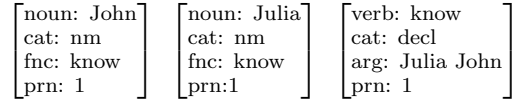
<sup>20</sup>In contradistinction to the bottom-up NEWCAT derivation 2.3.1, the DBS hearer mode derivation is not represented as a tree and must be read from top to bottom.

name, (ii) a rule package, (iii) an ss-pattern, (iv) an nw-pattern, and (v) a set of operations. Moreover, the NEWCAT and the LA-hear versions of Nom+FV share the same *cat* features and the associated operation *cancel NP*.

The two versions differ, however, in that the LA-hear version has the additional features [noun:  $\alpha$ ] and [fnc: ] in the ss-pattern, and [verb:  $\beta$ ] and [arg: ] in the nw-pattern. Vertical binding of the variable  $\alpha$  to the value *Julia* and the of  $\beta$  to *know* at the language level enables the operations *acopy  $\alpha$  nw.arg* and *ecopy  $\beta$  ss.fnc*. Consequently, the output proplet *Julia* specifies its functor as *know* and the output proplet *know* specifies one of its arguments as *Julia*.

Furthermore, while the result of the NEWCAT derivation 2.3.1 is the whole tree (which raises the same problems for storage in and retrieval from a database as the other sign-oriented representations), the result of the LA-hear derivation 2.4.2 is an *order-free* set of proplets. This set, shown below in a sequence using the alphabetical order of the core values, represents the content coded by the natural surface:

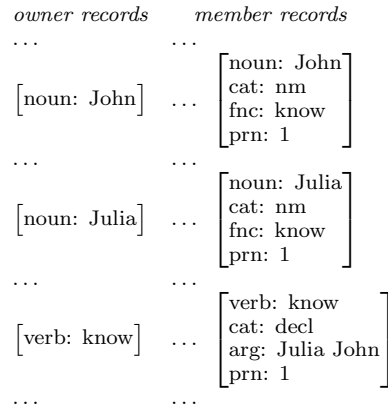
#### 2.4.4 CONTENT OF Julia knows John.



The proplets are order-free because the grammatical relations between them are coded solely by attribute-value pairs (and not in terms of dominance and precedence in a hierarchy<sup>21</sup>). Therefore, they are suitable for storage and retrieval in accordance with the methods of one's database.

Given that DBS uses the database schema of a classic network database, the proplets of 2.4.4 are stored as follows:

#### 2.4.5 STORAGE OF PROPLETS IN A WORD BANK



A network database containing proplets is called a Word Bank. The database schema consists of owner records in alphabetical vertical order, and member records stored horizontally behind their owner record. A horizontal sequence of an owner record followed by an arbitrary number of member records with the same core value is called a *token line*. The horizontal order of proplets in a token line reflects the order of their arrival, like sediment, because new incoming proplets are always stored at the end of their token line.

This method of storage is complemented by an equally simple method of retrieval: any core value provides access to all member records with the same core value by selecting the corresponding token line. Furthermore, given a core value and a *prn* (proposition number) value, the associated proplet may be

<sup>21</sup>For a detailed discussion see Hausser (2007)

retrieved by (i) going to the token line of the core value and (ii) searching through the token line for the proplet with the corresponding prn value.<sup>22</sup>

## 2.5 Database Semantics: Think Mode

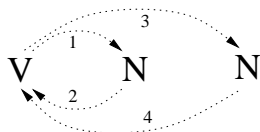
Next let us see how (i) the data structure of proplets, (ii) the algorithm of LA-grammar, and (iii) the database schema of a network database (Word Bank) serve to realize language production in the speaker mode. The task of language production is traditionally divided into *what to say* and *how to say it*.

In Database Semantics, these two aspects are viewed as (i) the *activation* of content in the agent's Word Bank and (ii) the *realization* of activated content in a natural language. The selective, autonomous activation of content is treated as a kind of thinking and based on a structural property of a Word Bank which goes beyond a classic network database.

This property is the concatenation of proplets in terms of the semantic relations of functor-argument and coordination structure. It constitutes something like a railroad system which supports a *navigation* along the semantic relations between proplets, thus selectively activating content and re-introducing a time-linear order.

For example, the *know* proplet in 2.4.4 specifies *Julia* as its first argument. Using the retrieval mechanism of the database, the navigation moves an imaginary focus point from *know* to *Julia*, then back to *know* and on to *John*, back to *know* and on to the next proposition.<sup>23</sup> This navigation may be shown schematically as follows:

### 2.5.1 NAVIGATING THROUGH A CONSTELLATION



The schema shows the constellation VNN (with V for verb and N for noun) and a navigation through the constellation, indicated by numbered arrows.

Given that any written representation of an order-free constellation of concatenated proplets must use some order, the V is written first because it specifies the connections to previous and following propositions (extrapositional relations) as well as to the nominal arguments (intrapositional relations). The first N is interpreted here as the subject and the second N as the object.

Navigating through a constellation of concatenated proplets is constrained by the following conditions:

### 2.5.2 CONDITIONS ON A NAVIGATION

1. A navigation is a *shortest* route to traverse
2. *all* proplets in the constellation such that
3. each successor proplet is *accessible* from the current proplet.

For example, after navigating from the V to the first N in a VNN constellation, it is necessary to return

to the V to get the continuation values for accessing the second N (object noun). After navigating to the second N, it is necessary to return to the V in order to get the continuation values for accessing the next proposition.

A navigation through a constellation is driven by the rules of an LA-think grammar. For example, traversal 1 in the constellation 2.5.1 is based on the following application of the rule VNs to proplets in the Word Bank 2.4.4 (explanations in *italics*):

### 2.5.3 EXAMPLE OF LA-THINK RULE APPLICATION

	i rule name VNs	ii rule package {NVs}	iii ss pattern	iv nw pattern	v operations
rule level			$\begin{bmatrix} \text{verb: } \beta \\ \text{arg: !X } \alpha \text{ Y} \\ \text{prn: k} \end{bmatrix}$	$\begin{bmatrix} \text{noun: } \alpha \\ \text{mdr: Z} \\ \text{fnc: } \beta \\ \text{prn: k} \end{bmatrix}$	output position nw
matching and binding variables					
Word Bank level	$\begin{bmatrix} \text{verb: know} \\ \text{cat: decl} \\ \text{arg: Julia John} \\ \text{prn: 1} \end{bmatrix}$	$\begin{bmatrix} \text{noun: Julia} \\ \text{cat: nm} \\ \text{fnc: know} \\ \text{prn: 1} \end{bmatrix}$		$\downarrow$	
output				$\begin{bmatrix} \text{noun: Julia} \\ \text{cat: nm} \\ \text{fnc: know} \\ \text{prn: 1} \end{bmatrix}$	

The explanations i–v show that this LA-think rule consists of the same components as the NEWCAT rules 2.3.1 and the LA-hear rule 2.4.3. The rule name VNs indicates that the rule moves the navigation from a V to an N and stays there.

By vertically binding the variable  $\beta$  in the ss-pattern to the value *know*, the variable  $\alpha$  to the value *Julia*, and the variable  $k$  to the value 1 at the Word Bank level, the retrieval mechanism of the database has the information needed for navigating to (touch, activate, traverse) the nw-proplet *Julia*. This output serves as the ss-proplet of the next LA-think rule application. Navigation step 2 in 2.5.1 returns to the V, based on the rule NVs (specified in the rule package of NVs). NVs resembles VNs except that the patterns for ss and nw are reversed.

The basic principle of navigating through the content of a Word Bank, using the semantic relations between proplets as a railroad system, is as simple as it is efficient.<sup>24</sup> Yet it has enormous descriptive potential because the rules may have more operations than the one in 2.5.3. For language production, for example, the LA-think rules may be turned into LA-speak rules by adding operations like *lex-n* [noun:  $\alpha$ ], which serve to realize specific surfaces. Other operations are used for inferences, as in the reconstruction of *modus ponens* in NLC'06, Sect. 5.3, and the answering of questions, as shown in NLC'06, Sect. 5.1.

## 2.6 Database Semantics: Speaker Mode

Having outlined the *what to say* aspect of language production, let us turn to the aspect of *how to say it*. The time-linear sequence of proplets activated by a navigation like 2.5.1 contributes the following structural properties to language production:

### 2.6.1 CONTRIBUTIONS OF NAVIGATION

1. core values
2. parts of speech

<sup>24</sup>The complexity of a navigation is linear as long as it does not branch, the number of operations per rule is below a finite upper bound, and the operations are simple.

<sup>22</sup>Please note that the structuring of a Word Bank indicated in 2.4.4 is purely conceptual and does in no way restrict the storage locations in the actual implementation of the database.

<sup>23</sup>Because the concatenated propositions in a Word Bank usually offer alternative continuations, there is the need to build an autonomous control structure for choosing between alternatives. This is a major task (cf. Hausser 2002, Hausser in print) which exceeds the limits of this paper. We concentrate here on the question of how a given navigation should be realized in a natural language.

3. semantic relations
4. traversal sequence
5. ingredients of perspective

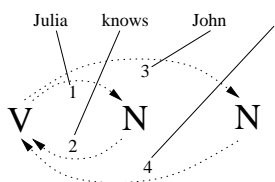
In order to map these structural details into a corresponding language-dependent surface, the LA-think rules activating the constellation must be turned into LA-speak rules by adding operations for handling the following properties of the language-dependent surface:

### 2.6.2 CONTRIBUTIONS OF GRAMMAR

1. language-dependent word order (defined on top of the traversal sequence)
2. language-dependent function word precipitation (utilizing proplet features)
3. selection of word forms (based on proplet features and rules of agreement)
4. lexical selection (driven by the core values of the proplets traversed)

These language-dependent properties of grammar are tightly interconnected, riding piggyback on the LA-think navigation rules.<sup>25</sup> For example, the activation of constellation 2.5.1 may be used to produce the English surface *Julia knows John.* as follows:

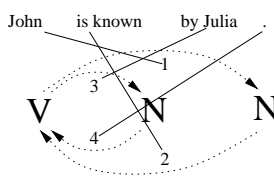
### 2.6.3 REALIZING A TRANSITIVE SURFACE



Transition 1 is used for realizing *Julia*, 2 for *knows*, 3 for *John*, and 4 for the full stop.

An alternative way to express the same content is passive.<sup>26</sup> It is based on an alternative activation order which is in concord with the conditions 2.5.2:

### 2.6.4 REALIZING A PASSIVE SURFACE



As indicated by the traversal numbers, the navigation first traverses the object and then the subject.<sup>27</sup> The realization of the passive surface in English provides a change of word order, but it also requires word forms different from the active as well as function word support.

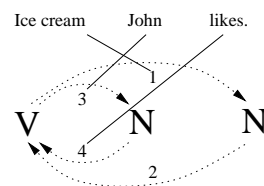
<sup>25</sup>In earlier work, LA-speak was treated as a separate component, which required a constant switching between LA-think and LA-speak. As shown by NLC'06, Chapt. 13, this turned out to be rather complicated. The current approach narrows the gap between LA-think and LA-speak by defining LA-speak as a variant of LA-think. Now the only difference between the two are additional operations for lexical realization in LA-speak, which provides for a much simpler solution.

<sup>26</sup>As pointed out by Givón (1997), one of the main functions of passive is the possibility of leaving the deep subject (agents) unspecified, as in *The car had been stolen.*

<sup>27</sup>SVO languages like English or German and SOV languages like Korean or Japanese use the navigation order of 2.6.3 as the unmarked case, whereas VOS languages like Fijian or Malagasy and OSV languages like Xavante or Warao use 2.6.4.

Other word orders in English are illustrated by *Ice cream John likes.*, which may be realized from the navigation of 2.6.4 using the realization steps 134:

### 2.6.5 REALIZING A TOPICALIZED OBJECT



Furthermore, in spoken language and in poetry the word order restrictions of a language are often handled rather freely. In Database Semantics, all such phenomena relating to word order and function word precipitation are accommodated by (i) the choice between different ways to traverse the constellation in question (for example, 2.6.3 vs. 2.6.4) and (ii) the choice of when to realize a surface during the traversal (for example, 2.6.4 vs. 2.6.5).

## 3 Treating the Phrasal and Clausal Levels

So far, the examples used have been functor-argument structures at the elementary level, without any modifiers and without an explicit treatment of function words. As an example at the phrasal level let us consider the sentence *The little black dog barked.*

### 3.1 Interpretation in the Hearer Mode

At the phrasal level, the hearer mode mapping of natural language surfaces into content involves *function word absorption* and *modification*, as illustrated by the following derivation:

#### 3.1.1 ADNOMINAL MODIFICATION (HEARER M.)

	the	little	black	dog	barked																		
lexical lookup	<table><tr><td>noun: n_1</td></tr><tr><td>fn:</td></tr><tr><td>mdr:</td></tr><tr><td>prn:</td></tr></table>	noun: n_1	fn:	mdr:	prn:	<table><tr><td>adn: little</td></tr><tr><td>mdd:</td></tr><tr><td>prn:</td></tr></table>	adn: little	mdd:	prn:	<table><tr><td>adn: black</td></tr><tr><td>mdd:</td></tr><tr><td>prn:</td></tr></table>	adn: black	mdd:	prn:	<table><tr><td>noun: dog</td></tr><tr><td>fn:</td></tr><tr><td>mdr:</td></tr><tr><td>prn:</td></tr></table>	noun: dog	fn:	mdr:	prn:	<table><tr><td>verb: bark</td></tr><tr><td>arg:</td></tr><tr><td>mdr:</td></tr><tr><td>prn:</td></tr></table>	verb: bark	arg:	mdr:	prn:
noun: n_1																							
fn:																							
mdr:																							
prn:																							
adn: little																							
mdd:																							
prn:																							
adn: black																							
mdd:																							
prn:																							
noun: dog																							
fn:																							
mdr:																							
prn:																							
verb: bark																							
arg:																							
mdr:																							
prn:																							
syntactic-semantic parsing:																							
1	<table><tr><td>noun: n_1</td></tr><tr><td>fn:</td></tr><tr><td>mdr:</td></tr><tr><td>prn: 23</td></tr></table>	noun: n_1	fn:	mdr:	prn: 23	<table><tr><td>adn: little</td></tr><tr><td>mdd:</td></tr><tr><td>prn:</td></tr></table>	adn: little	mdd:	prn:														
noun: n_1																							
fn:																							
mdr:																							
prn: 23																							
adn: little																							
mdd:																							
prn:																							
2	<table><tr><td>noun: n_1</td></tr><tr><td>fn:</td></tr><tr><td>mdr: little</td></tr><tr><td>prn: 23</td></tr></table>	noun: n_1	fn:	mdr: little	prn: 23	<table><tr><td>adn: little</td></tr><tr><td>mdd: n_1</td></tr><tr><td>prn: 23</td></tr></table>	adn: little	mdd: n_1	prn: 23	<table><tr><td>adn: black</td></tr><tr><td>mdd:</td></tr><tr><td>prn:</td></tr></table>	adn: black	mdd:	prn:										
noun: n_1																							
fn:																							
mdr: little																							
prn: 23																							
adn: little																							
mdd: n_1																							
prn: 23																							
adn: black																							
mdd:																							
prn:																							
3	<table><tr><td>noun: n_1</td></tr><tr><td>fn:</td></tr><tr><td>mdr: little black</td></tr><tr><td>prn: 23</td></tr></table>	noun: n_1	fn:	mdr: little black	prn: 23	<table><tr><td>adn: little</td></tr><tr><td>mdd: n_1</td></tr><tr><td>prn: 23</td></tr></table>	adn: little	mdd: n_1	prn: 23	<table><tr><td>adn: black</td></tr><tr><td>mdd: n_1</td></tr><tr><td>prn: 23</td></tr></table>	adn: black	mdd: n_1	prn: 23	<table><tr><td>noun: dog</td></tr><tr><td>fn:</td></tr><tr><td>mdr:</td></tr><tr><td>prn:</td></tr></table>	noun: dog	fn:	mdr:	prn:					
noun: n_1																							
fn:																							
mdr: little black																							
prn: 23																							
adn: little																							
mdd: n_1																							
prn: 23																							
adn: black																							
mdd: n_1																							
prn: 23																							
noun: dog																							
fn:																							
mdr:																							
prn:																							
4	<table><tr><td>noun: dog</td></tr><tr><td>fn:</td></tr><tr><td>mdr: little black</td></tr><tr><td>prn: 23</td></tr></table>	noun: dog	fn:	mdr: little black	prn: 23	<table><tr><td>adn: little</td></tr><tr><td>mdd: dog</td></tr><tr><td>prn: 23</td></tr></table>	adn: little	mdd: dog	prn: 23	<table><tr><td>adn: black</td></tr><tr><td>mdd: dog</td></tr><tr><td>prn: 23</td></tr></table>	adn: black	mdd: dog	prn: 23	<table><tr><td>verb: bark</td></tr><tr><td>arg:</td></tr><tr><td>mdr:</td></tr><tr><td>prn:</td></tr></table>	verb: bark	arg:	mdr:	prn:					
noun: dog																							
fn:																							
mdr: little black																							
prn: 23																							
adn: little																							
mdd: dog																							
prn: 23																							
adn: black																							
mdd: dog																							
prn: 23																							
verb: bark																							
arg:																							
mdr:																							
prn:																							
result of syntactic-semantic parsing:																							
	<table><tr><td>noun: dog</td></tr><tr><td>fn: bark</td></tr><tr><td>mdr: little black</td></tr><tr><td>prn: 23</td></tr></table>	noun: dog	fn: bark	mdr: little black	prn: 23	<table><tr><td>adn: little</td></tr><tr><td>mdd: dog</td></tr><tr><td>prn: 23</td></tr></table>	adn: little	mdd: dog	prn: 23	<table><tr><td>adn: black</td></tr><tr><td>mdd: dog</td></tr><tr><td>prn: 23</td></tr></table>	adn: black	mdd: dog	prn: 23	<table><tr><td>verb: bark</td></tr><tr><td>arg: dog</td></tr><tr><td>mdr:</td></tr><tr><td>prn: 23</td></tr></table>	verb: bark	arg: dog	mdr:	prn: 23					
noun: dog																							
fn: bark																							
mdr: little black																							
prn: 23																							
adn: little																							
mdd: dog																							
prn: 23																							
adn: black																							
mdd: dog																							
prn: 23																							
verb: bark																							
arg: dog																							
mdr:																							
prn: 23																							

The determiner *the* is treated lexically as a function word with the core value *n\_1*, called a substitution value. In line 1 of syntactic-semantic parsing, the value *little* is copied into the *mdd* (modifier) slot of *the* and the value *n\_1* is copied into the *mdd* (modified) slot of *little*. Similarly in line 2, in which *black* and *n\_1* are cross-copied, as indicated by the diagonal arrows.

In line 3, all instances of *n\_1* are simultaneously replaced by the core value of the *dog* proplet, which is then discarded (as indicated by the gap in line 4). In other words, the determiner and the noun proplets are being fused by absorbing the core value of the

noun into the determiner proplet,<sup>28</sup> and providing the adnominal modifiers with suitable mdd values.

The result of the hearer mode derivation is an order-free set of proplets, shown below using the order of a VNAA constellation (cf. 3.2.1 below):

### 3.1.2 CONTENT OF The little black dog barked

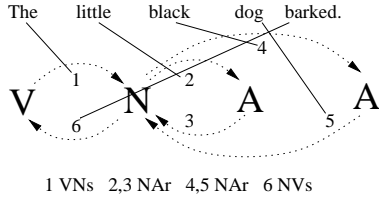
$\begin{bmatrix} \text{verb: bark} \\ \text{cat: decl} \\ \text{arg: dog} \\ \text{prn: 23} \end{bmatrix}$	$\begin{bmatrix} \text{noun: dog} \\ \text{cat: def sg} \\ \text{fnc: bark} \\ \text{mdr: little black} \\ \text{prn: 23} \end{bmatrix}$	$\begin{bmatrix} \text{adj: little} \\ \text{cat: adn} \\ \text{mdd: dog} \\ \text{prn: 23} \end{bmatrix}$	$\begin{bmatrix} \text{adj: black} \\ \text{cat: adn} \\ \text{mdd: dog} \\ \text{prn: 23} \end{bmatrix}$
--	--	--	---

In DBS, the phrasal nature of the noun **the little black dog** is not expressed by a single node dominating other nodes, as it would be in PS-grammar (cf. 2.2.1) or C-grammar (cf. 2.1.1). Instead of a hierarchy, it is treated as a constellation of bidirectionally concatenated proplets representing the modified and its modifiers. The contribution of the function word **the** appears as the cat<sup>29</sup> value def in the *dog* proplet.

## 3.2 Production in the Speaker Mode

Producing the English sentence **the little black dog barked.** from a constellation requires (i) realization of the determiner absorbed during interpretation and (ii) the correct positioning of the elementary adnominal modifiers between the determiner and their modified. This is shown by the following navigation with associated realization:

### 3.2.1 ADNOMINAL MODIFICATION (SPEAKER M.)



In this VNAA constellation, the V precedes the N for the reasons explained in connection with 2.5.1. The N precedes the As because they are accessible only from the N (cf. transitions 2,3 and 4,5).

The English surfaces are realized from the goal proplet of a transition by means of lex functions such as lex-d (for determiners), lex-nn (for elementary common nouns) and lex-n (for determiner noun sequences without intervening adnominal adjectives), as defined below for the regular cases:

### 3.2.2 LEXICALIZATION FUNCTIONS

#### 1. lex-d

If one of the following patterns matches an N proplet, then lex-d applied to this proplet produces the associated surface:

pattern	surface	pattern	surface
$\begin{bmatrix} \text{noun: } \alpha \\ \text{sem: indef sg} \end{bmatrix}$	a(n)	$\begin{bmatrix} \text{noun: } \alpha \\ \text{cat: snp} \\ \text{sem: pl exh} \end{bmatrix}$	every
$\begin{bmatrix} \text{noun: } \alpha \\ \text{sem: sel} \end{bmatrix}$	some	$\begin{bmatrix} \text{noun: } \alpha \\ \text{cat: pnp} \\ \text{sem: pl exh} \end{bmatrix}$	all
$\begin{bmatrix} \text{noun: } \alpha \\ \text{sem: def X} \end{bmatrix}$	the		

<sup>28</sup>The relevant content of a function word may also be absorbed into a content word. An example are punctuation signs, which are absorbed into the verb proplet, as illustrated in 2.4.2.

<sup>29</sup>In 3.1.1, the cat features as well as the sem, nc, pc, and idy features (cf. 1.5.1) are omitted for simplicity. For a complete definition of the LA-hearer grammar for elementary and phrasal intrapropositional functor-argument structure and extrapropositional coordination see NLC'06, Chapt. 13.

#### 2. lex-nn

If  $\begin{bmatrix} \text{noun: } \alpha \\ \text{cat: snp} \end{bmatrix}$  matches an N proplet, then lex-nn[noun:  $\alpha$ ] =  $\alpha$ .

If  $\begin{bmatrix} \text{noun: } \alpha \\ \text{cat: pnp} \end{bmatrix}$  matches an N proplet, then lex-nn [noun:  $\alpha$ ] =  $\alpha + s$ .

#### 3. lex-n

If one of the following patterns matches an N proplet, then lex-n applied to this proplet produces the associated surface:

pattern	surface	pattern	surface
$\begin{bmatrix} \text{noun: } \alpha \\ \text{cat: snp} \\ \text{sem: indef sg} \end{bmatrix}$	a(n) $\alpha$	$\begin{bmatrix} \text{noun: } \alpha \\ \text{cat: snp} \\ \text{sem: pl exh} \end{bmatrix}$	every $\alpha$
$\begin{bmatrix} \text{noun: } \alpha \\ \text{cat: snp} \\ \text{sem: sel} \end{bmatrix}$	some $\alpha$	$\begin{bmatrix} \text{noun: } \alpha \\ \text{cat: pnp} \\ \text{sem: sel} \end{bmatrix}$	some $\alpha + s$
$\begin{bmatrix} \text{noun: } \alpha \\ \text{cat: pnp} \\ \text{sem: pl exh} \end{bmatrix}$	all $\alpha + s$	$\begin{bmatrix} \text{noun: } \alpha \\ \text{cat: snp} \\ \text{sem: def X} \end{bmatrix}$	the $\alpha$
$\begin{bmatrix} \text{noun: } \alpha \\ \text{cat: pnp} \\ \text{sem: def X} \end{bmatrix}$	the $\alpha + s$		

Other lex functions are lex-v for the realization of a finite verb form, lex-adn for adnominal adjectives, and lex-p for the punctuation signs.

The lex functions are called by the rules of an LA-speak grammar. Consider the following definition, which replaces the definition of LA-speak.2 in NLC'06, Sect. 14.2:

### 3.2.3 FORMAL DEFINITION OF LA-SPEAK.E2

$\mathbf{ST}_S = \text{def } \{ ([\text{verb: } \alpha] \{ 1 \text{ VN} \}) \}$

$\mathbf{VN}s \quad \{ 2 \text{ NV}s, 3 \text{ NAr} \}$

$\begin{bmatrix} \text{verb: } \beta \\ \text{arg: !X } \alpha \text{ Y} \\ \text{prn: i} \end{bmatrix} \quad \begin{bmatrix} \text{noun: } \alpha \\ \text{mdr: Z} \\ \text{fnc: } \beta \\ \text{prn: i} \end{bmatrix} \quad \begin{array}{l} \text{if adn } \notin Z, \text{ lex-n [noun: } \alpha] \\ \text{if adn } \in Z, \text{ lex-d [noun: } \alpha] \\ \text{(where adn is an elementary} \\ \text{adnominal)} \end{array}$

$\mathbf{NV}s \quad \{ 4 \text{ VN}s, 5 \text{ VV}s \}$

$\begin{bmatrix} \text{noun: } \alpha \\ \text{fnc: } \beta \\ \text{prn: i} \end{bmatrix} \quad \begin{bmatrix} \text{verb: } \beta \\ \text{arg: X! } \alpha \text{ Y} \\ \text{cat: VT} \\ \text{prn: i} \end{bmatrix} \quad \begin{array}{l} \text{mark } \alpha \text{ in } \beta\text{-arg} \\ \text{if X = NIL, lex-fv [verb: } \beta] \\ \text{if Y = NIL, lex-p [verb: } \beta] \end{array}$

$\mathbf{NAr} \quad \{ 6 \text{ NAr}, 7 \text{ NV}s \}$

$\begin{bmatrix} \text{noun: } \alpha \\ \text{mdr: !X } \beta \text{ Y} \\ \text{prn: i} \end{bmatrix} \quad \begin{bmatrix} \text{adj: } \beta \\ \text{mdd: } \alpha \\ \text{prn: i} \end{bmatrix} \quad \begin{array}{l} \text{mark } \beta \text{ in } \alpha\text{-mdr} \\ \text{lex-a [adj: } \beta] \\ \text{if adn } \notin Y, \text{ lex-nn [noun: } \alpha] \end{array}$

$\mathbf{VV}s \quad \{ 8 \text{ VN}s \}$

$\begin{bmatrix} \text{verb: } \alpha \\ \text{nc: j } \beta \\ \text{prn: i} \end{bmatrix} \quad \begin{bmatrix} \text{verb: } \beta \\ \text{pc: i } \alpha \\ \text{prn: j} \end{bmatrix}$

$\mathbf{ST}_F = \text{def } \{ ( [\text{verb: } \beta] [\text{arg: X!}] \text{rPNVs}) \}$

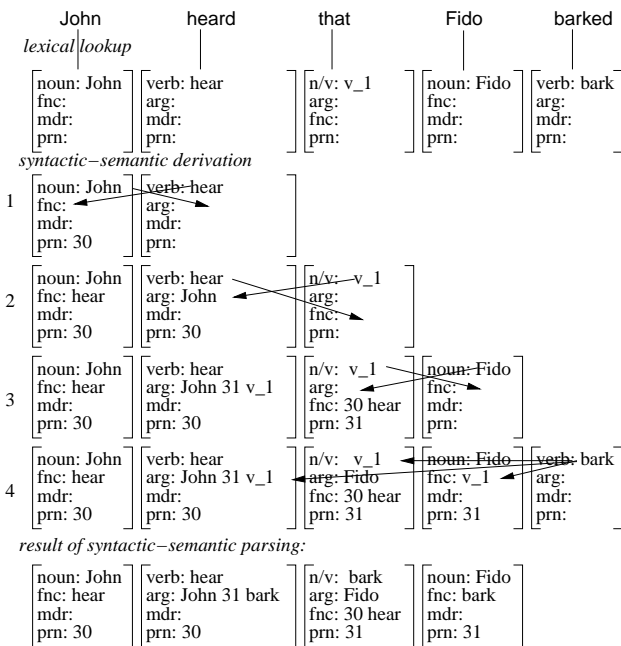
The rules of this grammar have two kinds of suffixes, s (for staying at the second proplet) and r (for returning to the first proplet). This way of specifying the output position in the rule name makes the use of the corresponding operations redundant (compare 2.5.3).

As indicated at the bottom of 3.2.1, rule VN<sub>s</sub> executes transition 1, thereby realizing the determiner using lex-d. Rule NAr executes transition 2, realizes the adnominal **little** with lex-a, and returns to the N (transition 3). Then NAr applies again (cf. rule package), executes transition 4, realizes **black**, and returns to the N (transition 5), thereby realizing **dog** with lex-nn (because adn  $\notin$  Y). Finally, NV<sub>s</sub> executes transition 6 and realizes **barked.** using lex-v and lex-p.

### 3.3 Interpretation at the Clausal Level

It remains to show the handling of clausal constructions. As an example, consider the following derivation of John heard that Fido barked.

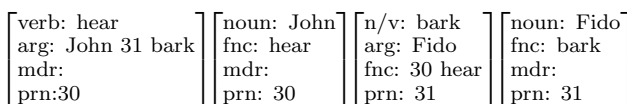
#### 3.3.1 OBJECT CLAUSE (HEARER MODE)



In the result, the proplets of the main clause and of the subordinate clause are distinguished by different prn values, 30 and 31. The connection between the main verb and its sentential object is coded by the arg value 31 bark of *hear*. The inverse connection is coded by the fnc value 30 hear of *bark*.

The result of the hearer mode derivation is an order-free set of proplets, shown below in the order of a VNV<sup>n</sup>N constellation (cf. 3.4.1 below):

#### 3.3.2 CONTENT OF John heard that Fido barked.

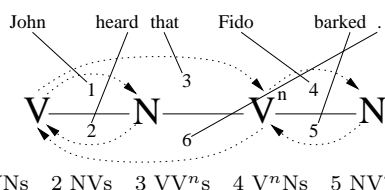


The double function of the *bark* proplet as the functor of the subordinate clause and as the argument of the main clause is indicated by its core attribute n/v (for nominal argument in the form of a subordinate verb).

### 3.4 Production at the Clausal Level

Turning to production, consider a schematic characterization of the proplet sequence 3.1.2 as a VNV<sup>n</sup>N constellation with an associated realization; V<sup>n</sup> represents the proplet with the n/v core attribute.

#### 3.4.1 OBJECT CLAUSE (SPEAKER MODE)



The LA-speak.3 grammar for realizing English functor-argument constructions with clausal arguments and clausal modifiers (presented systematically in NLC'06, Chapt. 7) has a total of 15 rules (including the 4 rules of LA-speak.2 defined in 3.2.3).

### 4 Conclusion

Is it really necessary to analyze the grammatical relations as a hierarchy, such that a functor (verb) must dominate its arguments (nouns), for example? Apart from well-known problems with discontinuous elements and the handling of coordination, the assumption of a hierarchical structure stands in the way of a time-linear interpretation in the hearer mode, a time-linear navigation in the think mode, and a time-linear production in the speaker mode.

For a computational model of these operations, Database Semantics treats the semantic relations of functor-argument and coordination structures instead as constellations of order-free proplets connected by attribute-value pairs. It has been shown here that this formal method easily accommodates a traditional approach to grammar based on nouns, verbs, and adjectives at the elementary, phrasal, and clausal level.

### 5 References

- AIJ'01 = Hausser, R. (2001) "Database Semantics for natural language," *Artificial Intelligence*, 130.1:27-74
- Ajdukiewicz, K. (1935) "Die syntaktische Konnexität," *Studia Philosophica*, 1:1-27
- Bar Hillel, J. (1953) "A quasi-arithmetical notation for syntactic description," *Language*, 29.1:47-63
- Benner, M. L. (2008) *Towson University online Writing Support*, <http://www.towson.edu/ows/>
- Chomsky, N. (1957) *Syntactic Structures*, The Hague: Mouton
- Chomsky, N. (1965) *Aspects of the Theory of Syntax*, Cambridge, Mass.: MIT Press
- CoL'89 = Hausser, R. (1989) *Computation of Language*, Symbolic Computation: Artificial Intelligence, Berlin Heidelberg New York: Springer
- Elmasri, R., and S.B. Navathe (1989) *Fundamentals of Database Systems*. Redwood City, CA: Benjamin-Cummings
- FoCL'99 = Hausser, R. (1999/2001) *Foundations of Computational Linguistics, Human-Computer Communication in Natural Language*, 2nd ed., Berlin Heidelberg New York: Springer
- Givón, T. (1997) *Grammatical Relations: A Functional Perspective*. Amsterdam: John Benjamins
- Hausser, R. (2002) "Autonomous Control Structure for Artificial Cognitive Agents," in H. Kangassalo et al. (eds) *Information Modeling and Knowledge Bases XIII*, Amsterdam: IOS Press Ohmsha
- Hausser, R. (2007) "Comparing the Use of Feature Structures in Nativism and in Database Semantics," H. Jaakkola et al. (eds) *Information Modelling and Knowledge Bases XIX*, Amsterdam: IOS Press Ohmsha
- Leśniewski, S. (1929) "Grundzüge eines neuen Systems der Grundlagen der Mathematik," Warsaw: *Fundamenta Mathematicae* 14:1-81
- MacWhinney, B. (2008) "How Mental Models Encode Embodied Linguistic Perspective," in L. Klatzky et al. (eds.) *Embodiment, Ego-Space, and Action*, New York: Psychology Press
- NEWCAT'86 = Hausser, R. (1986) *NEWCAT: Parsing Natural Language Using Left-Associative Grammar*, Lecture Notes in Computer Science 231, Berlin Heidelberg New York: Springer
- NLC'06 = Hausser, R. (2006) *A Computational Model of Natural Language Communication*, Berlin Heidelberg New York: Springer
- Post, E. (1936) "Finite Combinatory Processes — Formulation I," *Journal of Symbolic Logic*, 1:103-105
- Schegloff, E. (2007) *Sequence Organization in Interaction*, New York: Cambridge Univ. Press
- TCS'92 = Hausser, R. (1992) "Complexity in Left-Associative Grammar," *Theoretical Computer Science*, 106.2:283-308



# CONTRIBUTED PAPERS



## Business Process Integration: Method and Analysis

Evan D. Morrison

Alex Menzies

George Koliadis

Aditya K. Ghose

Decision Systems Lab

School of Computer Science and Software Engineering

University of Wollongong,

Wollongong NSW 2522, Australia,

Email: {edm92, am57, gk56, aditya}@uow.edu.au

### Abstract

*In the study of business management, process integration has become an interesting area of research that affects analysts studying and working on existing system plans. Process integration aims to investigate relationships across a business compendium to produce classifications and merge similar activities into a standardized system. Integration is the process of merging elements from two similar antecedent processes to create a single process that can be used to replace the original processes. This paper proposes a practical method for process integration and provides a theoretical framework and metrics for business process integration assessment. In the provision of metrics that take into account similarity of activities within processes we are able to offer solutions that provide minimal change reducing change costs, and minimizing change impact risks.*

**Keywords:** BPMN, Process Integration, Similarity Matching, SPNets

### 1 Introduction

The problem of *business process integration* is ubiquitous in a wide variety of domains. Consider, for example, a back-office financial service provider that serves a range of different pension funds. The service provider must support processes for clients (individuals) to be added to the system, new employers to be added to the system, consolidation of pension accounts in various funds into a single account etc. The provider must support different versions of these processes for the different pension funds that it serves, since each fund requires its own process to be followed in each instance. The provider notes that a given process (say that for changing client addresses) varies minimally from fund to fund, and seeks to ‘rationalize’ these variants to obtain a

single process. The problem is one of business process integration, where the fund-specific variants are combined into a single process that achieves the goals of *all* of the original processes.

Consider another example where a smaller insurance company is acquired by a larger insurance company. There is a need for the resulting entity to support a single claims handling process, which requires that the claims handling processes of both the acquiring and acquired companies be integrated into a single consolidated process that achieves the goals/objectives of both prior processes.

In both examples, there would be an implicit requirement that the consolidated process be as ‘close’ or as ‘similar’ as possible to the original processes in order to minimize disruption and to protect investments in existing process infrastructure. The business process integration problem can thus be viewed as the problem of identifying a single process that:

1. Achieves all of the goals/objectives of a set of prior processes while
2. Minimizing the extent of change required to the original processes.

The first requirement involves identifying a new goal that the integrated process must achieve, which can be a simple matter of taking the conjunction of the goals of the prior processes (we do not address more complex questions concerning goal merging when the goals are inconsistent, although this is an important problem). The second requirement involves assessing, and ideally measuring, the extent of change affected by a process variant relative to the original process.

We present a novel approach to business process integration that relies on a set of *process proximity metrics*. We assume that processes are represented in the industry-standard BPMN notation. BPMN provides little support for representing the semantics of the processes being modeled (beyond the nomenclature of the tasks involved, and the conditions that label decision gateways). We first describe an approach to supporting *lightweight semantic annotation* of BPMN models by analysts, bearing in mind that insisting on the use of ‘heavier’ formal methods for annotation, or the translation of BPMN models into formal semantic domains (on which there is no consensus within the community), would find little acceptance in practice. Based on this

scheme, we define a uniform graph-based encoding of semantically annotated BPMN models, called *semantic process nets* (or SPNets) (originally introduced in (Ghose & Koliadis 2007)). We then describe a class of process proximity metrics, and show how these can form the basis for an effective business process integration framework. We note that we do not address the problem of data integration (which has been the subject of considerable earlier research), but focus only on behavior integration. Our work improves on several earlier frameworks for process integration that have been discussed in the next section.

## 2 Background

The principal purpose of any model is to “identify the structural features that have the greatest implications for policy, and thus are worthy of further pursuit” (Fiddaman 1997). Using business process modeling as a means to express the operation of an organizational system based on a combination of artifacts and knowledge extracted from domain experts provides a level of formalism. Maintenance of the formal system can be viewed as problem to be solved within the notation.

### 2.1 Business Process Modelling

The Business Process Modeling Notation (BPMN) has received strong industry interest and support (White 2006), is highly mature (Becker et al. 2005), but has been found to have some limitations relating to the representation of process state and other ambiguities (Becker et al. 2005). Business processes are represented in BPMN using **flow nodes**: *events*, *activities*, and *decisions*; **connectors**: *control flow links*, and *message flow links*; and **swimlanes**: *pools*, and *lanes* within pools.

We model the quality aspects of a BPMN model using an algebraic scheme developed within the constraint modeling literature (Bistarelli 2001), which defines quality scales as a 5-tuple  $\langle \mathbf{A}, \oplus, \otimes, \mathbf{0}, \mathbf{1} \rangle$ . Under this scheme,  $\mathbf{A}$  is a set consisting of numeric, boolean or symbolic preference values,  $\oplus$  and  $\otimes$  are two commutative and associative operators closed in  $\mathbf{A}$ ,  $\mathbf{0}$  is the least preferred value in  $\mathbf{A}$ , and  $\mathbf{1}$  is the most preferred value in  $\mathbf{A}$ . In addition:  $\oplus$  is an idempotent *comparison* operator, which has an absorbing element of  $\mathbf{1}$ , and unit element of  $\mathbf{0}$ ; and,  $\otimes$  is a *combination* operator, which is usually decreasing, distributes over comparison, has an absorbing element of  $\mathbf{0}$ , and a unit element of  $\mathbf{1}$ .

### 2.2 Business Process Integration

In the following sub-sections, we introduce the state-of-the-art in business process matching and integration techniques, and describe how we address some of their drawbacks.

#### Matching

Process matching is the process of clustering and relating similar activities. These clusters can be derived using various methods each with strengths and weaknesses that can leverage the knowledge stored in a process.

Clustering techniques classify objects (such as business process models) into partitions so that the data in each subset share common traits. A number of clustering methods

and functions are outlined in (Huang & Ng 1999) using large set based k-mean algorithms. During the clustering phase each element is massaged into a group of related elements. In cases where data can not be disseminated using large data set averaging methods, the classification of objects in a particular domain can be completed by separating objects into classes based on their attributes, and giving criteria for determining whether a particular object in the domain is in a particular class or not. This is done in bi-clustering (Busygina et al. 2008) where a set of samples are simultaneously partitioned into subsets in a way that they have a high relevance to each other, k-mean clustering can be used with other methods to create of bi-clustered groups.

The problem of using these techniques within an organizational domain is the complexity associated with implementations. Most implementations of data clustering can be seen in large scale projects such as gene mapping and search engine crawling.

Smaller steps can be taken to reduce the complexity of large scale data classification requirements with the use of naming conventions. Activity names should carry clear and concise meanings. Each data set name will provide a significant meaning to the observer. During design analysts define models using meaningful naming conventions to provide clarity in some context. (Kementsietsidis et al. 2003) Kementsietsidis investigates methods for the logical design of data values to promote integration from heterogeneous data sources using data mapping tables. The tables maintain correspondences between, for example, business processes within a process repository. Thus, queries may result in alternate names, retaining knowledge in a particular domain.

An example of classification completed by system users can be seen implemented in the collaborative database schema integration tool SISIBIS (Beynon-Davies et al. 1997) where during the creation of enterprise data schemas, analysts and system users were asked to tag various elements with the semantic meanings (with respect to themselves) and how various data was designed using contextual descriptions.

The use of matching techniques to connect elements from incoming processes helps reveal contextual similarity. Contextual similarity is required in process integration, acting as a mapping function that shows direct similarities between two processes. In (van Dongen et al. 2008) Dongen, et al. presents a vector based proximity metric for contextual similarity. These metrics are found by beginning with cluster based semantic similarities across documents (using causal and contextual footprints and combining with semantic scoring), where an organization wide vector of artifacts are contrasted against one another using union operators. This approach allows an analyst to rank semantic equivalences between two or more processes. The adjunct system described in (Mendling & Simon 2006) (Mendling) shows structural integration methods using Event-Driven Process chains against a number of SAP process models. In this work a structure merge operator is defined for use on SAP models that can be used once a semantic similarity between functions has been defined. Mendling also shows a reduction method that can be achieved by eliminating redundant process pathways while keeping EPC based structural integrity. In the preceding research Event-Driven process chains are used to verify structural ‘soundness’ or non-recoverable errors (van Dongen et al. 2005) as well as reducing complexity within the structure. The problems asso-

ciated with relying on a union combination of various process information (van Dongen et al. 2008) is in considering similarity of functions (defined using synonyms and words, or similarly of footprints) ignoring the frequency of the metrics (Lewis 1998). We address these problems by considering differences and adding together algebraic distances.

### Integration

Process integration aims to investigate relationships across a business compendium to produce classifications and merge activities into a standardized system. This involves both matching and merging methods. The process of integrating various activities relies heavily on matching criteria. Once objects are considered close enough to integrate with one another, and if each object is not equal to the other, then the merging process will begin.

Integration is the process of merging elements from two similar antecedent processes to create a single process that can be used to replace the original processes. Integration is broken into two parts, aggregation and regression. Aggregation is the process of combining data elements after detecting common elements or common relations (Wang & Miller 2005). This is done in its simplest form by combining common elements from two antecedent processes.

Hinke (Hinke 1988) shows a further depth to aggregation by comparing general cardinality aggregations and inference aggregation in which predictions of inference can be made from data. Here not only are similar activities from antecedent processes joined in an integrated output, there is also a case where if an antecedent activity has a relation to an activity that does not have a direct role in a process but acts as a constraint on a future activity within the process, then the activity is included during integration as an inference activity. For example, consider a process where there is an activity of ‘stamping letters in a mail room’. During the integration of two processes that describe ‘*sending a letter to a customer*’, we must consider ‘*stamping a letter*’ as a constraint to be satisfied before ‘*mailing the letter*’. This activity should be included in the integrated output process even if it is not explicitly defined in one of the antecedent processes.

*Regression* is a stage within an integration system that involves reduction of the possible resulting process solution space while maintaining consistency. As a model of a process is aggregated a number of possible solutions can be generated. It is during regression that duplicate and structurally unnecessary data and information is removed to form explicit processes. These processes can then be analysed and a potential candidate implementation process can be chosen. In (Morimune & Hoshino 2008) Morimune offers a number of regression testing methods that can be used to for the creation of these candidate processes, using homogeneous constraints. Regression in the use of process integration is useful for selecting optimal solutions.

Research into the area of business process space has resulted in interesting work, where many of the technical aspects of integration have been addressed. In (Grossmann et al. 2004), a method for business process integration is presented, which relies on the introduction of detailed and explicit process states, inter-process dependencies, and synchronizations as integration criteria. In comparison, the work in this paper presents a goal and proximity-directed criterion (relying on minimal analyst intervention) allowing analysts to explore candidate integrations that maintain structural and semantic similarity to their antecedents. In

(Mendling & Simon 2006), a (database) view integration-inspired business process integration method, achieved via a view-merge operator, identity/ordering relations, and restructuring (or simplification rules) is presented. In comparison, we outline criteria that help establish identity relations, and minimize structural and (some) semantic differences during integration.

In the following sections we provide a conceptual framework that can be relatively easily implemented in decision-support tools to determine degree of similarity of process model integration options. A key challenge with BPMN is that it provides relatively little by way semantics of the processes being modeled. Another challenge is that there is no consensus on how the semantics of BPMN might be defined, although several competing formalisms have been proposed. Since integration clearly requires more information than is available in a pure BPMN process model, we propose a lightweight, analyst-mediated approach to semantic annotation of BPMN models, in particular, the annotation of activities with effects. Model checking is an alternative approach, but it requires mapping BPMN process models to state models, which is problematic and ill-defined. We encode BPMN process models into semantically-annotated digraphs called Semantic Process Networks (or SPNets). We then define a class of proximity relations that permit us to compare alternative modifications of process models in terms of how much they deviate from the original process model.

### 3 Semantic Process Nets (SPNets)

Semantic Process Nets (SPNets) (Ghose & Koliadis 2007) provide us with a uniform structural and semantic encoding of a BPMN model to which we will be developing our theory for business process integration.

**Definition 1** A *Semantic Process Network (SPNet)* is a graph  $\langle V, E, s, t, l_V, l_E \rangle$  such that:  $V$  is a set of nodes;  $E$  a set of edges;  $s, t : E \rightarrow V$  are source and target node mappings;  $l_V : V \rightarrow \Omega_V$  maps nodes to node labels; and,  $l_E : E \rightarrow \Omega_E$  maps edges to edge labels. Each label in  $\Omega_V$  and  $\Omega_E$  is of the form  $\langle id, type, value \rangle$ .

We note that a unique SPNet exists for each model in BPMN. This can be determined objectively through transformation. Each event, activity or gateway in a BPMN model maps to a node, with the *type* element of the label indicating whether the node was obtained from an event, activity or gateway in the BPMN model. Actors also map as nodes, with the *value* label referring to the name of the role associated with the pool and lane of the actor. The *type* element of an edge label can be either *control*, *message*, *assignment*, *immediate effect*, or *cumulative effect* depending on whether the edge represents a control flow, message flow, task assignment, immediate effect, or cumulative effect descriptor. The *value* element of edge labels are: guard conditions (for control edges); message descriptors (for message edges); actor names (for assignment edges); post conditions (for immediate effect edges); or, context descriptors (for cumulative effect edges). Note,  $s(e) = t(e)$  for an immediate effect, or cumulative effect edge  $e \in E$ .

The *value* elements for immediate effect, and cumulative effect edges are triples of the form

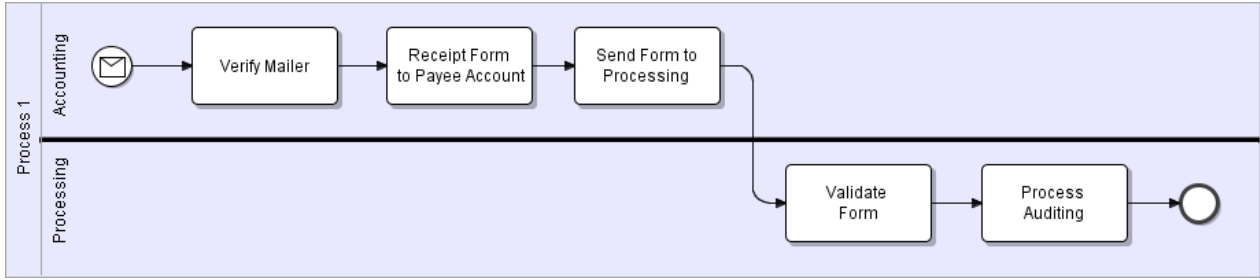


Figure 1. Example Process 1

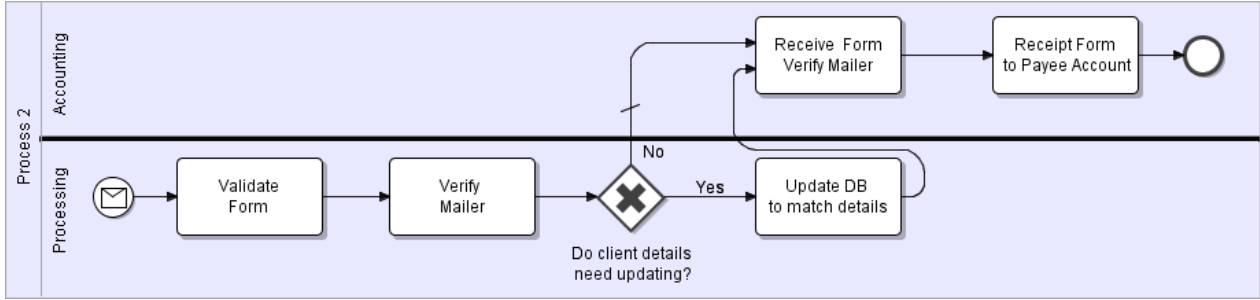


Figure 2. Example Process 2

$\langle id, function, quality \rangle$ . The *id* element of an immediate effect edge corresponds to the source node *id* label element. The *id* element of a cumulative effect edge is a scenario identifier (a vector) where each element is either: a node identifier; or, a set whose elements are (recursively) scenario identifiers. A scenario identifier describes the precise path that would have to be taken through the process model to achieve the cumulative effect in question.

The *function* element of an immediate effect, or cumulative effect edge label is a set of assertions, whereas the *quality* element is a vector of QoS evaluations. The *function* and *quality* elements of an immediate effect annotation edge label can be viewed as a context-independent specification of its functional and non-functional effects. These must be accumulated over an entire process to be able to specify, at the end of each activity, the contextual *function* and *quality* elements of cumulative effect annotation labels. These labels indicate the functional and non-functional effects that a process would have achieved had it executed up to that point.

### 3.1 Accumulating Functional Effects

We define a process for *pair-wise effect accumulation*, which, given an ordered pair of tasks with effect annotations, determines the cumulative effect after both tasks have been executed in contiguous sequence. The procedure serves as a methodology for analysts to follow if only informal annotations are available. In the case of formal annotations, we assume effects have been represented in Conjunctive Normal Form (CNF) where each clause is also a *prime implicate*, thus providing a non-redundant canonical form. Cumulative effect annotation involves a left-to-right pass through a participant lane. Activities which are not connected to any preceding activity via a control flow link

are annotated with the cumulative effect  $\{e\}$  where *e* is the immediate effect of the task in question. The process of obtaining cumulative effect annotations from a BPMN model annotated with immediate effects can be automated in the instance of formal or controlled natural language annotations. We note that this approach to obtaining functional effect descriptions comes with no guarantee of completeness. In other words, the quality of the descriptions that we obtain is a function of the quality of immediate effects and goals specified by analysts. Our experience suggests that the approach is nonetheless useful in providing an approximately adequate basis for change management.

Let  $\langle t_i, t_j \rangle$  be an ordered pair of tasks connected via a sequence flow such that  $t_i$  precedes  $t_j$ , let  $e_i$  be an effect scenario associated with  $t_i$  and  $e_j$  be the immediate effect annotation associated with  $t_j$ .

Let  $e_i = \{c_{i1}, c_{i2}, \dots, c_{im}\}$  and  $e_j = \{c_{j1}, c_{j2}, \dots, c_{jn}\}$  (we can view CNF sentences as sets of clauses, without loss of generality). If  $e_i \cup e_j$  is consistent, then the resulting cumulative effect, denoted by  $acc(e_i, e_j)$ , is  $\{e_i \cup e_j\}$ . Else,  $acc(e_i, e_j) = \{e_j \cup e \mid (e \subseteq e_i) \text{ and } (e \cup e_j \text{ is consistent}) \text{ and } (\text{there does not exist } e' [(e' \cup e_j \text{ is consistent}) \text{ and } (e \subset e')])\}$ .

The process continues without modification over splits. Joins require special consideration. In the following, we describe the procedure to be followed in the case of 2-joins only, for brevity. The procedure generalizes in a straightforward manner for *n*-way joins.

In the following, let  $t_1$  and  $t_2$  be the two tasks immediately preceding a join. Let their cumulative effect annotations be  $E_1 = \{es_{11}, es_{12}, \dots, es_{1m}\}$  and  $E_2 = \{es_{21}, es_{22}, \dots, es_{2n}\}$  respectively (where  $es_{ts}$  denotes an effect scenario, subscript *s* within the cumulative effect of some task, subscript *t*). Let *e* be the immediate effect an-

notation, and  $E$  the cumulative effect annotation of a task  $t$  immediately following the join.

For an *AND-join*, we define  $E = \{a_i \cup a_j \mid a_i \in \text{acc}(es_{1i}, e) \text{ and } a_j \in \text{acc}(es_{2j}, e) \text{ and } es_{1i} \in E_1 \text{ and } es_{2j} \in E_2 \text{ and } \{es_{1i}, es_{2j}\} \text{ are compatible}\}$ . A pair of effect scenarios are compatible if and only if their identifiers (representing the path and decisions taken during construction of the scenario) are consistent (the outcomes of their decisions match). Note that we do not consider the possibility of a pair of effect scenarios  $es_{1i}$  and  $es_{2j}$  being inconsistent, since this would only happen in the case of intrinsically and obviously erroneously constructed process models. The result of effect accumulation in the setting described here is denoted by  $\text{ANDacc}(E_1, E_2, e)$ .

For an *XOR-join* (denoted by  $\text{XORacc}(E_1, E_2, e)$ ), we define  $E = \{a_i \mid a_i \in \text{acc}(es_i, e) \text{ and } (es_i \in E_1 \text{ or } es_i \in E_2)\}$ .

For an *OR-join*, the result of effect accumulation is denoted by  $\text{ORacc}(E_1, E_2, e) = \text{ANDacc}(E_1, E_2, e) \cup \text{XORacc}(E_1, E_2, e)$ . The role of guard conditions within effect annotations is also important. Consider the first activity  $t$  on an outgoing sequence flow from an OR- or XOR-split.

Let  $E$  be the set of effect scenarios annotating the activity immediately preceding the XOR-split and let  $E' \subseteq E$  such that each effect scenario  $E'$  is consistent with the guard condition  $c$  associated with that outgoing flow. Then the set of effect scenarios of  $t$  is given by  $\{a \mid a \in \text{acc}(e \wedge c, e_t) \text{ and } e \in E'\}$ , where  $e_t$  is the immediate effect annotation of  $t$  and  $e \wedge c$  is assumed without loss of generality to be represented as a set of prima implicates.

We note that the procedure described above does not satisfactorily deal with loops, but we can perform approximate checking by partial loop unraveling. We also note that some of the effect scenarios generated might be infeasible. Our objective is to devise decision-support functionality in the compliance management space, with human analysts vetting key changes before they are deployed.

### 3.2 Accumulating Non-Functional Effects

We use scenario identifiers (see Section 3) to compute cumulative QoS measures. This leads to a cumulative measure per effect scenario. Recall that a scenario identifier is a sequence composed of activity identifiers or sets consisting (recursively) or scenario identifiers. We use the sets in the label to describe parallel branches. We therefore need to use our algebraic *parallel accumulation operator* ( $\otimes$ ), one for each QoS factor, to specify how cumulative QoS measures, propagated along parallel branches, get combined together at a join gateway.

## 4 Semantic Process Net Integration

Business process proximity is used during integration to establish a distance measure between two SPNets. Intuitively, this measure is used to ensure that the integrated model is as similar as possible to its two antecedents. In other words, we would like to minimize the deviation of an integrated model from its ancestors, thereby utilizing the previous legacy configuration and minimizing effort during integration.

**Definition 2** Associated with each SPNet is a proximity metric:  $d(p_i, p_j)$ ; which given an integrated process  $p_i$ , and one of its antecedents  $p_j$ , computes the distance of  $p_i$  from  $p_j$  w.r.t. a combination of structural and semantic criteria, alternatively defined as either (or by combining):

- $d_V(p_i, p_j) + d_E(p_i, p_j) + d_S(p_i, p_j)$ ;
- $w_V d_V(p_i, p_j) + w_E d_E(p_i, p_j) + w_S d_S(p_i, p_j)$ ;
- $\frac{d_V(p_i, p_j)}{D_V(p_i, p_j)} + \frac{d_E(p_i, p_j)}{D_E(p_i, p_j)} + \frac{d_S(p_i, p_j)}{D_S(p_i, p_j)}$ ;

such that:  $d_V$ ,  $d_E$ , and  $d_S$  are node, edge, and semantic (effect) proximity metrics;  $w_V$ ,  $w_E$ , and  $w_S$  are weights for each metric; and,  $D_V$ ,  $D_E$ , and  $D_S$  indicate the maximum hypothetical distance.

In order to compute our structural (node and edge) distance metrics, we consider sets of nodes  $V(p)$  and edges  $E(p)$  of each model  $p$  in the following way:  $d_V(p_i, p_j) = |V(p_i) \Delta V(p_j)|$ ; and,  $d_E(p_i, p_j) = |E(p_i) \Delta E(p_j)|$ . Note, that for an SPNet encoding of a BPMN model, we only consider edges of type: *control*, *message*, and *assignment*. In addition:  $D_V(p_i, p_j) = |V(p_i)| + |V(p_j)|$ , and  $D_E(p_i, p_j) = |E(p_i)| + |E(p_j)|$ . These measures are used in the last instance to help reduce the dominance of any one structural or semantic proximity metric.

Computing semantic proximity  $d_S$  is somewhat more complicated as it relies on the possible end effect (outcome or scenario) of either process. Firstly, we require a mechanism for matching the end effects of either process. Let  $e$  be some effect scenario, let  $E$  be a set of candidate effect scenario matches, and let  $m_S(e, E) = \{e_k \in E \mid |e \Delta e_k| \leq |e \Delta e_j| \text{ for all } e_j \in E\}$  denote the set of min-cardinality different elements of the set of candidate scenarios  $E$  w.r.t. the scenario  $e$ . Thus,  $\delta_S(e, E) = \delta \in \{|e \Delta e_k| \mid e_k \in m_S(e, E)\}$  denotes a non-deterministically chosen min-cardinality difference. Let  $\Delta_S(E_i, E_j) = \{\delta_S(e_i, E_j) \mid e_i \in E_i\}$  denote the asymmetric difference between the set of scenarios  $E_i$  and  $E_j$  for corresponding processes  $p_i$  and  $p_j$ , and  $d_S(p_i, p_j) = \sum |\delta|$ , for all  $\delta \in \Delta_S(E_i, E_j)$ , where  $E_i$  and  $E_j$  correspond to the end effect scenarios of process  $p_i$  and  $p_j$  respectively. Therefore,  $d_S(p_i, p_j)$  computes the sum cardinality of each difference between an end effect scenario in  $p_i$  and a matching end effect scenario in  $p_j$ . We note that symmetric versions of this metric exist, but omit their details, along with their proofs, for future work.

In addition, cost metrics could also be incorporated into our calculation of proximity in order to incorporate the cost associated with making changes to either antecedent of an integration.

### 4.1 Semantic Process Net Integration Criteria

Any approach to process integration should view both the process state (Mendling & Simon 2006), the domain knowledge (Fankhauser et al. 1991) (Beynon-Davies et al. 1997) (Li et al. 2006), as well co-ordination characteristics (Heinz & Eder 1999) (Grossmann et al. 2005). Under our integration scheme we provide a framework for process integration based on process structural and semantic descriptions. This framework may work in combination with one of the aforementioned approaches.

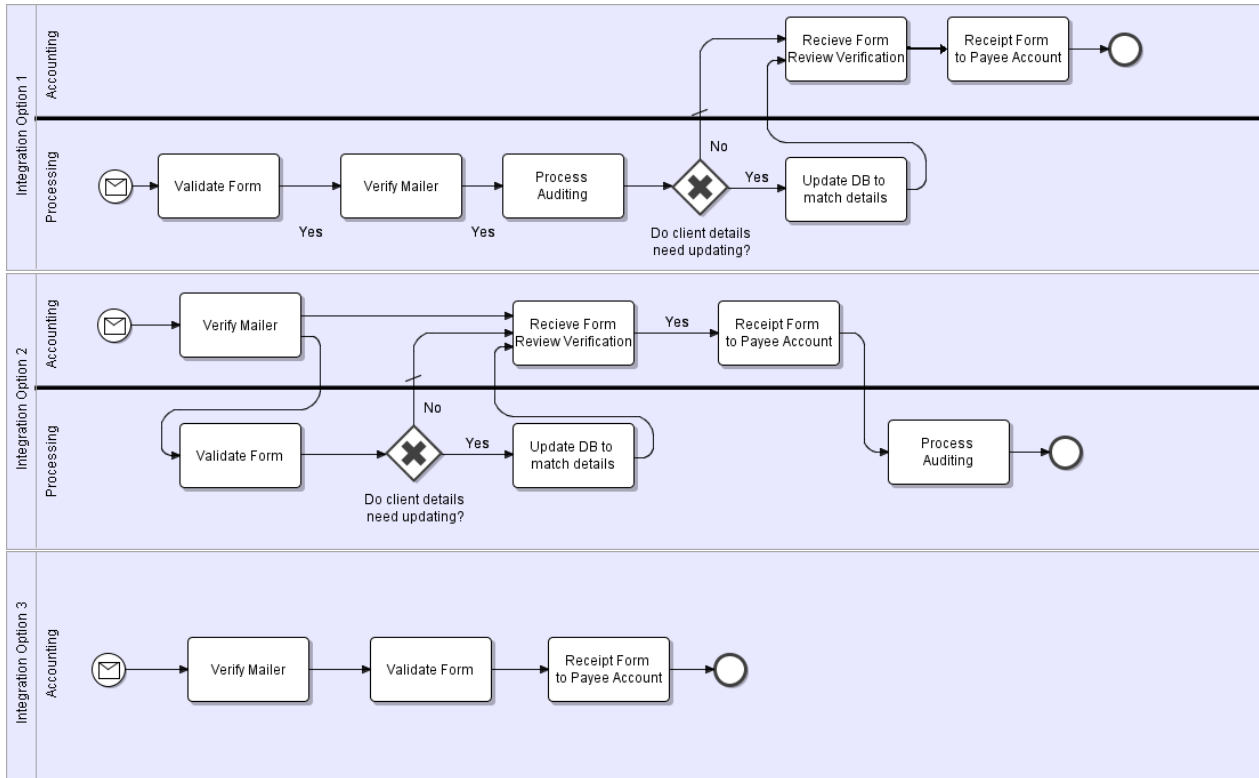


Figure 3. Example Integrations

**Definition 3** An SPNet  $p$  represents the integration of an SPNet  $p_1$  and SPNet  $p_2$  iff all of the following hold:

1. SPNet  $p$  achieves  $G_p$  (the goals associated with  $p$ ) where  $G_p = G_{p_1} \wedge G_{p_2}$  and  $G_{p_i}$  is the goal achieved by process  $p_i$ ;
2. there exists no  $p'$  such that  $p'$  achieves  $G_p$  and the following holds:  $d(p_1, p') + d(p_2, p') < d(p_1, p) + d(p_2, p)$ . Here  $d$  is a distance function between two processes. This defines an integration solution where the closest integration super process has no closer potential solution  $p'$ .

Note, that our definition of goals above applies to both the functional and non-functional properties of an SPNet.

#### 4.2 Semantic Process Net Integration Methodology

Business process integration in practice requires some effort on behalf of an analyst, during both process matching and selection of candidate integrations. The criteria we have outlined in the previous sections allow us to reduce analyst effort during the matching and selection steps (as outlined in the discussion below).

##### Step 1: Business Process Matching.

Prior to and during integration, matching is required to determine the likelihood that two business processes, or activities within a business process, share similarities or are equivalent. This may involve the use of three techniques. The first involves evaluating the labels of business processes and activities using linguistic techniques (mediated with an ontology) as in (Ehrig et al. 2007). This may help in, for example, determining that a *Package Consignment* process is semantically similar to a *Package Receiving* process. Another technique that may be applied (also in combination with an ontology) is the evaluation of the effect (or functionality) of a business process or activity. Here, the semantic aspect to our proximity metric can be re-used effectively. Finally, as processes may be represented at varying levels of abstraction, we can apply the aforementioned techniques to detect the part-whole relations among and within business processes in order to initially resolve abstraction conflicts. These three approaches to matching may be either completely automated, involve some automation, or be a simple guide applied to a completely manual integration.

##### Step 2: Business Process Integration Goals.

We firstly require a goal (describing a set of criteria) for the integrated business process to be determined. In this approach, the goal can be either given or determined by merging the effect scenarios of each business process to be integrated using an automated or semi-automated strategy. An automated strategy might involve: conjoining consistent effect scenarios; and/or, extracting the most common effects among effect scenarios. As these strategies only



**Table 1. Example Process Effects**

Accumulation Effects	Description Process 1	Description Process 2
Validation of Form	In processing after receipting	In Processing
Verification of Identity	In Accounting	In Processing
Receipting of Funds	In Accounting	In Accounting
Auditing of Work	In Processing	
Update of DB		After verification and validation
The correct payee is known to all (after verify)	In Accounting	In Processing
Database Updated	Completed after verification	

**Table 2. Example Integration Effects**

Accumulation Properties	Description Integration 1	Description Integration 2	Description Integration 3
Validation of Form	In processing	In processing	In accounting
Verification of Identity	In processing	In Accounting	In accounting
Receipting of Funds	In Accounting	In Accounting	In accounting
Auditing of Work	In processing	In processing	
Update of DB	Completed if needed after verification and verification	Completed after verification and validation	
The correct payee is known to all (after verify)	Account information given to accounting after validation and verification	Accounting passes verification information to processing and keeps information on their records	

lead to some approximate baseline, analysts will need to provide input. The requirement to firstly establish the common business goal for an integration step allows us to reduce the complexity of ad-hoc integration, as well as separating concerns and roles during the process. As discussed, the integration goal can be either computed in a bottom-up or top-down manner, and provides a concise description of the requirements for the integrated model.

### Step 3: Business Process Integration.

Business process integration involves a search through a space of possible integration options that is directed by our integration characteristics. One way to search for the most proximally efficient integration, can be to follow a local generate and test strategy. Consider an algorithm sketch: whose input is  $R$  (a repository of SPNets to be integrated); and, manipulates a set  $V$  of  $\langle spn, history \rangle$  pairs. The algorithm would 1:  $V = \{\langle spn, \langle \rangle \rangle\}$  (initialize with the model to be manipulated (possibly the intersection of nodes and edges among models); 2:  $While(!Accepted(V)) V = Step(V, R)$  (step through changes until an acceptable integration is identified). An implementation of the *Step* function would apply a single admissible addition or removal of a node or edge (possibly from elements of  $R$ ). The history would allow: poor candidates to be forgotten; ensure complementary operations are not applied to single models; ensure uniqueness is maintained across models; and provide a local basis for evaluating proximity and other heuristics. Firstly, termination could be an issue due to the infinite (gateways) nature of  $R$ , although results are anytime. There is also a large branching factor, although the metrics we have defined guide search.

## 5 Semantic Process Net Integration: An Example

In order to demonstrate the framework described within this paper, we will present a worked example of process integration, given two processes as in Fig. 1 and Fig. 2. Each process describes a way in which a business completes the task of '*receipting cheques into the organization*'. For the two given processes each activity contributes a number of effects that work to achieve a global organizational goal. These Effects are shown in Table. 1.

To compute the Node Proximity Metric we consider each node in each BPMN process diagram. Using Integration 1 from Fig. 3 we compute the delta with Process 1 from Fig. 1. In process one there are 7 BPMN process nodes (including events). In integration solution 1 there are 9 BPMN process nodes (including events and gateways). We follow the same process to compute the similarity of edges. In order to consider values for our semantic metric we have in the case of our example considered differences in the effect of completing each activity as a representation as described in Table.1 and Table.2 (this could as easily be replaced with word average per activity name, annotations, or effects).

Once these effects have been found and listed an analyst or automated task may then proceed to generate various integrated process schema's. We have provided 3 possible integration models in Fig. 3. These models vary in terms of Actors rolls and ordering. During this stage the naming conventions have remained the same (as such there is little semantic deviation), and relative ordering based on compliance constraints have been left untouched.

**Table 3. Example Integration Effects: Node Proximity**

Node Proximity	Process 1 ( $P_1$ )	Process 2 ( $P_2$ )
Integration( $I_1$ )	$\frac{nd_{P_1}(7)\Delta nd_{I_1}(9)}{ P_1 + I_1 } = \frac{1}{8}$	$\frac{nd_{P_2}(8)\Delta nd_{I_1}(9)}{ P_1 + I_1 } = \frac{1}{17}$
Integration( $I_2$ )	$\frac{nd_{P_1}(7)\Delta nd_{I_2}(9)}{ P_1 + I_2 } = \frac{1}{8}$	$\frac{nd_{P_2}(8)\Delta nd_{I_2}(9)}{ P_1 + I_2 } = \frac{1}{17}$
Integration( $I_3$ )	$\frac{nd_{P_1}(7)\Delta nd_{I_3}(5)}{ P_1 + I_3 } = \frac{1}{6}$	$\frac{nd_{P_2}(8)\Delta nd_{I_3}(5)}{ P_1 + I_3 } = \frac{3}{13}$

**Table 4. Example Integration Effects: Edge Proximity**

Edge Proximity	Process 1 ( $P_1$ )	Process 2 ( $P_2$ )
Integration( $I_1$ )	$\frac{ed_{P_1}(6)\Delta ed_{I_1}(9)}{ P_1 + I_1 } = \frac{1}{5}$	$\frac{ed_{P_2}(8)\Delta ed_{I_1}(9)}{ P_1 + I_1 } = \frac{1}{17}$
Integration( $I_2$ )	$\frac{ed_{P_1}(6)\Delta ed_{I_2}(10)}{ P_1 + I_2 } = \frac{1}{4}$	$\frac{ed_{P_2}(8)\Delta ed_{I_2}(10)}{ P_1 + I_2 } = \frac{1}{9}$
Integration( $I_3$ )	$\frac{ed_{P_1}(6)\Delta ed_{I_3}(4)}{ P_1 + I_3 } = \frac{1}{5}$	$\frac{ed_{P_2}(8)\Delta ed_{I_3}(4)}{ P_1 + I_3 } = \frac{1}{3}$

**Table 5. Example Integration Effects: Semantic Proximity**

Semantic Proximity*	Process 1 ( $P_1$ )	Process 2 ( $P_2$ )
Integration( $I_1$ )	$\frac{sd_{P_1}(7)\Delta sd_{I_1}(9)}{ P_1 + I_1 } = \frac{1}{8}$	$\frac{sd_{P_2}(8)\Delta sd_{I_1}(9)}{ P_1 + I_1 } = \frac{1}{17}$
Integration( $I_2$ )	$\frac{sd_{P_1}(7)\Delta sd_{I_2}(9)}{ P_1 + I_2 } = \frac{1}{8}$	$\frac{sd_{P_2}(8)\Delta sd_{I_2}(9)}{ P_1 + I_2 } = \frac{1}{17}$
Integration( $I_3$ )	$\frac{sd_{P_1}(7)\Delta sd_{I_3}(5)}{ P_1 + I_3 } = \frac{1}{6}$	$\frac{sd_{P_2}(8)\Delta sd_{I_3}(5)}{ P_1 + I_3 } = \frac{3}{13}$

Once these alternate integration option have been created, we now use ratio proximity relations to compare nodes, edges, and semantics between the original processes and the integrated possibilities. In the tables. 3, 4, 5 we have computed the various proximity relations in order to find an overall solution. After analyzing the results obtained, we have found the Integration 2 is the prime candidate for the role of an integration of process 1 and process 2.

In this example we have shown that using proximity metrics across node, edge and semantic values we are able chose an appropriate integration solution. These metrics can be further formulated using the weighted metrics for each candidate depending on the application. For a business wishing to integrate processes with a goal of broader knowledge consistency (language based), attention to structural deviation may be modulated to have a lesser impact in the decisional stages.

## 6 Summary and Conclusions

The interesting element of our method is in the use of *minimal change* of processes. This acts in favor of a business implementing a change management solution in terms of costs minimization (as it costs less to change less), and also in the reduction of change risks. This risk is of growing concern for compliance reasons, as with strict regulative control acting on many businesses it is assumed that broad innovative changes to processes as the result of any integration activity may leave an organization vulnerable to breaks in the value chain or penalties bought about by uncompleted activity steps.

In this work we have presented an innovative method of

process integration using Semantic Process Networks. This framework can be used a means to complete process integration. As a continuation of our work into this area we would like to explore additional instantiations of our proximity metrics and validate our approach in a controlled setting.

## References

- Becker, J., Indulska, M., Rosemann, M. & Green, P. (2005), Do process modelling techniques get better?, in 'Proceedings of the 16th Australasian Conference on Information Systems'.
- Beynon-Davies, P., Bonde, L., McPhee, D. & Jones, C. (1997), 'A collaborative schema integration system', *Computer Supported Cooperative Work (CSCW)* **6**, 1–18. URL: <http://dx.doi.org/10.1023/A:1008627102073>
- Bistarelli, S. (2001), *Soft Constraint Solving and Programming: a General Framework*, PhD thesis, Computer Science Department, University of Pisa.
- Busygina, S., Prokopyev, O. & Pardalos, P. M. (2008), 'Biclustering in data mining', *Comput. Oper. Res.* **35**(9), 2964–2987.
- Ehrig, M., Koschmider, A. & Oberweis, A. (2007), Measuring similarity between semantic business process models, in 'Proc. of the Fourth Asia-Pacific Conf. on Conceptual Modelling'.

- Fankhauser, P., Kracker, M. & Neuhold, E. J. (1991), 'Semantic vs. structural resemblance of classes', *SIGMOD Rec.* **20**(4), 59–63.
- Fiddaman, T. (1997), Feedback Complexity in Integrated Climate-Economy Models, Ph.d. thesis, MIT Sloan School of Management.  
**URL:** <http://metasd.com/papers/dissabstract.html>
- Ghose, A. & Koliadis, G. (2007), Auditing business process compliance, in 'Proceedings of the International Conference on Service-Oriented Computing', Springer LNCS, pp. 169–180.  
**URL:** [http://dx.doi.org/10.1007/978-3-540-74974-5\\_14](http://dx.doi.org/10.1007/978-3-540-74974-5_14)
- Grossmann, G., Ren, Y., Schrefl, M. & Stumptner, M. (2005), *Behavior Based Integration of Composite Business Processes*, Springer Berlin / Heidelberg, pp. 186–204.  
**URL:** [http://dx.doi.org/10.1007/11538394\\_13](http://dx.doi.org/10.1007/11538394_13)
- Grossmann, G., Schrefl, M. & Stumptner, M. (2004), Classification of business process correspondences and associated integration operators, Springer Berlin / Heidelberg, pp. 653–666.  
**URL:** <http://www.springerlink.com/content/5l0qh8f1dnf96m93>
- Heinz, F. & Eder, J. (1999), Towards an automatic integration of statecharts, in 'ER '99: Proceedings of the 18th International Conference on Conceptual Modeling', Springer-Verlag, pp. 430–444.
- Hinke, T. (1988), Inference aggregation detection in database management systems, in 'Security and Privacy, 1988. Proceedings., 1988 IEEE Symposium on', pp. 96–106.
- Huang, Z. & Ng, M. (1999), 'A fuzzy k-modes algorithm for clustering categorical data', *Fuzzy Systems, IEEE Transactions on* **7**, 446–452.
- Kementsietsidis, A., Arenas, M. & Miller, R. J. (2003), Mapping data in peer-to-peer systems: semantics and algorithmic issues, in 'SIGMOD '03: Proceedings of the 2003 ACM SIGMOD international conference on Management of data', ACM, New York, NY, USA, pp. 325–336.
- Lewis, D. D. (1998), Naive (Bayes) at forty: The independence assumption in information retrieval., in C. Nédellec & C. Rouveirol, eds, 'Proceedings of ECML-98, 10th European Conference on Machine Learning', number 1398, Springer Verlag, Heidelberg, DE, Chemnitz, DE, pp. 4–15.  
**URL:** [citeseer.ist.psu.edu/lewis98naive.html](http://citeseer.ist.psu.edu/lewis98naive.html)
- Li, Q., Shan, Z., Hung, P. C. K., Chiu, D. K. W. & Cheung, S. C. (2006), Flows and views for scalable scientific process integration, in 'InfoScale '06: Proceedings of the 1st international conference on Scalable information systems', ACM, New York, NY, USA, p. 30.
- Mendling, J. & Simon, C. (2006), Business process design by view integration, in 'Proceedings of the BPM 2006 Workshops, Workshop on Business Process Design BPD 2006', Vol. 4103 of *Lecture Notes in Computer Science*, Springer-Verlag, Vienna, Austria, pp. 55–64.
- Morimune, K. & Hoshino, Y. (2008), 'Testing homogeneity of a large data set by bootstrapping', *Math. Comput. Simul.* **78**(2-3), 292–302.
- van Dongen, B., Dijkman, R. & Mendling, J. (2008), Measuring similarity between business process models, in 'Advanced Information Systems Engineering', Springer, pp. 450–464.  
**URL:** <http://www.springerlink.com/content/xv3503k264370475>
- van Dongen, B., van der Aalst, W. & Verbeek, H. (2005), Verification of epcs: Using reduction rules and petri nets, in 'Advanced Information Systems Engineering', Springer, pp. 272–286.  
**URL:** <http://www.springerlink.com/content/g8cpj5rqbtchb6a1>
- Wang, G. & Miller, S. (2005), Intelligent aggregation of purchase orders in e-procurement, in 'EDOC Enterprise Computing Conference, 2005 Ninth IEEE International', pp. 27–36.
- White, S. (2006), Business process modeling notation (bpmn), Technical report, OMG Final Adopted Specification 1.0 (<http://www.bpmn.org>).



# Conceptional Modeling and Analysis of Spatio-Temporal Processes in Biomolecular Systems

Andreas Schäfer<sup>1</sup>Mathias John<sup>2</sup>

<sup>1</sup> Department of Computing Science, University of Oldenburg, Oldenburg, Germany  
Email: schaefer@informatik.uni-oldenburg.de

<sup>2</sup> Department of Computing Science, University of Rostock, Rostock, Germany  
Email: mjohn@informatik.uni-rostock.de

## Abstract

In life science, deeper understanding of biomolecular systems is acquired by computational modeling and analysis. For the modeling of several kinds of reaction networks, e.g. signaling pathways, information on intracellular space, like the locations and motions of molecules, has to be taken into account. In this paper, we introduce Labeled SpacePi, an extension of the  $\pi$ -calculus, in order to model spatio-temporal processes in cells. The formalism is tailored to the available data and knowledge about biomolecular systems. For the analysis, we employ model checking techniques known from the field of safety-critical systems. To this end, we develop a translation of Labeled SpacePi models into hybrid automata. Two use cases - one considering the activation of a signaling pathway and the other one concerning active transport in cells - demonstrate our concept by making use of the established analysis tools HyTech and HySat.

**Keywords:** conceptional modeling, spatio-temporal modeling, biological data,  $\pi$ -calculus, hybrid systems, model checking

## 1 Introduction

In the context of life science, intracellular processes on a molecular level become of increasing interest, e.g. (Polakis 2007, Thompson 1995). However, even with the help of modern wet-lab techniques, a complete understanding of biomolecular systems is hard to obtain. Therefore, biologists are supported on their investigations by deploying methods of computational modeling. To this end, the modeling needs to take spatial information into account, because several processes in human (eukaryotic) cells are strongly dependent on the locations of molecules (Kholodenko 2006).

The subject of this work is to present a modeling concept, that allows for the investigation of spatial effects in biomolecular systems. As a first contribution, based on the process algebra SpacePi (John, Ewald & Uhrmacher 2008), we define a modeling formalism that can take various kinds of spatial data into account but also avoids to require data that is hard or impossible to collect - an essential point for

a modeling approach, which is both, meaningful and practical. We extend the SpacePi semantics by labels similar to the labels known from hybrid logics (Areces & ten Cate 2006). Therefore, we call our formalism Labeled SpacePi. We use the labels for grouping and syntactically identifying subprocesses of a system that constitute individuals. The labels also carry information on position and movement. This allows for a clearer and more concise presentation than the original semantics. We develop a formal verification technique for a subset of Labeled SpacePi. It considers a system's entire set of possible evaluations and proceeds by constructing hybrid system models (Alur et al. 1992). As it is a domain of active research, we profit from the results achieved in the field of verification of hybrid systems and can make use of the existing tools. In this paper, we employ the bounded verification tool HySAT (Fränzle et al. 2007) and the classical tool HyTech (Henzinger et al. 2001). Finally, two use cases are given, that illustrate the application of Labeled SpacePi to the spatial modeling of intracellular processes. One focuses on the activation of the Wnt signaling pathway (Polakis 2007) and the other on active transport in cells.

For the spatial modeling of biomolecular systems, input data is mainly provided in the form of molecule locations, i.e. the spatial distribution of the different protein sorts, including intra-cellular structures, like membranes or microtubules. In this context, the image analysis technique presented in (Zhao & Murphy 2007) recently gained much attention. Based on a set of microscopic images, it computes intracellular maps, i.e. spatial models representing sets of observed cells. Another technique, stemming from the field of microscopy, called *Fluorescence Recovery After Photobleaching* (FRAP) (Meyvis et al. 1999), allows to obtain diffusion constants that describe the spreading of molecules. Databases provide the volumes of many molecules, e.g. (Letunic et al. 2006), and different tools exist for predicting the three-dimensional structure of proteins, e.g. (Zuker 2003, Rivas & Eddy 1999). Thus, a detailed image of intracellular space can be drawn by combining data of different sources. By contrast, only little information is given about the interaction of proteins, i.e. their logical order is mostly only assumed. Quantitative descriptions in the form of rate constants are rarely given, because reaction constants are hardly observable in experiments and although important for the modeling only of marginal interest for the biologists. Therefore, projects are planned that shall investigate the deployment of computational methods to determine rate constants (Takahashi et al. 2003).

The desired results of our modeling are statements about the temporal development of the spatial distribution of molecules. In particular, it is the goal to check, if, given some spatial topology, initial distribution of molecules, and logical order of reactions,

---

We thank Orianne Mazemondet for her support on the biological background. This research was partially supported by the DFG in the context of the Research Training School "dIEM oSiRiS".

Copyright ©2009, Australian Computer Society, Inc. This paper appeared at the Sixth Asia-Pacific Conference on Conceptual Modelling (APCCM 2009), Wellington, New Zealand, January 2009. Conferences in Research and Practice in Information Technology, Vol. 96. Markus Kirchberg and Sebastian Link, Eds. Reproduction for academic, not-for profit purposes permitted provided this text is included.

some observed spatial distribution of molecules can occur at a certain point in time. By this means, existing hypotheses about the system under study can be evaluated. Additionally, it shall be possible to approximate earliest time points of specific events, e.g. a certain number of molecules reaches some location, and thus to suggest new experimental settings.

Our modeling describes molecules as individuals of certain size and shape with some starting position in real space and some motion function. Reactions are represented by interactions of individuals resulting in new individuals, see Sec. 2. They occur whenever interaction partners are sufficiently close. It is also possible to define sets of non-interacting molecules of the same sort (multiplicities). In this way, model complexity can be reduced, such that the analysis process is less computational costly, an important point regarding practicability, see Sec. 5. In order to introduce obstacles that interfere with molecular motion, e.g. membranes, interactions can be marked as urgent, i.e. they are performed as soon as possible. By contrast, all other interactions can occur but do not necessarily need to. This allows for the approximation of stochasticity as e.g. the fact, that not every collision of two molecules is leading to a reaction.

The paper is structured as follows: in Sec. 2, the syntax and semantics of Labeled SpacePi is introduced, including the concepts of obstacles and multiplicities. Sec. 3 presents the model analysis approach that makes use of a translation of Labeled SpacePi into hybrid automata. In order to relate Labeled SpacePi processes to the constructed hybrid automata, a spatio-temporal bisimilarity is defined. Furthermore, the Sec. contains a short introduction to the verification tools HyTech and HySat. These are used in Sec. 4 for the analysis of the exemplary models. In Sec. 5 an overview about related work is given and Sec. 6 concludes the paper.

## 2 Labeled SpacePi

Our modeling formalism is based on SpacePi, which is itself a derivative of the  $\pi$ -calculus (Milner 1999). Therefore, it adopts the ideas presented in (Regev & Shapiro 2002) for describing biomolecular systems. Molecules are represented as concurrent processes and reactions as communication channels, on which processes can synchronize. Since communication in the  $\pi$ -calculus is always performed by one sender and one receiver, reactions are restricted to two reactants. By contrast, there is no maximum number of products, since after communication, a process can proceed with any number of concurrent processes. In general, a reaction network

$$\begin{aligned} r_1 : R_1 + R_3 &\Rightarrow P_1 + \dots + P_n \\ r_2 : R_2 + R_3 &\Rightarrow Q_1 + \dots + Q_n \end{aligned}$$

can be described with three processes

$$\begin{aligned} R_1 &= \overline{r_1}().(P_1 | \dots | P_n), R_2 = \overline{r_2}().(Q_1 | \dots | Q_n) \\ R_3 &= r_1() + r_2() \end{aligned}$$

where  $+$  denotes the choice of a process to participate in one of two (or more) reactions. Note, that the mapping from reaction networks to  $\pi$ -calculus models is not unambiguous.

SpacePi extends the  $\pi$ -calculus by processes that have real space positions, movement functions, and channels with distance values. Communication can only occur, if the distance between the sender and the receiver is smaller than or equals the distance value of

the corresponding channel. In this section, we refine SpacePi. Processes are now associated with shapes and have the ability to transmit positions and movement functions - a helpful feature as shown in Section 4.2. Additionally, a new, more concise, semantics is given, which makes use of labels as known from hybrid logics. Finally, concepts are introduced to describe membranes (obstacles) and to represent molecule sets as single individuals (multiplicities).

### 2.1 Syntax

The  $\pi$ -calculus employs the alphabet  $\mathcal{N}_c$  of *channel names*. Channels can be used for communication and also be communicated over other channels. As an extension, Labeled SpacePi additionally incorporates the alphabets  $\mathcal{N}_f$  of *symbols of movement constraints*,  $\mathcal{N}_p$  of *symbols of position constraints*, and  $\mathcal{N}_s$  of *symbols of size constraints* in order to assign spatial parameters, i.e. positions, sizes, and movements, to processes. The symbols in these alphabets are interpreted by an *interpretation*  $\iota$  which assigns a predicate to each of the constraints. Positions and sizes are predicates over coordinates in  $\mathcal{R}^n$  and movements are predicates over coordinates and their derivatives.

**Example 1.** A predicate  $\iota(f) = 1 \leq \dot{x}^2 \leq 2 \wedge \dot{y}^2 = 0$  specifies that the agent moves with velocity 2 in  $x$  direction and does not move in  $y$  direction.

Two agents can communicate over a channel if their distance is lower than or equal the real number specified by  $\iota$  for this channel. To identify processes, we use an alphabet  $\mathcal{N}_i$  of *identification labels*. The union of all alphabets, that are assumed to be disjoint, is called  $\mathcal{N}$ , the alphabet of *names*. Processes can exchange names of all these types.

Technically, we need to identify, which part of a process definition is considered to form an *agent* with an own identity, initial position, movement and size and also to assign the corresponding spatial parameters to it. To this end, we adopt the idea of labels, which stems from the field of hybrid logics (Areces & ten Cate 2006). The scope of the label defines, which subprocesses “belong together” and the label itself contains the corresponding parameters. We now define the syntax and semantics of Labeled SpacePi. We first consider that one agent represents a single biological entity. Subsequently, we show how to extend the model such that one agent can be interpreted as a representative of a set of equivalent biological entities.

**Definition 1** (Syntax). The set of Labeled SpacePi processes is defined by

$$\begin{aligned} P ::= & \sum \pi_i.P_i \mid P_1 \mid P_2 \mid \nu a : \mathcal{E}.P \mid \\ & \alpha_i(i).P \mid \alpha_p(p).P \mid \alpha_f(f).P \mid \alpha_s(s).P \mid A(\tilde{a}) \mid \\ & m : P \text{ if } P \text{ does not contain a label} \end{aligned}$$

where  $\pi_i$  is an action,  $a \in \mathcal{N}$  is a name and  $\mathcal{E}$  is a boolean combination of (in-)equalities for defining the interpretation. The names  $p \in \mathcal{N}_p$ ,  $f \in \mathcal{N}_f$ ,  $s \in \mathcal{N}_s$  are symbols for a position, movement, and size constraint, respectively.  $A$  is a process identifier, and  $\tilde{a}$  a vector of names. We assume that for each  $A$  there is exactly one defining equation  $A \triangleq P_A$ . An *action*  $\pi_i$  can be one of the three following forms:  $\bar{x}(y)$ , i.e. send  $y$  over channel  $x$ ,  $x(y)$ , i.e. receive  $y$  over channel  $x$ , or  $\tau_q$ , the silent timed action with  $q \in \mathcal{Q} \cup \{\infty\}$  indicating a delay of  $q$  time units. The  $\nu$  operator is used to create new names, i.e. *bound* names. Bound names are subject to alpha conversion, such that it is not possible to fix the interpretation  $\iota$  for symbols

in beforehand. Therefore, we allow to put a defining (in)-equality  $\mathcal{E}$ . For channels  $\mathcal{E}$  is a distance  $\in \mathbb{R}^+$ , for positions and sizes a set of predicates over coordinates  $\in \mathcal{R}^n$ , and for movements a set of predicates over coordinates and their derivatives. As labels represent the identity of agents and the three spatial parameters, we assume that they are 4-tuples having the form  $m = \begin{bmatrix} i & f \\ p & s \end{bmatrix}$  where  $i \in \mathcal{N}_i$  is an identifier, and  $f \in \mathcal{N}_f$ ,  $p \in \mathcal{N}_l$ ,  $s \in \mathcal{N}_s$  are predicate symbols for the three spatial parameters. To increase intelligibility, a label is denoted by a symbol  $m$ , if we do not refer to its components. The components of a label are modified using the apply operators  $\alpha$ . E.g. the process  $A \triangleq m : \nu i \in \mathcal{N}_i. \alpha_i(i). \alpha_f(f). B \mid A$  creates a new identifier  $i$  and assigns it to  $\alpha_f(f).B$ . The  $\alpha_f(f)$  action then modifies the movement of the new agent  $B$ . Due to the recursive definition, the process  $A$  can create an unbounded number of new agents, each having a movement function defined by  $f$ .

## 2.2 Semantics

The reduction semantics of a  $\pi$ -calculus process is defined by a transition system, where each node is a process term and each transition a possible evolution. The definition is usually given in terms of reduction rules. We adopt this technique. However, in Labeled SpacePi, nodes not only contain process terms but also information on the interpretation  $\iota$  of the predicate symbols from the alphabets  $\mathcal{N}_f$ ,  $\mathcal{N}_c$ , and  $\mathcal{N}_s$ , the current position for each label, and a clock for checking timeouts. In the two dimensional case, we write the interpretation  $\iota$  for position and size as an inequality over the variables  $x$  and  $y$  and for the movement as an inequality over the variables  $x, y, \dot{x}$ , and  $\dot{y}$ . The dotted variables represent derivatives. We further write  $(a, b) \in \iota(p)$  to denote that the pair  $(a, b)$  satisfies the predicate  $\iota(p)$ .  $\iota(p)(a, b)$  refers to the defining equality of  $\iota(p)$  in which the free variables are substituted by  $a$  and  $b$ .

**Example 2.** Let  $\begin{bmatrix} i & f \\ p & s \end{bmatrix}$  be a label. An interpretation  $\iota(f) = 1 \leq \dot{x}^2 + \dot{y}^2 \leq 2$  specifies that the agent labeled with  $\begin{bmatrix} i & f \\ p & s \end{bmatrix}$  moves with a velocity between 1 and 2 and the interpretation  $\iota(s) = -1 \leq x \leq 1 \wedge -1 \leq y \leq 1$  that the shape of the agent is a square of length 2. Similarly, interpretation  $\iota(p) = -1 \leq x \leq 1 \wedge -1 \leq y \leq -1$  describes the initial position of the agent.

For each label  $m$ , we introduce three variables: a timeout clock  $c_m$  for the  $\tau_t$  action that is reset after every reaction in which the agent is involved and two variables  $x_m, y_m$  that represent a reference point for the shape of the process that is identified by the label  $m$ . The states of the transition system are composed of a labeled process term  $P$ , an interpretation  $\iota$ , and a real valued valuation  $v$  of the clock and the position variables. We denote by  $v[c := a]$  the valuation that coincides with  $v$  except that  $c$  is reset to the value  $a$ .

Similarly to the timed automata semantics (Bengtsson & Yi 2003, Alur & Dill 1994), we distinguish two types of transitions. *Delay transitions*  $\xrightarrow{\delta}$  model the elapsing of  $\delta$  time units. *Actions transitions*  $\rightarrow$  arise from communication and other reductions and do not consume time. They correspond to the reductions of the  $\pi$ -calculus.

The  $\pi$ -calculus uses structural congruence rules for the semantics definition. To handle the labels, we extend the structural congruence to let the labels distribute over parallel composition and restriction.

$$m : (P_1 \mid P_2) \equiv m : P_1 \mid m : P_2$$

$$\begin{bmatrix} i & f \\ p & s \end{bmatrix} : \nu x. P \equiv \nu x. \begin{bmatrix} i & f \\ p & s \end{bmatrix} : P \text{ if } x \notin \{i, f, p, s\}$$

Furthermore, we modify the rule for alpha conversion and allow a bound name to be alpha converted to a name of the same alphabet.

We now define the transition rules for Labeled SpacePi. Every process can perform a delay transition where  $\delta$  time units elapse, the clock values are increased by  $\delta$ , and the position changes according to the predicate. This is captured by the following rule.

$$(P, \iota, v) \xrightarrow{\delta} (P, \iota, v') \quad (\text{DELAY})$$

if  $v'$  satisfies to following conditions:

1.  $v'(c) = v(c) + \delta$  for all clocks  $c$ ,
2. for all positions  $x_{\begin{bmatrix} i & f \\ p & s \end{bmatrix}}, y_{\begin{bmatrix} i & f \\ p & s \end{bmatrix}}$  used in labels occurring in  $P$  there is a continuous differentiable function  $\phi_{\begin{bmatrix} i & f \\ p & s \end{bmatrix}}[0, \delta]^2 \rightarrow \mathbb{R}^2$  such that the function  $\phi_{\begin{bmatrix} i & f \\ p & s \end{bmatrix}} + (v(x_{\begin{bmatrix} i & f \\ p & s \end{bmatrix}}), v(y_{\begin{bmatrix} i & f \\ p & s \end{bmatrix}}))$  satisfies the predicate  $\iota(f)$  and  $(v'(x_{\begin{bmatrix} i & f \\ p & s \end{bmatrix}}), v'(y_{\begin{bmatrix} i & f \\ p & s \end{bmatrix}})) = \phi_{\begin{bmatrix} i & f \\ p & s \end{bmatrix}}(\delta) + (v(x_{\begin{bmatrix} i & f \\ p & s \end{bmatrix}}), v(y_{\begin{bmatrix} i & f \\ p & s \end{bmatrix}}))$ .

**Example 3.** Assume that process  $m : A$  is at position  $(1, 2)$ , i.e.  $v(x_m) = 1$  and  $v(y_m) = 2$ , the clock has the value  $v(c_m) = 3$  and the predicate for the movement is  $\dot{x} = 1 \wedge \dot{y} = 2$ . If time progresses by 2 time units, denoted by a delay transition  $(m : A, \iota, v) \xrightarrow{2} (m : A, \iota, v')$ , the process arrives at position  $(3, 6)$ , i.e.,  $v'(x_m) = 3$  and  $v'(y_m) = 6$  and the clock has value  $v'(c_m) = 5$ .

A process  $m : \tau_5.P$  waits 5 time units and can then perform an action transition, such that it evolves to  $P$  without consuming time. We use the clock valuation  $v(c_m)$  to determine the elapsed waiting time of the process  $m : \tau_t.P$ . This is correct, as the clock is reset after every reaction involving the process identified by  $m$  and leads to the following rule:

$$(m : \tau_t.P + M, \iota, v) \rightarrow (m : P, \iota, v[c_m := 0])$$

$$\text{if } v(c_m) = t \quad (\text{TAU})$$

Two sums labeled by  $m_1 = \begin{bmatrix} i_1 & f_1 \\ p_1 & s_1 \end{bmatrix}$  and  $m_2 = \begin{bmatrix} i_2 & f_2 \\ p_2 & s_2 \end{bmatrix}$  can communicate over a common channel, if the distance between the represented agents is equal or below the channel's distance. The set of points that are covered by the agent labeled  $m_1$  is the set of points satisfying the size predicate  $\iota(s_1)$  translated by the current position of the reference point  $(v(x_{m_1}), v(y_{m_2}))$ . The set of points covered by agent  $m_2$  is defined analogously. The reaction rule reads:

$$(m_1 : \bar{x}(y).P_1 + N_1 \mid m_2 : x(z).P_2 + N_2, \iota, v)$$

$$\rightarrow (m_1 : P_1 \mid m_2 : P_2 \{y/z\}, \iota, v') \quad (\text{REACT})$$

if  $\exists (\tilde{x}_1, \tilde{y}_1) : \exists (\tilde{x}_2, \tilde{y}_2) : (\tilde{x}_1, \tilde{y}_1) \in \iota(s_1)$  and  $(\tilde{x}_2, \tilde{y}_2) \in \iota(s_2)$  and  $\|(\tilde{x}_1, \tilde{y}_1) + (v(x_{m_1}), v(y_{m_1})) - (\tilde{x}_2, \tilde{y}_2) + (v(x_{m_2}), v(y_{m_2}))\| \leq \iota(x)$  and where  $v' = v[c_{m_1} := 0, c_{m_2} := 0]$ .

The condition formalizes the requirement that the first and the second agent each covers one of two points that have an euclidean distance that equal or below the channel distance. Additionally, it resets the corresponding clock after communication.

**Example 4.** Consider two processes with the shape of a square of size 1, i.e.,  $\iota(s) = 0 \leq x \leq 1 \wedge 0 \leq y \leq 1$ . Let the lower left corner of  $m_1 : \bar{a}(b).A$  be at  $v(x_{m_1}, y_{m_1}) = (0, 2)$  and of  $m_2 : \bar{a}(c).B$  be at  $v(x_{m_2}, y_{m_2}) = (0, 3)$  and let the reaction distance  $\iota(a)$  of the channel  $a$  be zero. Then both processes can react to  $m_1 : A \parallel m_2 : B \{b/c\}$  because they share the point  $(0, 3)$ .

The  $\alpha$  operators modify the label while performing an action transition. This cannot be captured by structural congruence as the clocks and positions need to be updated accordingly. At first, we define the  $\alpha_f()$  operator for setting the movement.

$$([p_s^i] : \alpha_f(f').P, \iota, v) \rightarrow ([p_s^i] : P, \iota, v') \quad (\text{APPL-F})$$

where  $v' = v[c_{[p_s^i]}^i := 0, x_{[p_s^i]}^i := x_{[p_s^i]}^i, y_{[p_s^i]}^i := y_{[p_s^i]}^i]$ , i.e., clocks are reset and the position of the process is copied and does not change. The rule for the change of the identifier is similar:

$$([p_s^i] : \alpha_i(i').P, \iota, v) \rightarrow ([p_s^i] : P, \iota, v') \quad (\text{APPL-I})$$

where  $v' = v[c_{[p_s^i]}^i := 0, x_{[p_s^i]}^i := x_{[p_s^i]}^i, y_{[p_s^i]}^i := y_{[p_s^i]}^i]$ . The rule for the application of a new position  $\alpha_p(\cdot)$  ensures that the new position satisfies the predicate  $\iota(p')$ :

$$([p_s^i] : \alpha_p(p').P, \iota, v) \rightarrow ([p_s^i] : P, \iota, v') \quad (\text{APPL-P})$$

where  $v'(c_{[p_s^i]}^i) = 0, (v(x_{[p_s^i]}^i), v(y_{[p_s^i]}^i)) \in \iota(p')$  and  $v$  and  $v'$  coincide on all other values. The  $\alpha_s(\cdot)$  operator modifies the size component of the label, resets the clocks and copies the position:

$$([p_s^i] : \alpha_s(s').P, \iota, v) \rightarrow ([p_s^i] : P, \iota, v') \quad (\text{APPL-S})$$

where  $v' = v[c_{[p_s^i]}^i := 0, x_{[p_s^i]}^i := x_{[p_s^i]}^i, y_{[p_s^i]}^i := y_{[p_s^i]}^i]$ .

The semantics of the  $\nu$  operator corresponds to the one of the  $\pi$ -calculus. However, in addition, Labeled SpacePi also allows for the creation of new symbols for position, movement or size constraints. These symbols are interpreted by the function  $\iota$ . Concerning this intuition, the  $\nu$  operator acts as an existential quantifier, where the interpretation of the quantified name can change. This is captured by the following rule:

$$\frac{(m_1 : P, \iota, v) \rightarrow (m_2 : P', \iota, v')}{(\nu x : \mathcal{E}.m_1 : P, \iota, v) \rightarrow (\nu x : \mathcal{E}.m_2 : P', \iota, v)} \quad (\text{RES})$$

where  $\iota$  and  $\iota'$  coincide on all values except the value of  $x$  and the value of  $x$  satisfies  $\mathcal{E}$ . For the definition of the parallel composition rule, we have to consider the case in which the domain of the valuation  $\tilde{v}$  of the composite process is larger than the domain of the valuation  $v$  of a component:

$$\frac{(m_1 : P, \iota, v) \rightarrow (m_2 : P', \iota, v')}{(m_1 : P \mid m_3 : Q, \iota, \tilde{v}) \rightarrow (m_2 : P' \mid m_3 : Q, \iota, \tilde{v}')} \quad (\text{PAR})$$

where  $v$  coincides with  $\tilde{v}$  on all values from the domain of  $v$  and  $v'$  coincides with  $\tilde{v}'$  on all values from the domain of  $v'$ .

The last group of rules directly corresponds to the original reduction rules of the  $\pi$ -calculus.

$$(m : A(\tilde{z}), \iota, v) \rightarrow (m : P_A \{ \tilde{z}/\tilde{x} \}, \iota, v[c_m := 0])$$

$$\text{if } A(\tilde{x}) \triangleq P_A \quad (\text{CALL})$$

$$\frac{(m_1 : P, \iota, v) \rightarrow (m_2 : P', \iota, v'), Q \equiv P, Q' \equiv P'}{(\nu x.m_1 : Q, \iota, v) \rightarrow (\nu x.m_2 : Q', \iota, v')} \quad (\text{STRUCT})$$

**Remark 2** (Multiple dimensions). Although the semantics is presented for the two dimensional case, its generalization to more dimensions is straightforward by adding variables and extending the valuation  $v$ .

## 2.3 Extensions

**Obstacles** When modeling biological phenomena, the need arises to specify obstacles, which influence and restrict the movement of agents. The first solution is to impose invariants on movement functions, e.g. by specifying that a movement in  $x$ -direction does not exceed a given threshold. However, this does not allow for the representation of moving obstacles. A different possibility is to model the obstacles by agents that send a modified movement function to the moving agents. However, this requires the transmission of the new movement function and its application to occur as soon as possible. Therefore, we extend the model by the urgent transitions that must fire on activation. In particular, we use the concept of urgent channels as known from timed automata (Bengtsson & Yi 2003). For the definition of their semantics, we employ *urgent transitions*  $\rightarrow_u$  as a third type of transitions relation. The rules REACT and APPL are modified to yield urgent transitions and the composition rules are generalized accordingly.

A delay transition can only occur if no urgent transition  $\rightarrow_u$  is possible, formally  $(P, \iota, v) \xrightarrow{\delta} (P, \iota, v + \delta)$  only if there are no  $\delta' < \delta$ ,  $\iota$  and  $P', v'$  such that  $(P, \iota, v) \xrightarrow{\delta'} (P, \iota, v + \delta')$  and  $(P, \iota, v + \delta') \rightarrow_u (P', \iota', v')$ .

For the rest of the paper, we assume, that all apply transitions are urgent.

**Multiplicities** Up to now, we have focused on the modeling of an individual agent, like one molecule. However, it is possible to regard Labeled SpacePi agents as representatives of an equivalence class of non-interacting molecules, all exhibiting the same behavior (multiplicity). This helps to reduce computational costs of the analysis process, see Sec. 3, and therefore raises practicability. We introduce multiplicities by attaching a further variable  $\#_m$  to each label  $m$  similar to the clock  $c_m$  that denotes the number of represented elements. We use a predicate symbol  $\mu(\#_m, \#'_m)$  over the variable  $\#_m$  for the old state and the variable  $\#'_m$  for the new state to define evolution of multiplicities. We extend  $\iota$  to also yield the predicate corresponding to the predicate symbol  $\mu$ . Like for the spatial parameters, we use an apply  $\alpha_c(\mu)$  operator for setting the multiplicity predicate. The semantics is given by the following urgent transition rule:

$$(m : \alpha_c(\mu).P, \iota, v) \rightarrow_u (m : P, \iota, v'). \quad (\text{APPLY-S})$$

where  $v$  and  $v'$  coincide on every value except for  $\#$  and the predicate  $\iota(\mu)$  evaluates to true, i.e.,  $(v(\#_m), v'(\#'_m)) \in \iota(\mu)$ .

**Example 5** (Circular distribution). Let an agent start in point  $p$  representing  $n$  molecules, that diffuse in all directions. A circular target of radius  $r$  is located in distance  $d$ . The number of molecules that will eventually arrive at the target can be approximated as follows: Consider the two tangential lines to the circle passing through the starting point  $p$ . The angle between both tangential lines is given by  $2\arctan(\frac{r}{d})$ . The fraction of the molecules arriving at the target is therefore  $\pi^{-1} \arctan(\frac{r}{d})$ . Hence, the number of molecules that will eventually arrive at the target is approximated by the constraint  $\mu := \#' = \# \cdot \frac{\arctan(\frac{r}{d})}{2\pi}$ .



### 3 Verification

In order to analyze Labeled SpacePi processes, we introduce a translation into hybrid automata. Thereby, we make use of the work that has been done in the fields of automatic verification of hybrid systems, in particular of the tools HyTech (Henzinger et al. 2001) and HySat (Fränzle et al. 2007).

#### 3.1 Hybrid Automata

Hybrid automata have been introduced in (Alur et al. 1992) as an automaton model for describing the behavior of hybrid systems. Based on finite automata, timed automata (Alur & Dill 1994) are equipped with real valued clocks that guard transitions. This idea is generalized by hybrid automata. Instead of clocks, a set of real valued variables is used. The evolution of the variables is guarded by differential equations that are associated to the states (also called modes) of the hybrid automaton.

**Definition 3** (Hybrid Automaton). We fix a set  $X = \{x_1, \dots, x_n\}$  of real valued variables. A *hybrid automaton*  $A$  over  $X$  as defined in (Henzinger et al. 1998) is a directed multigraph  $(V_A, E_A)$ . The states are called *control modes* and the transitions *control switches*. To each state an *invariant*, i.e. a guard on the variables, and an *activity*, i.e. a guard on the derivatives of the variables, is assigned, that are usually written as (in-) equalities. Activities are also called *flow conditions*. The transitions of hybrid automata are guarded by predicates over the free variables  $X \cup X'$  where  $x \in X$  denotes the value *before* the transition and  $x' \in X'$  the value of the same variable *after* the transition. Furthermore, events for synchronization can be assigned to transitions. The semantics is defined in terms of a transition system, whose states are pairs of control modes and valuations of the variables. As for timed automata action and delay transitions are used.

#### 3.2 From Labeled SpacePi to Hybrid Automata

We construct a hybrid automaton for a Labeled SpacePi process by computing the state space. To keep track of timeouts and the spatial positions of each movement unit, we introduce three variables and define flows and transition guards accordingly.

**Construction of the Hybrid Automaton** Let  $P$  be a Labeled SpacePi process and  $\iota$  be a name interpretation. The state space of the corresponding automaton is the set of process terms modulo structural congruence that are reachable from an initial state  $P$  according to the Labeled SpacePi semantics. For each label  $m$  occurring in the transition system, we introduce a clock  $c_m$  for handling timeouts. Additionally, the two variables  $x_m$  and  $y_m$  are defined to capture the actual position of the process that is identified by the label  $m$ . For the transitions  $\xrightarrow[c]{u}$  of the hybrid automaton, we split the guards of the Labeled SpacePi semantics into a condition  $c$ , which must evaluate to true for the transition to fire, and an update statement  $u$  defining the new values. This notation is also used by the tool HyTech. The definition of the transitions is inductive, similar to the definition of the Labeled SpacePi semantics.

To represent a delay in the hybrid automaton, we use the clock  $c_m$  corresponding to the location of the waiting process. The automaton can perform the

transition if the clock reaches the timeout value. After that the clock is reset.

$$m : \tau_t.P + M \xrightarrow[c_m := 0]{c_m = t} m : P$$

To encode the reaction, we add the constraint on the variables as the guard and reset the corresponding clocks.

$$m_1 : \bar{x}(y).P_1 + N_1 \mid m_2 : x(z).P_2 + N_2 \xrightarrow[u]{c} m_1 : P_1 \mid m_2 : P_2 \{y/z\}$$

where  $c = \iota(s_1)(\tilde{x}_1, \tilde{y}_1) \wedge \iota(s_2)(\tilde{x}_2, \tilde{y}_2) \wedge ((\tilde{x}_1, \tilde{y}_1) + (x_{m_1}, y_{m_1}) - (\tilde{x}_2, \tilde{y}_2) + (x_{m_2}, y_{m_2}))^2 \leq \iota(r)^2$  and  $u = c'_{m_1} = 0, c'_{m_2} = 0$ . The notation  $\iota(s)(\tilde{x}, \tilde{y})$  denotes the predicate, i.e., the corresponding (in-)equalities, in which the free variables are substituted by  $(\tilde{x}, \tilde{y})$ . The rules addressing the application operator mimic the semantics and reset the corresponding clocks.

$$[i f]_{p s} : \alpha_f(f').P \xrightarrow[c'_{[i f]_{p s}} = 0]{} [i f']_{p s} : P$$

$$[i f]_{p s} : \alpha_i(i').P \xrightarrow[c'_{[i f]_{p s}} = 0]{} ([i f']_{p s} : P)$$

$$[i f]_{p s} : \alpha_p(p').P \xrightarrow[u]{} ([i f']_{p s} : P)$$

where  $u = c'_{[i f]_{p s}} = 0, \iota(p)(x_{[i f]_{p s}}, y_{[i f]_{p s}})$

$$[i f]_{p s} : \alpha_s(s').P \xrightarrow[c'_{[i f]_{p s}} = 0]{} ([i f']_{p s} : P)$$

Calling a process identifier does not have a condition but resets the clock. This is necessary because the defining equation can start with a  $\tau_t$  prefix.

$$m : A(\tilde{z}) \xrightarrow[c'_m = 0]{} m : P_A \{\tilde{z}/\tilde{x}\}$$

if the identifier  $A$  is defined by  $A(\tilde{x}) \triangleq P_A$ . The last rules are needed for the inductive definition to handle parallel composition, the new operator and structural congruence.

$$\frac{m_1 : P \xrightarrow[u]{c} m_2 : P'}{(m_1 : P \mid m_3 : Q) \xrightarrow[u]{c} (m_2 : P' \mid m_3 : Q)}, \quad \frac{m_1 : P \xrightarrow[u]{c} m_2 : P', Q \equiv P, Q' \equiv P'}{m_1 : Q \xrightarrow[u]{c} m_2 : Q' \mid m_3 : Q},$$

$$\frac{m_1 : P \xrightarrow[u]{c} m_2 : P'}{m_1 : \nu x \tilde{P} \xrightarrow[u]{c'} m_2 : \tilde{P}'}$$

For the  $\nu$  operator, the condition  $c'$  is obtained from  $c$  by removing the conditions involving the symbol  $x$ , thereby realizing the existential quantification of the symbol  $x$  regarding the interpretation  $\iota$ . Furthermore, after constructing the state space, we ensure by alpha conversion that the set of bound and the set of free names are disjoint.

The activity is the same for all states. Each clock  $c_{[i f]_{p s}}$  has a derivative of 1 and for each pair of variables  $x_{[i f]_{p s}}, y_{[i f]_{p s}}$  we add the corresponding (in-)qualities  $\iota(f)$ . Furthermore, to minimize the model we reduce the number of variables by reusing them when translating into the language of a model checker.

### 3.3 Spatio-Temporal Bisimilarity

In the  $\pi$ -calculus, the set of processes that are reachable by reduction does not need to be finite. However, for finite control  $\pi$ -calculus processes not involving parallel composition under recursion, the set of reachable states is known to be finite (Dam 1997). Therefore, we use the same assumption of finite control for Labeled SpacePi to obtain a finite state hybrid automaton. To show that for finite control Labeled SpacePi processes the corresponding hybrid automaton constructed above has the same behavior, we first define (strict) spatio-temporal bisimilarity. Subsequently, we state the correspondence theorem and provide a proof sketch.

**Definition 4.** Let  $V_1$  and  $V_2$  be two finite sets of real valued variables. Let further  $T_1 = (Q_1 \times \mathbb{R}^{|V_1|}, \rightarrow_1)$  be a transition system, where the states are pairs of states from  $Q_1$  and a valuation of the variables in  $V_1$ . Let further  $T_2 = (Q_2 \times \mathbb{R}^{|V_2|}, \rightarrow_2)$  be a transition system, where the states are pairs of states from  $Q_2$  and a valuation of the variables in  $V_2$ .

A pair  $\sigma = (\sigma_s, \sigma_v)$  is a *spatio-temporal simulation* if there are constants  $\lambda_1, \dots, \lambda_{|V_1|}$  and  $b_1, \dots, b_{|V_1|}$  such that the following conditions are met:

1.  $\sigma_s \subseteq (Q_1 \times \mathbb{R}^{|V_1|}) \times (Q_2 \times \mathbb{R}^{|V_2|})$  is a relation on the states.
2.  $\sigma_v \subseteq V_1 \times \mathbb{R}^2 \times V_2$  is a relation on the variables involving a translation.
3.  $(s_1, v_1)\sigma_s(s_2, v_2)$  and  $v_1(x_1) = \lambda v_2(x_2) + b$  for all  $(x_1, \lambda, b, x_2) \in \sigma_v$  and  $(s_1, v_1) \rightarrow (s'_1, v'_1)$  implies that there is a  $(s'_2, v'_2)$  with  $(s_2, v_2) \rightarrow (s'_2, v'_2)$ ,  $(s'_1, v'_1)\sigma_s(s'_2, v'_2)$ , and  $v'_1(x_1) = \lambda v'_2(x_2) + b$  for all  $(x_1, \lambda, b, x_2) \in \sigma_v$ .

A simulation is *strict* iff  $\sigma_v \subseteq V_1 \times \{1\} \times \{0\} \times V_2$ . A simulation  $\sigma = (\sigma_s, \sigma_v)$  is a *spatio-temporal bisimulation* if  $\sigma^{-1} = (\sigma_s^{-1}, \sigma_v^{-1})$  with  $(x_2, \lambda, b, x_1) \in \sigma_v^{-1}$  iff  $(x_1, \lambda^{-1}, -b, x_2) \in \sigma_v$ .

Note, that this definition considers a  $\tau_d$  statement to be observable from the outside. Using this definition, we can relate the Labeled SpacePi process to the corresponding hybrid automaton.

**Theorem 5.** Let  $P$  be a *finite control* Labeled SpacePi process, i.e., a Labeled SpacePi process not involving parallel composition under recursion and let  $\iota$  be an interpretation of the names. Let further  $R$  be the set of states reachable from  $P$  modulo structural congruence and let  $A$  be the hybrid automaton constructed as described above. Then the transition system for the semantics of  $P$  and the transition system for the semantics of the hybrid automaton  $A$  are strictly spatio-temporal bisimilar.

At first, we show that the set of reachable states by the reduction relation is finite. Starting with a finite control Labeled SpacePi process  $Q$ , we construct a  $\pi$ -calculus process  $\tilde{P}$  by removing labels, and replacing the  $\alpha$  and  $\tau_\iota$  operators by  $\tau$  prefixes. It is easy to see that the state set of the hybrid automaton is a subset of the set  $R = \{P' \mid P \rightarrow^* P'\} / \equiv$  of  $\pi$ -calculus terms reachable by reduction from  $P$  modulo structural congruence. The  $\pi$ -calculus process is furthermore finite control by construction. From Montari and Pistore's result (Montanari & Pistore 2001) follows that every finite control process  $P$  is also finitary and therefore the set  $R$  must be finite. Therefore, the state space of the hybrid automaton is finite. As the construction of the guards of the automaton encodes the Labeled SpacePi semantics, it is clear that the semantics of the automaton and the process are strictly bisimilar related by the identity on the states and the variables.

**Urgency** Urgencies can be modeled by introducing the negation of the conjunction of all guards for the outgoing transitions as an invariant plus the bound. For example if the outgoing edge has guard  $x \leq 5$  then the invariant is  $x > 5$ . The case  $x = 5$  must be allowed, because the transition does not consume time and the invariant must be satisfied when the transition takes place. However, in tools, invariants are normally required to be convex, i.e., if the invariant is satisfied when entering and leaving the state, it must also be satisfied at all time points in between. Thus, e.g. for an invariant of the type  $x < a \vee y < b$  which is not convex, the state must be duplicated, one having the invariant  $x < a$  and the other  $y < b$ .

### 3.4 Verification Tools

**HyTech** is the classical tool but is restricted to the checking of linear hybrid automata, having only conjunctions of linear guards as flow conditions. Starting from a given state of the automaton and a linear conjunction of conditions on the initial values of the variables, it iteratively computes the possible successor states. Since this is a semi-decision procedure, termination is not guaranteed. Additionally, the tool is very sensitive to the number of variables. However, in contrast to the bounded model checking tool HySat, analysis is not limited to a predetermined number of steps, such that the entire space of reachable states can be explored.

**HySat** is a bounded model checker that combines a SAT solver with a solver for real arithmetics. Its main advantage is that it can handle more variables than HyTech. As it turned out in experiments, it is also faster and thus more suitable for analyzing larger systems with more agents. Furthermore, HySat offers a richer language for specifying arithmetical constraints, including e.g. trigonometric functions. Yet, analysis results only consider a bounded number of state transitions. However, we believe that bounded model checking is well suited for the analysis of biological systems, where observation time is naturally limited. In HySat, the discrete state space and the transition relations are encoded in boolean formulae and the continuous behavior in arithmetical constraints. The SAT approach creates boolean variables for every step of the automaton. HySat is able to determine whether a state is not reachable within a given number of steps. However, if it determines that a given state is reachable, this result is only an approximation depending on the accuracy of the arithmetic. The tool provides intervals for each real valued variable in which a satisfying valuation is expected to be found.

## 4 Examples

This Sec. gives two examples for Labeled SpacePi models and their analysis. The first example, which describes an initial step of the activation of the Wnt signaling pathway, focuses on the integration of available biological data and the handling of multiplicities. By contrast, the second example, a model of active transport in cells, is not based on biological data. Its purpose rather on presenting the treatment of process communication. In the lines of Sec. 3, the two tools HyTech and HySat are used for model analysis.

### 4.1 Activation of the Wnt Pathway

Signaling pathways are reaction networks that relay signals from the cell membrane to the cell core (nucleus) leading to changes in gene expression, see e.g.

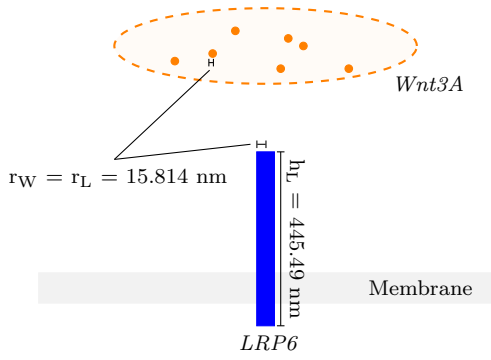


Figure 1: A simplistic illustration of the arrival of *Wnt3A* molecules at an *LRP6* receptor.

---

```
//channels
stop: 0.0
//movements
stay :=  $\dot{x}^2 + \dot{y}^2 = 0$ 
mov :=  $\dot{x}^2 + \dot{y}^2 \leq 4000$  //4000 = diffusion constant
//positions
pos_L :=  $x=0 \wedge y=0$ 
pos_W :=  $-25 \leq x \leq 25 \wedge 725 \leq y \leq 775$ 
//shapes
shape_L :=  $0 \leq x \leq 31.628 \wedge 0 \leq y \leq 445.49$ 
shape_W :=  $x^2 + y^2 \leq 15.814^2$  //circle, radius =  $r_W$ 
//multiplicities
 $\mu := \# = \# \arctan(\frac{r_W}{750-445.49}) \frac{1}{\pi}$ 
//processes
Wnt3A(stay)  $\triangleq$ 
 $\alpha_p(\text{pos}_W). \alpha_s(\text{size}_W). \alpha_f(\text{mov}). \overline{\text{stop}}(). \alpha_c(\mu). \alpha_f(\text{stay}). \tau_\infty$ 
LRP6()  $\triangleq$   $\alpha_p(\text{pos}_L). \alpha_s(\text{size}_L). \alpha_f(\text{stay}). \text{stop}(). \text{LRP6}()$ 
//initial process
( $m_1: \alpha_c(\# = 100). \text{Wnt3A}(\text{stay}) \mid m_2: \text{LRP6}()$ )
```

---

Figure 2: A simplistic Labeled SpacePi model of the arrival of *Wnt3A* at *LRP6*.

(Gomperts et al. 2002). In life science, they are of major interest, since they play a key role in the cure of e.g. cancer or Parkinson's disease. An important step in the Wnt signaling pathway is its activation by the arrival of *Wnt3A* proteins at *LRP6* receptors. In the following, a simplistic model of this event is described, see Figure 1. It illustrates the concept of composing and analyzing Labeled SpacePi models, based on a given set of biological data.

The model is given in Figure 2. *Wnt3A* is represented by the process *Wnt3A*. *Wnt3A* first applies an initial position and size and then performs the diffusive motion *mov* until it reaches *LRP6*, where it stops. The arrival of *Wnt3A* at *LRP6* is signaled by synchronization on the channel *stop*.

The data to be integrated into the model are: the volumes and shapes of molecules, their positions and diffusion constants, and their numbers. Positions and shapes are defined by boolean combinations of (in)-equalities over the considered spatial dimensions. They are bounded by the molecules' possible locations and their volumes, respectively. For *Wnt3A* we consider the volume  $V_W = 1.6566 \times 10^4 \text{ nm}^3$  and for *LRP6*  $V_L = 3,5281 \times 10^5 \text{ nm}^3$  (provided by (Letunic et al. 2006)). Because of folding processes the shapes of proteins need to be defined for each model individually. As *Wnt3A* is about 20 times smaller than *LRP6*, its shape has rather little impact, such that it is abstracted as a sphere. We assume that, as a receptor, *LRP6* is rarely folded. Thus, it is represented as a cylinder. To simplify, the radius of *LRP6* is set to  $r_L = r_W = 15.814 \text{ nm}$ , yielding the height

$h_L = 445,49 \text{ nm}$ . Additionally, we reduce the system to its two-dimensional projection. Molecular motion is defined by combinations of (in)-equalities over the considered spatial dimensions and their derivatives, which are upper bounded by diffusion constants. For *Wnt3A*, we assume the diffusion constant  $D_W = 4000 \text{ nm}^2/\text{s}$ , which is a common value for intracellular motion. Molecule numbers are mapped to multiplicities, see Sec. 5. In the initial process, the number of *Wnt3A* molecules is set to 100.

#### 4.1.1 Analysis Using HySat

We choose HySat for analysis as this tool offers a richer variety of built-in mathematical functions. Bounded model checking is sufficient to explore the whole state space here as the hybrid automaton has only two states and no loops. In the following, we first discuss the HySAT model resulting from the translation. This also sheds additional light on the translation method in general. Subsequently, we demonstrate how HySAT can be used to analyze the model.

We exemplify the translation from the Labeled SpacePi process into the input language of HySat by the snippet presented in Listing 1. The discrete state space of the hybrid automaton has two states, *init* and *stop*. The state *init* represents the system before and the state *stop* after the communication on *stop*. The continuous state space is defined by the variables corresponding to the label  $m_1$  of the *Wnt3A*, i.e.  $xm1$ ,  $ym1$  representing the spatial position, and the clock  $cm1$ . More precisely, the variables  $xm1$ ,  $ym1$  denote the center of the circle of *Wnt3A*. We further use auxiliary variables for the definition of the transitions and flow conditions.

The flow condition (flow) defines how the continuous variables evolve while the system is in the *init* state. This flow represents the movement of *Wnt3A* towards *LRP6*. In the HySat model, the flow condition is indicated by the *!jump* condition in the premise of the implication. The condition  $cm1' = cm1 + dt$  defines that the clock  $cm1$  advances by  $dt$ . The movement of the *Wnt3A* in  $x$  and  $y$  direction corresponds to the variables  $dx1$  and  $dy1$ , respectively. The constraint  $(dx1 * dx1 + dy1 * dy1) * \pi \leq DW * dt$  ensures that *Wnt3A* can at maximum proceed according to the diffusion rate  $DW$ . The variable  $m1count$  models the multiplicity.

The jump condition, indicated by *jump*, defines the discrete state transitions. The second formula encodes that *stop* is the only possible successor of the state *init*. The third formula defines the transition from state *init* to state *stop* when *Wnt3A* and *LRP6* communicate over *stop*. This transition can only occur if there is an overlapping of the participants. Formally, this requires one point ( $xsm2, ysm2$ ) inside *LRP6*, defined by conditions (i1) - (i4), that is equal to a point ( $xsm1, ysm1$ ) inside *Wnt3A*. The equality condition is formalized in (e1) and (e2), whereas Condition (i5) specifies that the point ( $xsm1, ysm2$ ) is inside *Wnt3A*. Conditions (a1) and (a2) calculate the constraint on the multiplicity similar to the approach shown in Example 5.

```
(!jump and init ->
  init' and  $cm1' = cm1 + dt$  and  $xm1' = xm1 - dx1$  and
   $ym1' = ym1 - dy1$  and  $(dx1 * dx1 + dy1 * dy1) * \pi \leq DW * dt$ 
  and  $m1count' = m1count$ ); -- (flow)
```

```
(jump and init -> stop');
```

```
(jump and init and stop' ->
   $xsm1 = xsm2$  -- (e1)
  and  $ysm1 = ysm2$  -- (e2)
  and  $xsm2 > xlPR$  -- (i1))
```

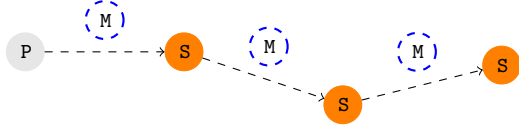


Figure 3: Active transport in SpacePi - P produces molecules M that move along static parts S.

```

and ysm2 > yiLPR          -- ( i2 )
and xsm2 < (xiLPR + xsLPR) -- ( i3 )
and ysm2 < (yiLPR + ysLPR) -- ( i4 )
and (xsm1 - xm1)^2 + (ysm1 - ym1)^2 < rW^2 (i5)
and cm1' = cm1 + dt
and sin(angle) * (yiW - ysLPR) = rW * cos(angle) -- ( a1 )
and mlcount * pi = angle * mlcount ; -- ( a2 )

```

Listing 1: Snippet of HySat Specification

We are interested in the timing behavior of the system. The goal is to establish a bound on the time of the interaction at LRP6. To this end, we define the target property that the state *stop* is reached and the clock is below a given value and let HySat check this property for our model. Iteratively decreasing the given value for the clock in the target property, reveals that HySat cannot find a solution in which *Wnt* reaches LRP6 in 17 seconds. However, for 18 seconds it can find a solution. The analysis further yields that on the long run 1.5% of *Wnt* will eventually reach LRP6. Using different constraints for the movement function or the constraint on the multiplicity, more specialized models of diffusion, e.g. with the Fick-Equations, can be modeled and analyzed in Labeled SpacePi. This is, however, outside the scope of this paper.

## 4.2 Active Transport

The term active transport addresses different sorts of molecular motion that are performed against concentration gradients and thus enable cells to overcome equal molecular distributions. The form of active transport, we focus on in this example, refers to the motion of molecules along fixed intracellular structures, like microtubules, by consecutive binding to structure parts under energy consumption. It is significantly faster than diffusion and thus leads to an acceleration of ongoing intracellular processes. The main purpose of this example is to illustrate the use of abstraction as part of the analysis process.

Figure 3 shows the basic principle of the model. The shapes of all molecules are abstracted as circles, since the included values are not related to experimental data. Processes that represent moving molecules, called M, start at process P and move along static processes S, abstracting the structure parts. M moves between two S processes by receiving the needed movement function from the first S to reach the second one.

Figure 4 reveals more details about the model. The three movement functions describe the movement of M to an S above it, underneath it, and to its right, respectively. Instances of M are produced by P with a delay of  $t_P$ . M moves from one S to the next, by receiving the appropriate movement function from *trans*. In order for M and S to communicate, the arrival time between two M at S has to be greater  $t_S$ , as denoted by  $\tau_{t_S}$ . The impact of  $t_S$  and  $t_P$  on the model is investigated in the analysis Sec. below.

### 4.2.1 Analysis

To illustrate the treatment of process communication in more detail, we analyze the interplay of the constants  $t_P$  and  $t_S$  defining the frequency with which

```

//channels
trans: 0
//constants investigated in analysis Sec.
ts := ..., tp := ...
//shapes
shapeM := x^2+y^2 ≤ 0.5^2 //circle, radius = 0.5
shapes := x^2+y^2 ≤ 0.5^2 //circle, radius = 0.5
//movements
mu := x=1, y=1
md := x=1, y=-1
ms := x=1, y=0
//processes
M() Δ trans(m).αf(m).M()
S(m) Δ τtrans(m).τts.S(m)
P() Δ τtp.(P()|νm.αi(m).αp(p1).αf(ms).M())
//initial process
m1:(αp((0,0)).P()|αp((4,1)).S(md)|
αp((4,5)).S(mu)|αp((6,1)).S(md))

```

Figure 4: A Labeled SpacePi model of active transport

new agents can be created at P and the time it takes at each location S to handle one molecule. For a concise presentation of the example, we focus on the first instance of S located at (4,1) and check whether it is possible that a M does not synchronize at this location. To obtain a finite state hybrid automaton, we underapproximate the system behavior by fixing the number of M. This is further justified as due to the  $t_P$  delay, there can always be only a finite number of M between P and the first S. For conciseness, we choose to analyze two M.

**HyTech** The construction of the HyTech model from the Labeled SpacePi process is very similar to one presented in Sec. 4.1.1. However, HyTech can only check hybrid automata in which the guards are conjunctions of linear inequalities. Therefore, we abstract the reaction radius by a rectangle. Listing 2 shows a snippet of the HyTech specification. The state *L1* represents the configuration where the first M has been released from P. The convex invariant  $4 - xm1 \geq \text{distT} \ \& \ cm1 \leq tP$  in line (1) ensures a transition as soon as either the first M labeled m1 arrives at position S or the P process has completed the waiting time. The second line encodes the movement function  $m_s$  for the first M. In this expression terms  $dx$  denote the time derivative of the variable  $x$ . For the transition from state *L1* two cases have to be distinguished. The first case (3) is that the timeout of P occurs first and the second case (4) is that the reaction occurs first.

To find out if one M does not interact at the first S, we check whether the variable  $xm2$  associated with the label of the second instance of M can reach a value right of S. This is sufficient as we only consider one S for the analysis. The requirement is fulfilled if the time  $t_P$  is much lower than the reaction time  $t_S$  at the first S such that the second M passed the inactive S. But it turns out that this can also occur if  $t_P$  is higher than initially expected which is due to the fact that the first M going down after the reaction with S can react a second time at the same S if the delay time is smaller than the time for the molecule to leave the reaction radius of S. This is a result which is not directly obvious from the model.

The experiment shows, that the use of abstraction is crucial as HyTech is otherwise not able to analyze the whole state space. Abstraction removes potentially unreachable or irrelevant states. Due to space limitations, abstraction techniques are not discussed in this paper. Similar to the previous example, the

tool HySat was also used for the analysis. It provided similar advantages as discussed above but does not explore the whole state space. Details have to be omitted due to space limitation.

```

loc L1: while xp1 - xm1 >= distT & cm1 <= tP wait --
(1)
{dxm1 = xmstr, dym1 = ymstr, dxm2 = 0, dym2 = 0}
-- (2)
when cm1 = tP
do {cm1' = 0, xm2' = xp0, ym2' = yp0}
goto L2; -- (3)
when xp1 - distT <= xm1 & xm1 <= xp1 + distT & ...
do {cmp1'=0} goto L3; -- (4)

```

Listing 2: Snippet of HyTech Specification

## 5 Related work

Different approaches have been applied to spatial modeling. Differential equations support a population-based view on systems, where species concentrations are of interest. Thus, detailed information about individual molecules and their locations cannot be taken into account. Various concepts base on space discretization (Elf & Ehrenberg 2004, Regev et al. 2004, Cardelli & Gordon 1998, John, Lhousaine, Niehren & Uhrmacher 2008), i.e. they assume sub-volumes in which molecules are equally distributed. Instead of continuous functions, these approaches make use of reactions to describe movements as events that lead to position changes. Therefore, the process definitions of the Wnt pathway example would read in e.g. stochastic Pi like  $Wnt3A() \triangleq \bar{stop}().\tau_{\infty}.LRP6() \triangleq stop().LRP6()$ . Thereby, the interaction on `stop` denotes that  $Wnt3A()$  has fulfilled its movement from its initial position to  $LRP6()$ . This more abstract view hampers the modeling of important spatial effects in cells, like molecular crowding (Takahashi et al. 2005), where molecular motion is constrained by limited space. Additionally, methods that assume sub-volumes require reaction rate constants, that are in general not at hand and hard to estimate in case of complex systems. Molecular and Brownian Dynamics, e.g. (Takahashi et al. 2003), provide a very detailed view on systems. However, their computational costs for large systems are too high to cover the normal time scale of wet-lab experiments spanning several hours.

Multiple approaches for the model checking of biological systems exist, e.g. (Ciocchetta & Hillston 2008, Calzone et al. 2006, Heath et al. 2008, Batt et al. 2005). However, they only consider spatial information in form of compartments and do not take protein locations and shapes or intracellular structures into account. The spatial logics model checker (Vieira et al. 2005) allows for the checking of a subset of  $\pi$ -calculus specifications. Yet, it does not consider physical space. Other, more recent developments for checking the  $\pi$ -calculus (Meyer et al. 2008) use methods for Petri Nets. The tool MoDiShCa (Quesel & Schäfer 2006) verifies physical mobility of systems. It translates a Shape Calculus (Schäfer 2007) specification into monadic second order logic and uses a tool for this logic as verification back-end. However, the specification of the system is declarative in a logic, it does not allow for communication of positions and movements and the verification in MoDiShCa is limited to discrete time and finite space.

There are also several approaches that extend process algebras for the modeling of hybrid systems, like the  $\Phi$  calculus (Rounds & Song 2003). However, as they do not provide built-in support for describing mobility, positions and movements have to be encoded. Developed for the modeling of satellite communication,

J. C. M. Baeten and J. A. Bergstra introduce the Real Space Process Algebra in (Baeten & Bergstra 1991), where communication has a three or four dimensional position. Yet, this approach neither considers the movement of processes nor provides automatic verification.

## 6 Conclusion & Outlook

In this paper, we introduced Labeled SpacePi, a formalism for the modeling of time and space in biomolecular systems, which is tailored to the available knowledge and data. It refines the former work in (John, Ewald & Uhrmacher 2008) by a more concise reduction semantics that makes use of labels as known from the field of hybrid logics. We believe that the idea of using labels in process algebra can be beneficially applied to other purposes, e.g. to specify in a  $\pi$ -calculus logic that agents share a common name. Furthermore, Labeled SpacePi provides the concept of multiplicity, i.e. sets of non-interacting can be represent as single processes, which helps to lower computational costs of model analysis. The presented approach for model analysis considers a system's entire set of possible evaluations and is based on a translation from Labeled SpacePi to hybrid automata. The advantage of having such a translation is that Labeled SpacePi allows for a simpler modeling of biological systems than hybrid automata. This is because, several concepts of Labeled SpacePi cannot be directly expressed in hybrid automata, e.g. the transmission of movement functions from one agent to another or the creation of new independent agents using the  $\nu$  operator. They can only be encoded by building a single complex hybrid automaton for the entire system yielding high modeling effort. By defining a spatio-temporal bisimilarity, we were able to relate Labeled SpacePi processes to the constructed hybrid automata. We applied our approach to two use case studies, addressing the activation of the Wnt signaling pathway and active transport in cells, and presented how properties of Labeled SpacePi models can be derived using the established tools HyTech and HySat.

Regarding future work, we would like to develop a logic for specifying system properties of biological systems, especially regarding spatial phenomena. This shall be used as a querying language for model checking. Another goal is to further lower the computational costs of the analysis process by additional abstraction techniques for obtaining state finite systems with finitely many variables. Furthermore, we would like to investigate how recent results on  $\pi$ -calculus verification like (Meyer et al. 2008) can be used for Labeled SpacePi.

## References

- Alur, R., Courcoubetis, C., Henzinger, T. A. & Ho, P.-H. (1992), Hybrid Automata: An Algorithmic Approach to the Specification and Verification of Hybrid Systems, *in* 'Hybrid Systems', pp. 209–229.
- Alur, R. & Dill, D. L. (1994), 'A Theory of Timed Automata', *TCS* **126**(2), 183–235.
- Areces, C. & ten Cate, B. (2006), Hybrid Logics, *in* 'Handbook of Modal Logics', Elsevier.
- Baeten, J. C. M. & Bergstra, J. A. (1991), Real Space Process Algebra, *in* 'International Conference on Concurrency Theory', Vol. 527 of *LNCS*, Springer, pp. 96–110.

- Batt, G., Ropers, D., de Jong, H., Geiselmann, J., Mateescu, R., Page, M. & Schneider, D. (2005), Analysis and Verification of Qualitative Models of Genetic Regulatory Networks: A Model-Checking Approach, in 'International Joint Conferences on Artificial Intelligence', pp. 370–375.
- Bengtsson, J. & Yi, W. (2003), Timed Automata: Semantics, Algorithms and Tools, in 'Lectures on Concurrency and Petri Nets', Vol. 3098 of *LNCS*, Springer, pp. 87–124.
- Calzone, L., Fages, F. & Soliman, S. (2006), 'BIOCHAM: An Environment for Modeling Biological Systems and Formalizing Experimental Knowledge', *Bioinformatics* **22**(14), 1805–1807.
- Cardelli, L. & Gordon, A. D. (1998), Mobile Ambients, in 'Foundations of Software Science and Computation Structures', Vol. 1378 of *LNCS*, Springer, pp. 140–155.
- Ciocchetta, F. & Hillston, J. (2008), 'Bio-PEPA: An Extension of the Process Algebra PEPA for Biochemical Networks', *ENTCS* **194**(3), 103–117.
- Dam, M. (1997), 'On the Decidability of Process Equivalences for the Pi Calculus', *TCS* **183**(2), 215–228.
- Elf, J. & Ehrenberg, M. (2004), 'Spontaneous Separation of Bi-Stable Biochemical Systems into Spatial Domains of Opposite Phases', *Systems Biology* **1**(2), 230–236.
- Fränzle, M., Herde, C., Teige, T., Ratschan, S. & Schubert, T. (2007), 'Efficient Solving of Large Non-linear Arithmetic Constraint Systems with Complex Boolean Structure', *J. on Satisfiability, Boolean Modeling and Computation* **1**, 209–236.
- Gomperts, B. D., Kramer, I. M. & Tatham, P. E. R. (2002), *Signal Transduction*, 1 edn, Academic Press.
- Heath, J., Kwiatkowska, M., Norman, G., Parker, D. & Tymchyshyn, O. (2008), 'Probabilistic Model Checking of Complex Biological Pathways', *TCS* **319**(3), 239–257.
- Henzinger, T. A., Kopke, P. W., Puri, A. & Varaiya, P. (1998), 'What's Decidable about Hybrid Automata?', *J. Comput. Syst. Sci.* **57**(1), 94–124.
- Henzinger, T. A., Preussig, J. & Wong-Toi, H. (2001), Some Lessons from the HyTech Experience, in 'Conference on Decision and Control', Vol. 3, IEEE Press, pp. 2887–2892.
- John, M., Ewald, R. & Uhrmacher, A. M. (2008), 'A Spatial Extension to the Pi Calculus', *ENTCS* **194**(3), 133–148.
- John, M., Lhoussaine, C., Niehren, J. & Uhrmacher, A. M. (2008), The Attributed Pi Calculus, in 'Computational Methods in Systems Biology', Vol. 5307 of *LNBI*, Springer, pp. 83–102.
- Kholodenko, B. N. (2006), 'Cell-Signalling Dynamics in Time and Space', *Nature Reviews Molecular Cell Biology* **7**(3), 165–176.
- Letunic, I., Copley, R. R., Pils, B., Pinkert, S., Schultz, J. & Bork, P. (2006), 'Smart 5: domains in the context of genomes and networks', *Nucleic Acids Research* **34**(Database-Issue), 257–260.
- Meyer, R., Khomenko, V. & Strazny, T. (2008), A Practical Approach to Verification of Mobile Systems Using Net Unfoldings, in 'Applications and Theory of Petri Nets', Vol. 5062 of *LNCS*, Springer, pp. 327–347.
- Meyvis, T., De Smedt, S., Van Oostveldt, P. & Demeester, J. (1999), 'Fluorescence Recovery After Photobleaching: A Versatile Tool for Mobility and Interaction Measurements in Pharmaceutical Research', *Pharmaceutical Research* **16**(8), 1153–1162.
- Milner, R. (1999), *Communicating and Mobile Systems: the Pi-Calculus*, Cambridge University Press.
- Montanari, U. & Pistore, M. (2001), History-Dependent Automata, Technical report, Istituto Trentino di Cultura.
- Polakis, P. (2007), 'The Many Ways of Wnt in Cancer', *Current Opinion in Genetics & Development* **17**(1), 45–51.
- Quesel, J.-D. & Schäfer, A. (2006), Spatio-Temporal Model Checking for Mobile Real-Time Systems, in 'Theoretical Aspects of Computing', Vol. 4281 of *LNCS*, Springer, pp. 347–361.
- Regev, A., Panina, E. M., Silverman, W., Cardelli, L. & Shapiro, E. (2004), 'BioAmbients: An Abstraction for Biological Compartments', *TCS* **325**(1), 141–167.
- Regev, A. & Shapiro, E. (2002), 'Cells as Computation', *Nature* **419**, 343.
- Rivas, E. & Eddy, S. R. (1999), 'A Dynamic Programming Algorithm for RNA Structure Prediction including Pseudoknots', *J. Mol. Biol.* **285**(5), 2053–2068.
- Rounds, W. C. & Song, H. (2003), The Phi-Calculus: A Language for Distributed Control of Reconfigurable Embedded Systems, in 'Hybrid Systems: Computation and Control', pp. 435–449.
- Schäfer, A. (2007), 'Axiomatisation and Decidability of Multi-Dimensional Duration Calculus', *Information and Computation* **205**(1), 25–64.
- Takahashi, K., Ishikawa, N., Sadamoto, Y., Sasamoto, H., Ohta, S., Shiozawa, A., Miyoshi, F., Naito, Y., Nakayama, Y. & Tomita, M. (2003), 'E-Cell 2: Multi-Platform E-Cell Simulation System', *Bioinformatics* **19**(13), 1727–1729.
- Takahashi, K., Nanda, S., Arjunan, V. & Tomita, M. (2005), 'Space in Systems Biology of Signaling Pathways : Towards Intracellular Molecular Crowding in Silico', *FEBS letters* **579**(8), 1783–1788.
- Thompson, C. B. (1995), 'Apoptosis in the Pathogenesis and Treatment of Disease', *Science* **267**, 1456–1462.
- Vieira, H., Caires, L. & Viegas, R. (2005), The Spatial Logic Model Checker User's Manual v1.0, Technical Report TR-DI/FCT/UNL-05/2005, Universidade Nova de Lisboa.
- Zhao, T. & Murphy, R. F. (2007), 'Automated Learning of Generative Models for Subcellular Location: Building Blocks for Systems Biology', *Cytometry Part A* **71A**(12), 978–990.
- Zuker, M. (2003), 'Mfold Web Server for Nucleic Acid Folding and Hybridization Prediction', *Nucleic Acids Res.* **31**(13), 3406–3415.



# Conceptual Application Domain Modelling

Bernhard Thalheim<sup>1</sup>

Klaus-Dieter Schewe<sup>2</sup>

Hui Ma<sup>3</sup>

<sup>1</sup> Christian-Albrechts-University Kiel, Institute of Computer Science, Kiel, Germany,  
thalheim@is.informatik.uni-kiel.de

<sup>2</sup> Information Science Research Centre, Palmerston North, New Zealand  
kdschewe@acm.org

<sup>3</sup> Victoria University Wellington, School of Mathematics, Statistics and Computer Science, Wellington, New Zealand, Hui.Ma@mcs.vuw.ac.nz

## Abstract

Application domain description precedes requirements engineering, and is the basis for the development of a software or information system that satisfies all expectations of its users. The greatest challenge in this area is the evolution of the application domain itself. In this paper we address this problem by explicit consideration of *application cases* that are defined by user profiles and intentions and the system environment, i.e. scope and context. User profiles and intentions are captured through the concept of *persona*. We show how the application domain description can be mapped to requirements and discuss engineering of application domain descriptions.

## 1 Modelling Information and Software Systems

Information Systems are software systems with a focus on operating on data and thus the information they provide. Typically, the engineering of Information Systems (or software systems in general) is divided into several phases, the first of which being requirements engineering. However, requirements may vary depending on the evolution of the application itself or changes in the technical environment or the system users.

An example consider the classical example of elevator control that is used in many textbooks, e.g. (Wieringa 2003). An elevator control system coordinates the movement of a number of elevator cages that serve a number of floors. However, this problem description already contains a fundamental decision with respect to the application domain, which is to enable easy movement of a number of users from one floor to another one. Instead of giving preference to an elevator solution the decision could have been to use only stairs or escalators, or to install a “pater noster”, i.e. a system using open boxes that turn around continuously. In terms of throughput and energy costs, a pater noster is significantly better than an elevator, but for elderly people it may be a safety hazard, and for wheelchair access it is not suited. Nevertheless, in many old business buildings in Northern Germany pater noster were quite common.

A similar observation can be made for the specification of traffic control using signaling, control by a computer, street topology, error handling, etc.

(Jackson & Sztipanovits 2006). In this case the technical solution has already been fixed prior to the elicitation of requirements.

This shows that different solutions envisioned in the application domain may lead to completely different requirements and consequently system specifications. Which solution is the most appropriate and which flexibility and variability is needed depends on the application domain. Moreover, Jackson observes that solutions based on the single-model paradigm are not achievable and integration of different viewpoints is only achievable for very simplistic situations (Jackson & Sztipanovits 2006).

### 1.1 Application Domain Description and Modelling

Bjørner divides Software Engineering into three main phases: *application domain description*, *requirement prescriptions*, and *system specifications* (Bjørner 2006). He calls these phases the *system development triptych*. The application domain description is a model describing the application domain, its entities, functions, events, and behaviour. It utilises formal, semi-formal or natural language, which permits the formulation of a set of theorems or postulates or properties that are claimed to hold for the domain model.

Thus, application domain modelling (similarly, but more specifically: product line engineering, enterprise modelling) is the first step of software development processes, for which a large body of knowledge has already been developed, e.g. (Boehm 2006, Lowe 2003, Maciaszek 2001, Wieringa 2003). In particular, the goal-oriented framework KAOS (Darimont & van Lamsweerde 1996) uses an outer semi-formal layer for capturing requirements engineering concepts and their structuring and presentation, and an inner formal assertion layer for their precise definition and reasoning about them.

A theory basis for application domain languages is currently under development, based for instance on description logics (Lambrix & Padgham 1996), artificial intelligence (Anh & Moore 1996), logical calculi (Hansen & Hung 2007), ontologies (Jackson & Sztipanovits 2006, Missikoff & Schiappelli 2005), or formal methods (Bjørner 2006). In the last case the formal VDM language is used to cover most (but not all) facets of an application domain such as business processes, intrinsics, support technology, management and organisation, rules and regulations, scripts, and human behaviour, but it is insufficient for the main facet of information systems: modelling of information states and evolution of states.

Typically, application domain modelling is based on the elicitation of properties of one application domain solution and the description of this solution. This approach is appropriate as long as the application domain is relatively stable and solutions do

not change. There are, however, applications for which the solutions evolve and change over time. In this case, requirements that have been the basis for the software system are also evolving and changing, whereas the problem supported by the system remains relatively stable. Therefore, we propose to first model the application problem itself.

## 1.2 Requirements Prescription and Information Systems Modelling

Application domain modelling can be based on features or business events (Robertson & Robertson 2005), mission statements (Wieringa 2003), or business use cases (Maciaszek 2001). The application domain model is mapped to requirements that prescribe further system development. A *requirement* (Berziss 2001, Gunter, Gunter, Jackson & Zave 2000) is a verifiable statement prescribing some property that a software system should possess. Requirements are specified through declarations or remarks, reports of facts, or opinions. Their fulfillment can be verified or measured.

Requirement statements are compiled into a documentation which prescribes desired properties of machines, i.e. what the machine should and should not offer regarding data, control and functions. Since machines do not operate without an environment, environment description is often included into requirements. Such statements can be supported by diagrammatic methods, e.g. UML.

Following requirements engineering the association between the requirements model and the conceptual model is well understood (Rolland 2006) and supported by a large variety of UML diagrams or by advanced ER models (Thalheim 2000a, Thalheim 2000b). The database schema specifies the structuring of the database. Functionality can be added on the basis of the HERM algebra. Distribution is specified through extended views and services. The user presentation system can be described on the basis of the website specification language SiteLang (Schewe & Thalheim 2005). Typically it is required that the specification satisfies the requirements.

## 1.3 Application Solution Modelling used for most Applications

Classical software development is based on a vision of the solution to a number of problems to which solution the software system has been devoted. Requirements describe the application solution for which a software support is going to be developed. Problem analysis targets in development of application solutions. These solution can be mapped to requirements. Therefore, classical software engineering is devoted to *application solution engineering*, and requires severe changes within the system, if another solution is selected.

Application solution modelling is the basis for a prescription of requirements for one envisioned solution. This solution solves a problem that might have a number of other solutions. The solution that has been chosen for the software system has advantages over other solutions, but may also be a less optimal one, if the environment, the culture or the users change.

Therefore, in this paper we extend application domain description by a framework for *application cases* that are mapped to business use cases, stories, and portfolios. The challenge of application domain description and modelling is

- to achieve a set of languages, principles, methods, theories and techniques,

- as well as a set of management practices,
- which together cover all of today's and the immediately foreseen applications, their variations and their evolution in future,
- which by careful use and thoughtful consideration support software systems during evolution from initial development via repeated adaptive and perfective maintenance to final disposition, and
- which ensure software correctness as much as humanly conceivable.

The grand challenge of application domain modelling is the evolution of the application domain itself. In particular, this becomes crucial for web information systems (WISs) due to their low 'half-life' period and the high potential of WISs for evolution, migration and integration.

## 2 Specific Demands of Web Information Systems

Our framework of application cases has already been successfully applied in the area of WISs, where a broad coverage of solutions for the same problem can be observed. Developing more than three-score WIS such as infotainment, community, edutainment and e-commerce websites we had to realise that the application domain itself must be far better taken into consideration than it is classically done for information systems or for software systems. The classical approach of assuming that the application solution is fixed is not applicable to WIS. The application solutions evolve and change over time, vary for different regions or countries, depend on the user, are dependent on the context, and must be robust for different architectures. The treatment of the application cases depends on the habits of users, on laws and regulations of countries, on the culture of system utilisation and on the policy and profile of supporting companies.

A given problem in the application domain has typically more than one solution. Some of these solutions might be senseless at the given moment of time but might be preferable at another. This change of preference is often observed when application domains become a broader scope or merge with others. Whenever users or providers are changing a different solution must be enforced. Therefore, we also base our modelling approach on problem models.

### 2.1 Web Information Systems Engineering

WIS engineering also requires description, prescription, and specification of the presentation system. Presentation systems generalise approaches developed for human-computer interfaces (Lewerenz 2000) by storyboarding, by explicit treatment of portfolio of users, by adaptation and syndication facilities, by orchestration for different platforms, and by services for delivery, collection and compilation of information. User models are typically broader and must cover a large variety of users with specific requirements. Web applications also require systems that are easy and intuitively to use. Classical user-oriented and information-intensive applications could be nowadays based on WIS.

Classically user interfaces are built in dependence of the facilities the software system is supporting. In this case, the user has to learn how the system behaves and must adapt his/her behaviour to the systems behaviour. We overcome this mismatch by primarily considering the user worlds, the user stories,



and the applications. Our approach is going to be based on *application domain description*. It extends ethnographical approaches developed for software engineering.

The main ingredients for application domain description for WIS are the characterisation of the user, the determination of the kind or scope of the system and of the context, and the description of the subject world. The latter is described through *application cases* or life case. We extract the portfolio and the tasks from application cases. Context and scope are defined in (Kaschek, Schewe, Thalheim & Zhang 2003) and (Schewe & Thalheim 2005) and thus not considered in this paper. We base the description of the profile and intention of users on (Schewe, Thalheim & Tretjakow 2006). Task modelling (Schewe & Thalheim 2005, Schewe & Thalheim 2007b) extends participatory task modelling (O'Neill & Johnson 2004) by explicit, integrated and formal specification of tasks into WIS specification. We, thus, concentrate our this paper on the first dimension of the triptych, i.e. application domain description through application cases.

## 2.2 Challenges of Modern Web Applications and of Evolving Applications

Nowadays software systems are supported by web presentation systems or are becoming web systems. They allow to cope with more complex tasks compared with systems of the past, with application stories in a large variety, and with dynamic requirements from the user and system side. They support users in their everyday life independently from their environment, their knowledge and skills level, their progress and their preferences. This change of attitudes and applications can only be satisfied if the user tasks and the user abilities are taken into consideration, if systems can be adapted on demand and on context, and if the application domain is driving the software. Early reports such as the Cutter Consortium report (Epner 2000) show that most WIS projects fail in meeting user and business needs, suffer from project delays, result in budget overrun, lack of required functionality, and provide poor quality of deliverables. These pitfalls illustrate the urgent necessity of a conceptual WIS development that is based on a severe analysis of application domain properties.

On-demand systems support utilisation just on demand, just to the right place, just for the user, just for the content that is available and just for the application situation that happens at the moment. These systems support applications from an application domain.

**EXAMPLE 1** Let us consider three situations typical for web application systems into which development we are or have been involved:

**SeSAM parliamentary support:** The SeSAM system<sup>1</sup> aims in supporting parliamentarians in their everyday life as parliamentarian. The variety of applications changes with the change of rules how groups of parliamentarians are acting, with new rules that are settled after each new election, with new situations parliamentarians are involved into, and with discarding old approaches.

**Digicult scout services:** Scouts are collecting content for the Digicult system<sup>2</sup>. They interview

witnesses and knowledgeable people, compare their narrations with content obtained so far, annotate the new content depending on the situation, and interact between each other. These scouts are typically people that are temporarily hired.

**eGovernment 'Fachverfahren':** Government applications are supporting the work of governmental institutions though sophisticated web-based system. Classical system are 'hard-wired' in the sense that a small number of possible variants of governmental processes whereas much more variants are requested<sup>3</sup>. Companies such as DataPort are currently developing 'generic' processes.

These systems are oriented towards utilisation by users with very different background, knowledge, abilities and habits. Users do neither want to go through lengthy and annoying learning of software usage nor want to study and to study manuals. They want to continue with their everyday life and want to use software systems that are *embedded* into their application domain.

## 2.3 Manifold of Possible Solutions for the same Problem

WIS must support application solutions is a wide range. For instance, the German government initiative also targets on a proliferation and syndication of governmental processes ('Fachverfahren'). These processes typically vary from county to county, from region to region and from city to city. For instance, Schleswig-Holstein is currently heading the initiative for syndication of processes supporting sovereign rights<sup>4</sup>.

**EXAMPLE 2** The illustration example in the project is the application case *relocation* of a person, which consists of

- the change of basic relocation data including the possible removal of data on the old location,
- the change of official documents such as the passport,
- the optional change of relation enhancements such as the registration of pets, relocation of cars,
- the change of personal specific data such as family enhancements, or relationships to religious bodies,
- the change of data for additional relocation announcements such as tax, insurance changes, and
- specific additional tasks such as applications for housing allowances.

The person acts in the role of an *issuer*. We observe that *relocation* is enhanced by the profile of the issuer, by the specific tasks related to the *relocation* of the issuer, by specific laws and regulations, and

Schleswig-Holstein. It is currently extended for support on demand for visitors and employees of museums and for information scouts who are collecting folklore knowledge.

<sup>3</sup>Governmental processes are of high complexity and of super-high variability. The German book of governmental processes consists of more than 40 books (more than 7.500 pages) describing in detail any possible process and sketching possible variations. This complexity is not supported by any system. Rather cities are selecting their process bundle and are interpreting it in dependence on their (organisational) structure, their needs and opportunities.

<sup>4</sup>The project is headed by DataPort and the author acts as a consultant in the project.

<sup>1</sup>The system has been developed as an addendum to infotainment services of cities such as [www.cottbus.de](http://www.cottbus.de), regions and counties. It is currently extensively used, e.g. in Cottbus.

<sup>2</sup>The Digicult system is currently the museum portal of

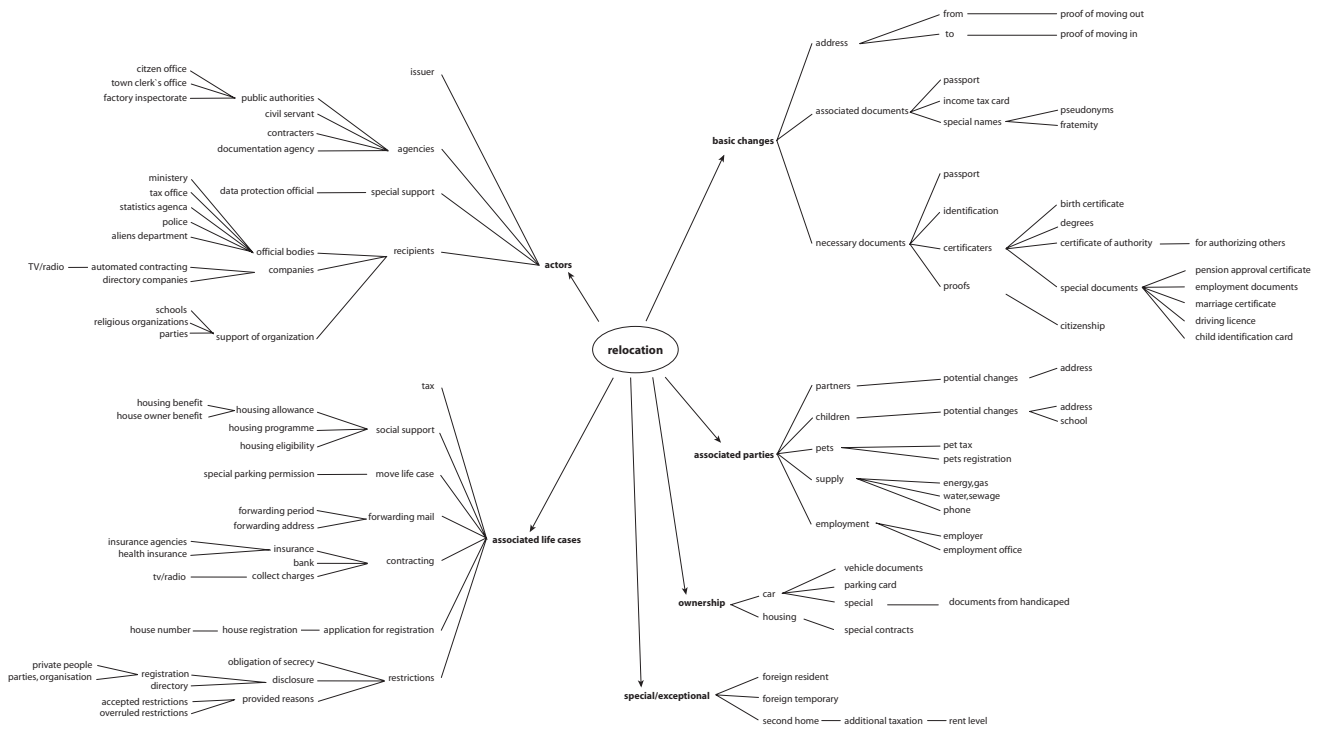


Figure 1: Facets of the relocation application case

by advanced functionality required for associating the application case with other application cases.

The application case *relocation* in Figure 1 consists of steps such as *change of address data*, *change of data for associated people*, *change of registration data* for cars, pets, etc., *change of specific data*, e.g. data for public authority responsible for aliens, *change of data for social aid*, etc. These steps are bundled together due to their relationship to one person and to one application case. The associations may be represented by adhesion of different steps, e.g. representing the association of steps by a hypergraph.

The application case in the example above has overall more than 150 variations in Germany. The general target is the same everywhere. The treatment is very different. We can group these variations in more than 2 dozen different solutions. About half of them are already supported by corresponding WIS. The integration or matching of these WIS is far from being trivial.

Moreover, the application case is associated to more than a dozen other application cases. Some of those cases are still only managed through paper exchange. This application case is one of the simplest among the application cases for proliferation and syndication of sovereign rights.

### 3 Application Cases and Problem Spaces for WIS

#### 3.1 Demands and Essentials of Application Domain Description

Application domain description is the first dimension of WIS development. (Björner 2006) describes how the entire application domain theory can be developed. A general application domain model can be given that describes any facet within the application. This theory becomes huge and superficial complex. The corresponding description might consist of hundreds of VDM pages. We claim that we do not need to

describe the entire application domain. We describe those parts that should be supported by the WIS.

The application domain is described by a number of properties, phenomena and various aspects that are desirable. A description includes designations, definitions, and refutable assertions. It can be formal or informal, sets a scope and a span, and expresses moods. Designations consist of a name, a recognition rule which purports to designate the phenomenon, and a general specification of the set of possible interpretations. Lexical definitions state the meaning of an expression in terms of other expressions. Ostensive definitions point to examples. Stipulative definitions assign a new meaning to an expression. Definitions may set bounds. Refutable assertions are claims that may be shown to be true or to be wrong.

The application domain description can be based on the following major steps:

1. Define the purpose of the application domain model
2. Select a paradigm for the application domain model
3. Determine the specific domain, specific situations, or scope of the application domain model
4. Identify an optimal process on which to develop the application domain model
5. Develop general criteria for goals, methods, and conditions
6. Develop goals for the application domain model
7. Develop methods for the application domain model
8. Identify conditions for the application domain model
9. Create a variable taxonomy or ontology for the application domain model
10. Finalize the application domain model prototype

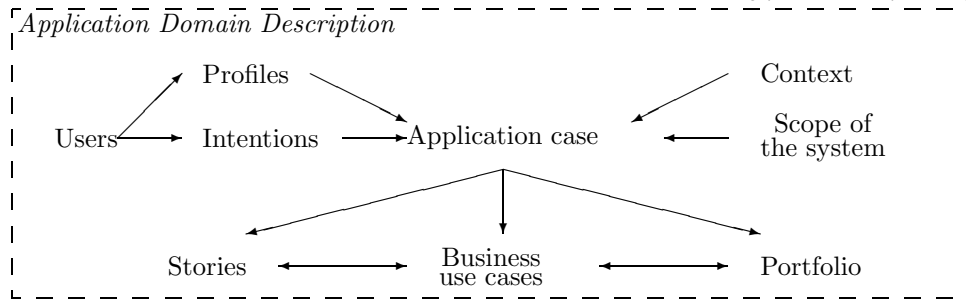


Figure 2: Elements to be specified for the application domain description

11. Formatively research the prototype application domain model
12. Revisit the goals, methods and conditions
13. Derive plans for testing the application domain model
14. Write up the application domain model
15. Derive requirements from the application domain model and improve the model

**Application cases** are the major element of the application domain model. They characterise something that occurs, happens or takes place in an application or actions or instances of occurring. They may apply to a happening without intent, volition, or plan. They may denote events, incidents, episodes, and circumstances within an application. They are associated with the context description, the characterisation of the kind of the system, and the user characterisation that consists of profiles of users and of description of their intention. The description framework depicted in Figure 2 supports an elegant and sophisticated elicitation of requirements and the derivation of the presentation system specification.

The application case study combines cognitive techniques with contextual approaches. It is extended during WIS utilisation analysis by traditional approaches. This combination allows to avoid the known disadvantages of the more specific elicitation approaches.

Users are characterised (Schewe & Thalheim 2007b) through their profile and their intentions. The profile consists of the working profile, the knowledge and abilities profile and of the psychological profile. The context specification allows to characterise the entire environment of the application case. Additionally, the solution may be restricted to certain systems and platforms.

The concept of business use cases (Maciaszek 2001, Robertson & Robertson 2005) generalises the use case concept and reflects cases or case of basic tasks in the reality. Context can be specified using the solution discussed in (Kaschek et al. 2003). The storyboard describes the story space that consists of all possible stories for the given application (Schewe & Thalheim 2005). Portfolio have been formally described in (Schewe & Thalheim 2007b) and consist of tasks and the supporting instruments that are necessary. In the sequel we decompose the application cases into business use cases and extract the portfolio. The decomposition must be invertible by a composition operation. This composition operation is the kernel of basic stories that consist of the business use cases and their story flow.

We may condense this application domain description by an abstraction of users into groups depending on their general profile and their main intentions. A group is called *actor*. Actors can be represented

by *persona*, e.g. ‘Jack-of-all-trade’ as a representation of a business man. The persona is characterized by an expressive name, their kind of profession, their purposes and intents, their technical equipment, their behaviour, skills and profile, disabilities, and specific properties such as hobbies and habits. Context and scope may be condensed to environment. We thus might use the elements in Figure 3.

### 3.2 Description of Application Cases

We may extract application cases from observations in reality, which are very useful source, whenever a WIS is going to be developed from scratch or is required to provide a ‘natural’ behaviour. In the latter case users are not required to learn the behaviour of the WIS. Instead, the user can continue using a ‘classical’ behavioural pattern.

The classical principle of the six big W and one H suffices for description of application cases: Who will be using the system? When will the system be used? Where is the information system used? What is represented in the system? Why is the system used? Which problem needs to be solved? How will the system be used?

The W<sup>6</sup>H principle can be structured into three dimensions: user dimension describing the ‘who’ and ‘why’, the flow dimension describing the ‘what’ and ‘how’, and the context dimension describing the ‘when’ and ‘where’. The entry dimension is the description of the purpose and the problems that should be solved. So we shall use the formal template:

Application case:	(application case name)
Purpose:	(purpose description)
Problems:	(list of problems)
Characterisation:	(outcome description)
Activities:	(list of user activities)
Background:	(general characterisation)
Objectives:	(list of objectives)
Application case flow:	(general description)
Milestones:	(graph of milestones)
Content consumed:	(consumed content items)
Content produced:	(produced content items)
Themes:	(class of intents)
Actors:	(list of actors involved)
Characterisation of actors:	(general profile)
	( and intension description)
Collaboration among actors:	(general collaboration )
	( description)
Context:	(general context description)
Time:	(temporality limitations)
Place:	(assignment of places)
System:	(general system context)

#### EXAMPLE 3 (Continuation of examples 1 and 2)

The system examples introduced for illustration of the challenges are based on a number of complex application cases:

- Application cases of parliamentarians are related to their official portfolio, their need in support, their context, their technical environment, and their life circumstances. The purpose of these cases ranges

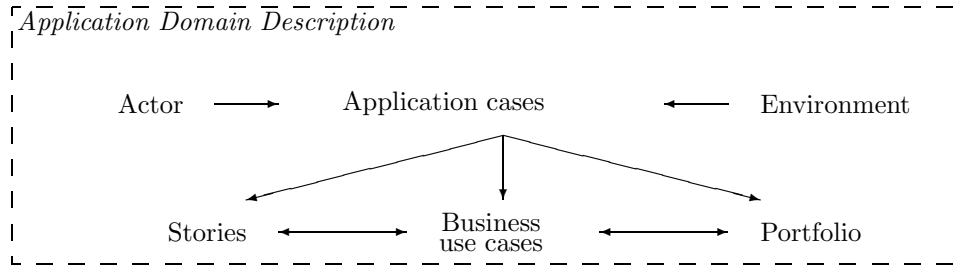


Figure 3: The abstractions of users, profile, intention, context and scope to actors and environment

from session support to support for contributions. Typically, collaborations become very complex.

- The Hallig scout in the Digicult project is gathering (folklore) information based on the content that is currently available, based on the people to be interviewed, based on their environments, evaluating their beliefs and their observations, and condensing the content produced for integration into the Digicult database.
- A typical eGovernment application is the registration support of inhabitants. It includes a number of basic stories such as change of address date, notification of contractors about the change, consideration of social etc. support, special registration for foreigners, extended registration for dependents etc.

### 3.3 Problem Space Specification

The application cases describe problems for which solutions may be envisioned. The same problem can have several solutions. Therefore, we are interested in a characterisation of problems. This characterisation is given through the problem space. This space characterises the

The **problem space** specification can be given in a natural language. We prefer a more formal approach whenever this is possible. Consider for instance the example depicted in Figure 1. Which solution is going to be supported depends on the state, on the county or on the city in Germany. The preference of one solution over others is either based on official rules and laws or on habits of the area. Therefore, we must be very flexible in our choices that are mapped to requirements. The same flexibility is also necessary for systems such as the parliamentary support system SeSAM. The last system is used in the states of Brandenburg and of Saxony and cannot be used in its current form in the state Schleswig-Holstein.

A problem in an application domain may be formally specified by four components. The following frame generalises the approach of (Polya & Polya 1945) or approaches used in AI:

The **state space** consists of the collection of all those states that are reachable from the *initial state*. Some of the states are considered to be desirable, i.e. are *goal states*. States can be modelled through languages such as ER. State may have properties such as suitability for certain purposes.

The **actions** allow to move from one state to another state under certain conditions. We may assume that the effect of the actions is observable to a certain extent by the user. User may use several actions in parallel. Actions may be blocked or enabled depending on conditions. Actions may be used at some cost.

The **goal test** determines whether a given state or state set satisfies the goals. The goal test may be

defined through a set of states or through properties. The goal test may also allow to state which quality has the state set for the problem solution.

The **problem solution controller** evaluates the actions undertaken by the user. Some solutions may be preferred over other, e.g. have less costs, or are optimal according to some optimality criterion. Controllers can be based on evaluators of the pathes from the initial state to the current state.

We assume typically that states can be observed and distinguished from each other to a certain extent by users.

### 3.4 The Algebra for Application Cases

The *application case world*  $(\mathcal{A}, \mathcal{O}, \mathcal{P})$  consists of

- basic application cases  $\mathcal{A}$ , and
- algebraic operations  $\mathcal{O}$  for computing complex cases such as combination  $\boxtimes$  of cases, abstraction  $\boxdot$  of cases by projections, quotient  $\boxdiv$  of cases, renaming  $\rho$  of cases, union  $\boxplus$  of cases, intersection  $\boxcap$  of cases, full negation  $\neg$  of cases, and minimal negation  $\rightarrow$  of cases within a given context, and
- predicates  $\mathcal{P}$  stating associations among cases such as the sub-case relation  $\preceq$ , a statement  $\boxdot$  whether cases can be potentially associated with each other, a statement  $\boxnot\dot$  whether cases cannot be potentially associated with each other, a statement  $\boxplus$  whether cases are potentially compatible with each other, and a statement  $\boxnot\plus$  whether cases are incompatible with each other.

We require that the sub-case relation is not transitively reflexive. The compatibility and incompatibility predicate are not contradicting. The potential association and its negation must not conflict.

We may use expressions defined by the operations and derived predicates:

The predicate  $\gamma := \boxdot \wedge \boxnot\plus$  is used for diverging cases.

The predicate  $\delta := \boxnot\dot \wedge \boxplus$  is used for isolated from each other cases.

The predicate  $\lambda := \boxdot \wedge \boxplus \wedge \not\preceq \wedge \not\preceq$ , and is used for homogenizable cases.

The predicate  $\oplus := \boxdot \wedge \boxplus$  is used for heterogeneous cases.

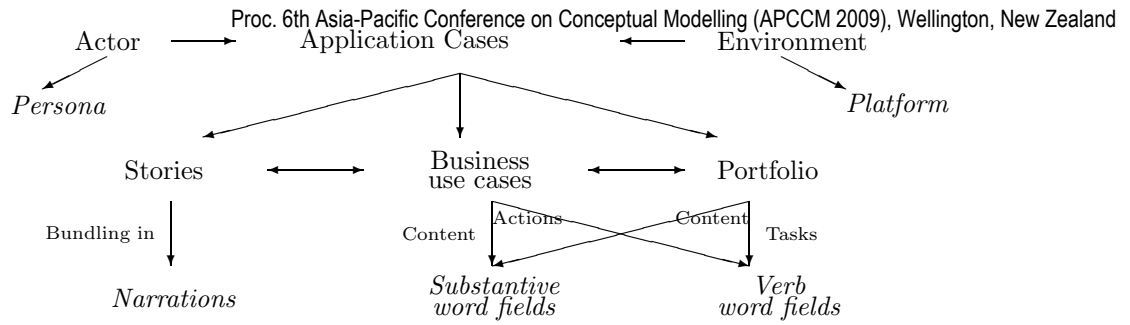


Figure 4: The natural language representation of application domain elements

Application cases are either basic or constructed cases. We use a number of representations for visualisation of the application cases within the application domain:

*Application case maps* are graphs consisting of nodes representing application cases and edges representing the sub-case relation and the combination of cases to complex cases.

*Application case hierarchies* are forests representing cases and their sub-case relation.

*Full application case maps* are graphs consisting of nodes representing application cases and edges representing the sub-case relation and the combination, abstraction, union, intersection, quotient, full negation and minimum negation of cases if these cases are used within the application.

These operations form the basis for the mapping of application cases to business use cases, to stories and to portfolio. The extraction of business use case from application cases is supported by five operations that are easing the task of efficiently acquiring relevant requirement prescriptions and of documenting them:

1. Application case *projection* narrows or scopes the application case to those parts (entities or concepts, axioms or invariants relating entities, functions, events, and behaviours) that are of concern for the business use cases.
2. Application case *instantiation* lifts the general cases to those that are of interest within the solution and instantiates variables by values which are fixed for the given system.
3. Application case *determination* is used for selecting those traces or solutions to the problem under inspection that are the most perspective or best fitting for the system envisioned. The determination typically results in a small number of scenarios for the application cases that are going to be supported.
4. Application case *extension* is used for adding those facets that are not given by the application case but are given by the environment or by the platforms which might be chosen or that might be used for simplification or support of the application case (e.g., additional data, auxiliary functionality).
5. Application case are often associated, adjacent, interact or fit with each other. Application case *join* is used to combine application cases into more complex and combined cases that describe a complex solution.

Business use cases prescribe the “operating” part of the application cases for actors and the supporting system. Therefore, we determine which part of the application cases is going to be left out, which parts are emulated, and which solution is going to be preferred. The stories combine these business use cases to flows of activities. The combination is determined though the application of the five operations.

The application of these operations also allows to extract which subcases, which functionality, which events and which behaviour is *shared* among the business use cases. These shared facilities provide cross-cutting concerns among all business use cases and the exchange activities or scenes in the stories. They also hint on possible architectures of information systems and on separation into candidate components. For instance, entity sharing describe which information flow and development can be observed in the application. Functionality may be either shared by several components or may be a part of the user interface.

The profiles are specifying tasks and obligations or permissions of users of the system. These profiles can be extracted from the application cases by application of these operations. The interaction between the users and the system can directly be mapped to message or life sequence charts. Additionally, we can extract quality properties (Jaakkola & Thalheim 2005) such as faithfulness, didactic behaviour, pedagogic surveyability, physiological widgets, psychological behaviour and user-friendliness for the *human-computer interface*.

This information is later enhanced by computer requirements such as performance, dependability, maintenance, platform, and documentation requirements.

## 4 Transforming Application Cases to Requirements

### 4.1 Mapping Application Cases to Stories, Portfolio and Business Use Cases

Application domain engineering is based on a tight collaboration with the customer, with the user, and with the owners and planers. We may use diagram techniques at that level. In this case we assume that partners in the development process are well informed and well educated. Another representation technique is the sophisticated use of natural language utterances. The sophistication is based on a theory of word fields (called capsules in (Berztiss 2001)). Word fields consist of syntactical and semantical representations of words similar to approaches developed for WordNet (WordNet 2007). Additionally a pragmatical dimension can be added to word fields. *Verb word fields* can be classified into 10 word categories (Kunze 1992). A similar classification has been developed for *substantive word fields* in the RADD project (Albrecht, Altus, Buchholz, Cyriaks, Düsterhöft, Lewerenz, Mehlan, Steeg, Schewe & Thalheim 1998). These theories allow to use substan-

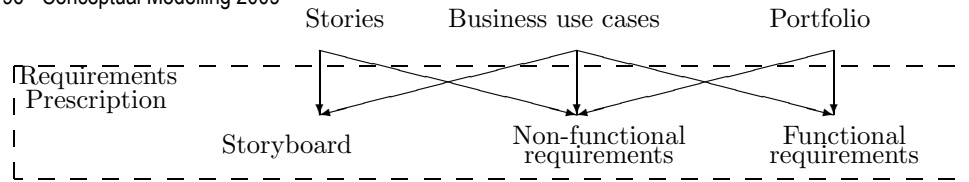


Figure 5: The use of application domain information for requirements elicitation and analysis

tive and verb word fields for the verbal description of the application. We thus support the verbal descriptions for description of application cases. We also support *narrations* since verbal description are often given as real-life application *stories* describing real application cases.

The mapping is based on the first three phases of the C3S3P framework (Krogstie & Jorgensen 2004): concept study, scaffolding, scoping, solution modelling, platform integration, piloting in real projects and performance monitoring and management. Concept study uses word fields and extracts the concept. The scaffolding phase is used for analysis of the current solution to the application problem and for envisioning other solutions. Scoping focuses on creation of executable pilots supporting the application case. This knowledge is used for identification and consolidation of requirements for structuring and functionality of the (information) system, for platform plans, methodology choice and sketching these requirements.

Word fields allow to derive generic functions and generic content (Bienemann, Schewe & Thalheim 2006). These functions and content provide the basis for a specification of tasks and business use case (Robertson & Robertson 2005). Tasks are the main element of *portfolio*. The action components of business use cases are represented by verbs and the content component is characterised by substantives. Therefore business use cases and portfolio are templates that are used for representation of chunks of word fields. Finally, the environment is often given through a hint to *platforms* intended or requested.

#### 4.2 Mapping Application Cases to Requirements

The application domain description can be used for requirements gathering. Application cases describe all parts of the application domain that are relevant for the application. The application cases are subject to peer reviews. It starts with a feasibility and completeness study. Next we check whether all activities are represented. Additionally, we check whether application cases are truly generic in the sense that they do not go into detail that may apply to the present application alone. Finally, we check whether there are any special cases additional to those that have already listed in the application cases. Applying this check results in a set of tasks that are integrated into the application portfolio, in a set of business use cases and in a set of stories.

We can now map the three results of application domain description to properties typically used for requirements prescription. Software engineering has divided properties into functional and non-functional properties, restrictions and pseudo-properties. This separation can be understood as a separation into essential properties and non-essential ones. Functional and non-functional properties require prescription of main data structures, of the main functionality, of control facilities, and of collaboration of components of a system envisioned.

If we separate the information system from the presentation system then this separation leads to a

far more natural separation into information system requirements and presentation systems requirements. The system perspective considers properties such as performance, efficiency, maintainability, portability, and other classical functional requirements. Typical presentation system requirements are usability, reliability, and requirements oriented to high quality in use, e.g., effectiveness, productivity, safety, privacy, and satisfaction. Safety and security are also considered to be restrictions since they specify undesired behaviour of systems. Pseudo-properties are concerned with technological decisions such as language, middleware, operating system or are imposed by the user environment, the channel to be used, or the variety of client systems.

Figure 5 depicts the mappings of elements of the application domain description to elements used in requirements engineering. The storyboard can be understood as a step-wise or scene-wise prescription of how a particular actor interacts with the system.

As requirement acquisition and elicitation progresses there is likely a shift in the preferred representation from textual and formal to graphical. A number of approaches have been developed for graphical representation such as storyboarding approaches (Schewe & Thalheim 2005), quality-driven design methodologies (Jaakkola & Thalheim 2005), and UML diagrams (use case, class, sequence, statechart, and activity diagrams). The storyboard is going to be mapped to a number of activity diagrams.

#### 5 Conclusion

This paper enhances application domain modelling by explicit consideration of application cases. These application cases can be mapped to requirements. The extraction of requirements is based on the C3S3P framework. The approach reported has intentionally been applied in about half dozen of our website development projects. We realised that requirements engineering is to system-focused and that we need a formal representation of the elements at the strategic layer. Our description of this layer led to the development of a theory of application domain description, to a number of formal templates that can be used for description, and to mappings to the elements of the requirements prescription layer. This paper summarises those ideas.

Application cases are typically already based on proposals for solutions or on a number of solutions for which one of them is going to be preferred at the moment. For instance, eGovernment application cases must correspond to the expectations of users and their needs within life situations. These life situations might be supported by application cases. Due to the variety of users, to the variety of utilisation of the system, to the variety of real usage, we extend the application domain description by life cases (Schewe & Thalheim 2007a). Life cases reflect the life situations in the complexity of everyday life. Life cases are going to be decomposed, segmented and mapped to application cases.

## References

- Albrecht, M., Altus, M., Buchholz, E., Cyriaks, H., Düsterhöft, A., Lewerenz, J., Mehlan, H., Steeg, M., Schewe, K.-D. & Thalheim, B. (1998), 'RADD - Rapid application and database development. Readings - Main papers published in the RADD project', CAU Kiel, Department of Computer Science, <http://www.is.informatik.uni-kiel.de/~thalheim/indeerm.htm>.
- Anh, D. N. & Moore, R. (1996), Formal modeling of large domains, in 'APSEC', IEEE Computer Society, pp. 246–.
- Bertziss, A. (2001), Requirements engineering, in 'Handbook of Software Engineering and Knowledge Engineering', Vol. I, World Scientific Pub. Co., pp. 59–70.
- Bienemann, A., Schewe, K.-D. & Thalheim, B. (2006), Towards a theory of genericity based on government and binding, in 'Proc. ER'06, LNCS 4215', Springer, pp. 311–324.
- Bjørner, D. (2006), *Software Engineering 3: Domains, requirements, and software design*, Springer, Berlin.
- Boehm, B. (2006), A view of 20th and 21st century software engineering, in 'Proc. ICSE'06', ACM Press, pp. 12–29.
- Darimont, R. & van Lamsweerde, A. (1996), Formal refinement patterns for goal-driven requirements elaboration, in 'SIGSOFT FSE', pp. 179–190.
- Epner, M. (2000), Poor project management number-one problem of outsourced e-projects, Research briefs, Cutter Consortium.
- Gunter, C. A., Gunter, E. L., Jackson, M. & Zave, P. (2000), 'A reference model for requirements and specifications', *IEEE Software* **17**(3), 37–43.
- Hansen, M. R. & Hung, D. V. (2007), A theory of duration calculus with application, in 'Domain Modeling and the Duration Calculus', pp. 119–176.
- Jaakkola, H. & Thalheim, B. (2005), Software quality and life cycles, in 'ADBIS'05', Springer, Tallinn, pp. 208–220.
- Jackson, E. K. & Sztipanovits, J. (2006), Towards a formal foundation for domain specific modeling languages, in 'EMSOFT', ACM, pp. 53–62.
- Kaschek, R., Schewe, K.-D., Thalheim, B. & Zhang, L. (2003), Integrating context in conceptual modelling for web information systems, web services, e-business, and the semantic web, in 'WES 2003', LNCS 3095, Springer, pp. 77–88.
- Krogstie, J. & Jorgensen, H. (2004), Interactive models for supporting networked organisations, in 'CAISE'04', LNCS, Springer, Berlin, pp. 1–14.
- Kunze, J. (1992), Generating verb fields, in 'Proc. KONVENS', Informatik Aktuell, Springer, pp. 268–277. in German.
- Lambrix, P. & Padgham, L. (1996), A description logic for composite objects for domain modeling in an agent-oriented application, in 'Description Logics', Vol. WS-96-05 of *AAAI Technical Report*, AAAI Press, pp. 146–149.
- Lewerenz, J. (2000), Human-computer interaction in heterogeneous and dynamic environments: A framework for its conceptual modelling and automatic customization, PhD thesis, Brandenburg University of Technology at Cottbus, Faculty Mathematics, Natural Sciences and Computer Science.
- Lowe, D. (2003), 'Web system requirements: an overview', *Requirements Engineering* **8**(2), 102–113.
- Maciaszek, L. (2001), *Requirements analysis and design*, Addison-Wesley, Harlow, Essex.
- Missikoff, M. & Schiappelli, F. (2005), A method for ontology modeling in the business domain, in 'EMOI-INTEROP', CEUR Workshop Proceedings, CEUR-WS.org.
- O'Neill, E. & Johnson, P. (2004), Participatory task modelling: users and developers modelling users' tasks and domains, in 'TAMODIA', ACM, pp. 67–74.
- Polya, G. & Polya, G. (1945), *How to solve it: A new aspect of mathematical method*, Princeton University Press, Princeton.
- Robertson, S. & Robertson, J. (2005), *Requirements-led project management*, Pearson, Boston.
- Rolland, C. (2006), From conceptual modeling to requirements engineering, in 'Proc. ER'06', LNCS 4215, Springer, Berlin, pp. 5–11.
- Schewe, K.-D. & Thalheim, B. (2005), 'Conceptual modelling of web information systems', *Data and Knowledge Engineering* **54**, 147–188.
- Schewe, K.-D. & Thalheim, B. (2007a), Life cases: An approach to address pragmatics in the design of web information systems, in J. Filipe, J. Cordeiro, B. Encarnacao & V. Pedrosa, eds, 'Proc. WebIST', Vol. II (WIA), pp. 5–12.
- Schewe, K.-D. & Thalheim, B. (2007b), 'Pragmatics of storyboarding for web information systems: Usage analysis', *Int. Journal Web and Grid Services* **3**(2), 128–169.
- Schewe, K.-D., Thalheim, B. & Tretjakow, A. (2006), Formalization of user preferences, obligations, and rights, in R. Kaschek, ed., 'Perspectives of Intelligent Systems Assistents, PISA'05', IDEA.
- Thalheim, B. (2000a), *Entity-relationship modeling - Foundations of database technology*, Springer, Berlin.
- Thalheim, B. (2000b), 'Readings in fundamentals of interaction in information systems', Reprint, BTU-Cottbus, accessible through <http://www.is.informatik.uni-kiel.de/~thalheim>, Collection of papers by C. Binder, W. Clauß, A. Düsterhöft, T. Feyer, T. Gutacker, B. Heinze, J. Lewerenz, M. Roll, B. Schewe, K.-D. Schewe, K. Seelig, S. Srinivasa, B. Thalheim.
- Wieringa, R. (2003), *Design methods for reactive systems: Yourdan, Statemate, and the UML*, Morgan Kaufmann, Amsterdam.
- WordNet (2007), 'Antonym finder and synonym thesaurus - Synonyms and definitions for english words', <http://www.synonym.com/>.





# Conceptual Business Document Modeling using UN/CEFACT's Core Components

Philipp Liegl

Institute of Software Technology and Interactive Systems  
Business Informatics Group  
Vienna University of Technology,  
Favoritenstrasse 9-11/188, A-1040 Vienna,  
Email: liegl@big.tuwien.ac.at

## Abstract

Before two businesses can engage in a business-to-business process an agreement about the process execution order and the business documents exchanged in the collaborative process must be found. Although several initiatives from different industries have started standardization initiatives for business documents a set of shortcomings still remain. (1) The different standards do not have a common semantic basis causing inter-operability problems between them. (2) Furthermore, they try to include every possible element any industry might need into the business document standard. (3) Moreover, most of the standards are transfer syntax specific and do not provide a conceptual representation mechanism. In this article a new concept for the standardization of business documents called UN/CEFACT's Core Components Technical Specification is presented which solves these shortcomings. Using Core Components the business document modeler can unambiguously define documents with a common semantic basis on a conceptual level. In order to allow for a better integration into UML modeling tools we introduce the UML Profile for Core Components. With the UML based core component model and an XML schema generator the modeler can derive XML schema artifacts from the conceptual model.

**Keywords:** Business Document Modeling, Business Document Meta Modeling, UN/CEFACT's Core Components

## 1 Introduction

If two businesses want to get involved in an automated B2B process they first have to agree upon a common choreography uniquely defining the exchange order of the different business documents. Several approaches for the standardization of a business choreography exist nowadays e.g. (UN/CEFACT 2006b), (Jung et al. 2004) or (Rinderle et al. 2006). While a process choreography describes the exchange order of business documents in detail, little to nothing is said about the harmonization of business documents which are being exchanged. One of the best known approaches for the standardization of exchanged data is UN/EDIFACT (Berge 1994) maintained by the United Nations Center for Trade Facilitation and Electronic Business (UN/CEFACT). The UN/EDIFACT standard provides a set of syntax rules used

to structure business document data. The document format uses designated symbols and letter codes as delimiters between the different data fields.

Since XML was introduced in 1996 (W3C 2006) its popularity has constantly increased due to its versatility, flexibility and easy applicability. An additional boost has been brought by the introduction of Web Services and their related technologies such as WSDL (W3C 2007b), SOAP (W3C 2007a) and UDDI (OASIS 2007). In particular in the context of Web Services the clear and precise definition of a business document is of importance. Usually interfaces defined by WSDL import the appropriate XML schema defining the type of business document the interface accepts.

Given the popularity of XML as the representation format of choice for data, several initiatives have been started in order to standardize exchanged data using XML. An overview of different XML based standards for describing data and business documents is given by (Li 2000). However, the transition from a delimiter based approach such as UN/EDIFACT to an XML based solution did not solve the interoperability problems between business documents. We address the following shortcomings in regard to business document standardization:

(1) *Standard incompatibilities.* Due to the multiple initiatives which have been started, several XML based business document representation mechanisms exist, which are competing against each other. Eventually this results in large incompatibilities between the different standards.

(2) *All-in-one approach.* Furthermore a lot of standards aim at the integration of every possible element into a standardized business document, resulting in a strong document overhead. E.g. a cross industry invoice which should be applicable in any industry context has to include every possible element any of the different industries might need. Whereas for instance `number of nights per person` is critical in a tourism context this attribute is rather unlikely to be needed in an oil industry context. However, in order to be cross-industry compatible every possible element has to be included in the standardized invoice. A partial solution is given by so called message implementation guidelines (MIG), cutting down a standard to a set of agreed elements which are used between a well defined subset of business partners. However, an extensive use of different MIGs undermines the idea of a holistic standard and results in a multitude of different and incompatible standard subsets.

(3) *Lack of conceptual document description.* Standards such as UN/EDIFACT or XML based solutions for business documents are tightly bound to the implementation syntax. Often the document semantics are defined on the logical level (e.g. XML schema) instead of being defined on an higher, conceptual level. Changes in the transfer syntax therefore result in re-engineering tasks for the standard,

making it inflexible to future adaptations. In regard to implementation tasks, a logical level business document model is difficult to communicate between software developers and other stakeholders in the implementation process.

Knowing these limitations UN/CEFACT started the development of the so called Core Components Technical Specification (UN/CEFACT 2003). The idea is to develop an ontology of re-usable building blocks for business documents. Using these building blocks, a shared library is built from which modelers can retrieve artifacts in order to assemble a business document.

The development of the core components standard started in the late nineties as part of the ebXML (OASIS 2001b) initiative. The main goal of ebXML was to provide a framework allowing potential business partners to engage in B2B processes in an interoperable, secure and consistent manner. One part of the ebXML technology stack were so called core components, used to uniquely define the exchanged data between two enterprises. UN/CEFACT's Technologies and Methodologies group, of which we are members of, continued the development of core components and today the standard is known as the Core Components Technical Specification (CCTS). The most recent version of the standard is 2.01 (UN/CEFACT 2003) with the development of version 3.0 (UN/CEFACT 2008) currently going on.

In this paper we present the UN/CEFACT Core Component Technical Specification and a UML profile which is built on the technical specification. It will be shown how core components can help to overcome the four limitations mentioned above and how XML schema specifications can automatically be derived from a core component model. The remainder of this article is structured as follows: section 2 gives an overview about related work in the field of conceptual XML schema modeling and section 3 introduces the core components approach. The UML Profile for Core Components is outlined in section 4 and an example is given in section 5. The derivation of XML schema artifacts from core components is shown in section 6 and section 7 concludes the paper and gives an overview about future work.

## 2 Background - Related Work

A general introduction into business document modeling is given by (Glushko & McGrath 2005). Glushko and McGrath provide a thorough and holistic overview about current approaches for business document modeling, document model interoperability and integration into business processes.

The conceptual modeling of data has existed for a while and forms an integral part of data engineering. One of the most important methodologies for data modeling is the so called entity relationship model (Chen 1976) used to design a relational database model. The entity relationship model (ER) provides its own modeling methodology consisting of entities, attributes belonging to entities, and relationships between the different entities. A database modeler uses the entity relationship model to derive the appropriate data definition language (DDL) artifacts for creating the database model. The main goal of a database is to reliably store and retrieve information from it. If a hierarchical business document is stored in a database it is first broken up into the relational model and then stored in the appropriate database tables. Retrieving the document means querying the database for the relevant information parts and re-assembling the business document. The relational database model has therefore less context than the

business document model because its main goal is to store and retrieve pure information and avoid inconsistencies.

Both, the business document model and the relational database model serve their own purpose. On the one hand the relational model focuses on a multitude of business documents and not on a single instance, since its goal is the consistent storage of normalized data in the large scale. Thus, the avoidance of large scale data redundancy is an integral part of the relational model. In some cases business document models must deliberately allow data redundancy due to the requirements of a given business case. As an example an `invoice bundle` is taken which groups `invoices` of the same enterprise. The left hand side of figure 1 shows an `invoice bundle` containing multiple instances of `invoices` numbered 1, 2 and 3. Each instance of the `invoice` contains the same `tax number` (3) although all `invoices` are of the same enterprise and hence the `tax number` is the same for each `invoice`. Since the `invoice` itself has to be a self contained document, the `tax number` cannot be stored in the embracing `invoice bundle` but must be part of the `invoice`. In comparison the right hand side of figure 1 shows the relational model for the same scenario. An `invoice bundle` groups multiple `invoices` of the same enterprise. Since the data is stored normalized, the `tax number` is part of the `invoice bundle` and not of the `invoice`. It follows, that the relational model is not well suited for business document modeling and a more appropriate approach must be found.

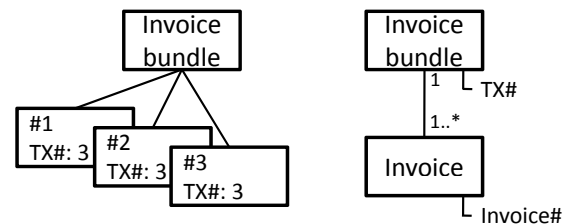


Figure 1: Business document model vs. relational model

Related work in the field of conceptual XML schema modeling concentrates on two main fields. On the one side research is conducted in the area of forward engineering e.g. deriving XML schema artifacts from conceptual models such as UML. On the other side a lot of effort is invested in the reverse engineering approach e.g. generating conceptual models such as UML class diagrams from XML schema artifacts. An overview on research on the reverse engineering of XML schemes to conceptual models such as UML class diagrams is given in (Yu & Steele 2005). The authors examine different reverse engineering approaches and assess their applicability. Although several techniques for a forward engineering from conceptual models to XML representations exist today, only a few solutions are available for transformations in the opposite direction. The generation of UML models out of XML schema data proves to be difficult since not all of the features of an XML schema can be represented in a UML diagram. E.g. UML does not support the concept of inheritance by restriction as XML schema does. Another open issue is the ordering of attributes which is important in an XML schema but not supported by UML class diagrams. A thorough solution for a reverse engineering approach is presented by (Salim et al. 2004). Using a set of transformation rules for the corresponding XML schema elements, the authors present appropriate representation solutions in UML. However, the authors do

not address the representation of an `<xs:restriction>` which cannot be demonstrated in UML.

In contrast to the reverse engineering of conceptual UML models from XML schema several approaches exist for the forward engineering approach. One of the first research propositions for the representation of XML using UML has been presented by (Skogan 1999). Built upon these research results (Combi & Oliboni 2006) introduce the so called UXS model (UML & XML Schema) based on UML. UXS is a methodology for designing XML documents using a set of graphical elements which correspond to the appropriate XML schema components. Furthermore a translation mechanism is introduced allowing the generation of XML schema artifacts according to the three well known patterns "Russian Doll", "Salami Slice", and "Venetian Blind" (Malik 2003).

Although several approaches for the conceptual modeling of XML schema exist, only a few of them consider the semantic data modeling aspects. A mutation analysis model used to verify the general semantic correctness of an XML schema is introduced by (Li & Miller 2005). Using their approach (Li & Miller 2005) compare different XML schema validators in regard to their effectiveness in finding semantic errors within XML schemas. A formalization for a data modeling approach is introduced by (Mani et al. 2001), also taking into account the semantic dependencies between the different elements within an XML schema. The introduced methodology called *XGrammar* allows for a precise definition of features necessary for data modeling such as n-ary relationships, generalizations etc. The application of the Active XML Schema approach for the semantic enrichment of XML schema documents is discussed in (Bernauer et al. 2003). In their paper the authors examine the trade-off between semantic enrichment of XML schema and its interoperability.

A similar approach to UN/CEFACT's Core Components is pursued by OASIS and has become known as the Universal Business Language (OASIS 2006) (UBL). UBL is a standard for XML document formats based on UN/CEFACT's core components and provides a mapping of the syntax neutral core components to real XML constructs. The initiative follows a similar approach as the naming and design rules (UN/CEFACT 2006c) (NDR) provided by UN/CEFACT. In order to overcome the redundancies in regard to standardization a merger of the UBL initiative with the core components initiative of UN/CEFACT has been decided during the UN/CEFACT Forum meeting 2007 in Stockholm.

In regard to domain specific business standards several initiatives have been started in recent years. RosettaNet Business Documents (RosettaNet 2006) is an initiative of the electronic components and telecommunications industry. In the insurance domain the ACORD (ACORD 2007) standard plays a significant role and CIDX (CIDX 2007) is pursuing document standardization for the chemical industry. Other initiatives include SWIFT (SWIFT 2007) from the finance industry, HL7 (HL7 2007) from the health care industry, Papinet (papiNet 2007) from the forest and paper industry, and PIDX (PIDX 2007) from the oil and gas industry.

As outlined in this section several approaches to the conceptual modeling of business documents and XML and the forward and reverse engineering thereof exist. Although applicable to the general purpose of XML modeling, the different approaches do not consider the business semantics and business requirements necessary for business document modeling. The following section will introduce the core components standard as a methodology of choice for the concise modeling of business documents exchanged in

inter-organizational business processes.

### 3 UN/CEFACT's Core Components

#### 3.1 The core component meta model

*Core Components* form the central building blocks of the Core Components Technical Specification (CCTS) (UN/CEFACT 2003). By definition *core components* are syntax and platform independent and the standardization of *core components* is done using regular spreadsheets. If a *core component* is used in a certain business context it becomes a so called *business information entity*. The two packages in the CCTS meta model in figure 2 show the two fundamental concepts of a core and a business context.

As shown on the left hand side of figure 2 the core components standard distinguishes between three different types of core components: *aggregate core components* (ACC), *basic core components* (BCC) and *association core components* (ASCC). An *aggregate core component* forms a self contained entity which consists of several *basic core components*. For example an *address* would be an *aggregate core component* whereas the different details of an *address* such as *street*, *postal code* etc. would be *basic core components*. In order to build relationships between different *aggregate core components* the concept of so called *association core components* is used. An *association core component* may for instance relate the two *aggregate core components* *person* and *address*. A *basic core component* such as *postal code* in *address* has a certain type - a so called *core data type*. *Core data types* are based on primitive types referred to as *Core Component Types*. *Core Component Types* are e.g. *Integer*, *String*.

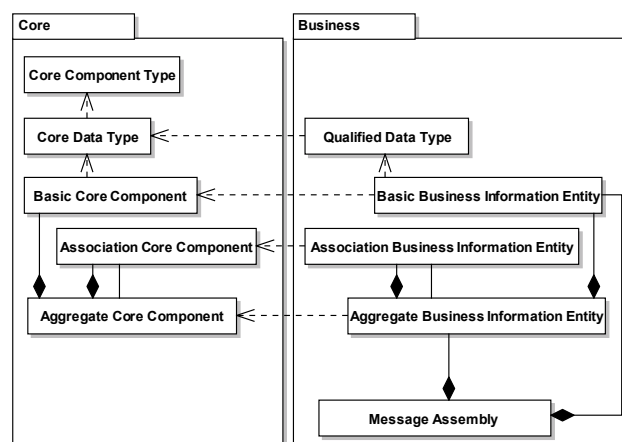


Figure 2: CCTS meta model

*Core components* are standardized by UN/CEFACT and are independent of a specific industry context or business domain. In order to derive a business document solution for a specific business context the business modeler takes a generic *core component* and tailors it to the specific need of a business domain. Hence *core components* serve as the generic basis for industry specific document formats.

If core components are put into a specific business context they become so called *business information entities*. As shown on the right hand side of figure 2 the core components standard differentiates between three different types of *business information entities*: *aggregate business information entities* (ABIE), *basic business information entities* (BBIE) and *association business information entities* (ASBIE). Similar to the concept of *core components* an *aggregate business information entity* forms a self contained block which



consists of several *basic business information entities*. For example a *cargo box* would be an *aggregate business information entity* whereas the different details of the *cargo box* such as *height* or *weight* would be *basic business information entities*. A *basic business information entity* has a certain type - a so called *qualified data type*. A *qualified data type* is based on a *core data type*. Similar to the relationship between a generic *core component* and a business context specific *business information entity*, a business specific *qualified data type* is based on a generic *core data type*.

Using *association business information entities* the modeler builds relationships between different *aggregate business information entities*. An *association business information entity* could for instance relate the two *aggregate business information entities* *cargo box* and *cargo good* in order to indicate the content of the *cargo box*.

The concept of a *message assembly* as shown on the lower right hand side of figure 2 is used to aggregate different *business information entities* together, forming the final business document.

The relationship between the core and the business context is denoted by the four dependencies in figure 2. As already mentioned a *qualified data type* is based on a *core data type*. Likewise a *basic business information entity* is based on a *basic core component* and an *association business information entity* is based on an *association core component*. Finally an *aggregate business information entity* is based on an *aggregate core component*. Since the specific dependencies between *core components* and *business information entities* are rather difficult to conceive on the meta level a simple example will be used in order to elaborate the basic concepts of *core components* and *business information entities*.

### 3.2 A simple core component example

In order to allow for a better understanding of the core components methodology we already use the UML based notation in this section as specified in the UML Profile for Core Components (UN/CEFACT 2006a). In order to give the reader a first impression about core components figure 3 shows a simple core component example. *Aggregate core components* are modeled using UML classes and *basic core components* are shown as attributes thereof. *Association core components* are denoted using compositions between UML classes.

On the left hand side two *aggregate core components* are shown - *invoice* and *line item*. In this example an *invoice* consists of three *basic core components*: an *invoice number*, a *country identifier*, and a *description*. In a real world example the *invoice* would contain more *basic core components* - for presentation purposes however they have been left out as indicated in figure 3. Furthermore an *invoice* has an *association core component* named *item* leading to the *aggregate core component* *line item*. *Line item* in turn also has three *basic core components*: *identifier*, *net price* and *description*. Due to space limitations only three *basic core components* are shown.

The two example core components shown on the left hand side of figure 3 are standardized independently of any business context by UN/CEFACT. These generic *core components* are used to derive industry specific business document formats. In our case an example from the United States tourism industry is used. On the right hand side of figure 3 the business context with two *aggregate business information entities* *US invoice* and *US line item* is shown.

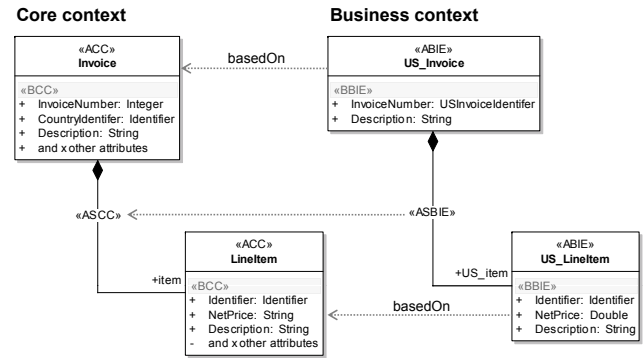


Figure 3: Dependency between core and business context

US invoice has two *basic business information entities* namely *invoice number* and *description*. Furthermore it has an *association business information entity* named *US item* connecting the two *aggregate business information entities* *US invoice* and *US line item*. The *aggregate business information entity* *US line item* has three different *basic business information entities*: *identifier*, *net price* and *description*.

The relationship between *core components* and *business information entities* is as follows. If a modeler constructs a business document for a certain business context or industry he first searches the generic core component library for an appropriate document representation of his business case. After having found an appropriate *core component*, the modeler restricts the *core component* to the specific needs of the business domain. Thereby the *core component* becomes a *business information entity*. A *business information entity* is always derived from a *core component* by restriction. Hence, a *business information entity* cannot contain any attributes which are not specified in the underlying *core component*. As shown in figure 3 the *aggregate business information entity* *US invoice* contains only two attributes namely *invoice number* and *description*, because it restricts the generic *aggregate core component* *invoice* to those types needed in the specific business context. The same applies to the *aggregate business information entity* *US line item* and its underlying *aggregate core component* *line item*. This specific relationship between *aggregate core components* and *aggregate business information entities* is denoted by the *basedOn* dependency in figure 3.

Likewise a *basic business information entity* is based on a *basic core component*. No *basedOn* dependencies between *basic business information entities* and *basic core components* are shown in figure 3, since the *aggregate core components* and *aggregate business information entities* containing the *basic core components* and *basic business information entities* are already connected using a *basedOn* dependency. The same *basedOn* relationship applies to *association core components* and *association business information entities* as shown in figure 3.

As business context for the *business information entities* in figure 3 we assume an example from the tourism industry. In order to help to distinguish *core components* from *business information entities* the concept of *qualifiers* is used for *aggregate business information entities* and *association business information entities*. The qualifier used in figure 3 is *us\_*, indicating a fictional invoice from the United States tourism industry. A qualifier can be chosen arbitrarily by the business document modeler and does not need to comply with any constraints.

*Basic business information entities* and *basic core*

components are of a certain type. The *basic core component invoice number* of the *aggregate core component invoice* is of type *Integer*. We refer to data types of *basic core components* as *core data types* (CDT). The *basic business information entity invoice number* of the *aggregate business information entity US invoice* is of type *US invoice identifier*. We refer to data types of *basic business information entities* as *qualified data types* (QDT). However, please note that a *basic business information entity* can also use a *core data type* if necessary. The example in section 5 further elaborates this necessity.

As indicated on the left hand side of figure 3 the *basic core component invoice number* of the *aggregate core component invoice* is of type *integer*. However, the *basic business information entity invoice number* of the *aggregate business information entity US invoice* as shown on the right hand side of figure 3 is of type *US invoice number*. Hence, the modeler has the possibility to restrict the data type of a *basic business information entity* to the specific needs of a certain business context. Not shown in figure 3 is the fact, that the modeler can also restrict which *association core components* are becoming *association business information entities*. It follows, that an *aggregate business information entity* does not necessarily has to have all associations like the underlying *aggregate core component*.

This section has given an overview about the fundamental principles of the *core components* standard. Since *core components* are standardized independently of any implementation platform or technology a representation mechanism for *core components* had to be found. The next section will introduce the UML Profile for Core Components.

#### 4 UPCC - A UML Profile for Core Components

Recent years have shown an increasing trend toward the usage of UML in the area of business process modeling and business document modeling. Several tool vendors have developed UML modeling tools supporting the most recent version of the UML meta model (OMG 2007). In order to allow for an easy integration of the core components methodology into such tools, a representation mechanism for *core components* using the UML syntax had to be found.

As indicated in the previous section, *core components* are standardized independent of any business context or specific syntax using regular spreadsheets. The core components technical specification defines its own MOF-like meta model as shown in figure 2. However this MOF-like meta model is entirely independent of the UML meta model. Therefore no unique representation mechanism for *core components* in UML is given. If every modeler defines its own UML representation mechanism the different *core component* models are unlikely to match. Furthermore, the storage and retrieval of *core component* artifacts in a central and public accessible registry is impossible since no commonly agreed representation format for *core components* is available. This represents a strong contradiction to the initial purpose of *core components*: cross-industry alignment and reusability of business documents and business information.

Therefore a unique representation mechanism for *core components* in UML is necessary. The authors of this article, together with other contributors, have submitted a UML representation format for *core components* to UN/CEFACT. Since then this proposal has become known as the UML Profile for Core Components (UPCC) standard (UN/CEFACT 2006a). A

UML profile customizes the UML meta model to the specific needs of a certain application scenario. Using stereotypes, tagged values, and OCL constraints the generic UML meta model is tailored to the specific needs of business document modeling. Figure 4 gives an overview of the different stereotypes used in the UML Profile for Core Components. Since the full names of the different stereotypes are rather long, abbreviations have been used. Stereotypes representing modeling artifacts are presented using a black background. In UPCC modeling artifacts are structured using the concept of packages. In the meta-model these packages are shown with a white background.

The UPCC standard aims at a precise and unambiguous representation of *core components* in UML. Where possible, native concepts of UML have been used to depict *core component* principles. The very basic stereotype is a *primitive type* (PRIM). A PRIM is used to express basic types such as *String*, *Integer* and is very similar to the UML concept of a *type*. The *core component* standard defines six *primitive types* partly overlapping with the types defined in UML. In order to restrict a *primitive type* to a specific set of values an *enumeration* (ENUM) is used. Thereby the modeler can restrict a *primitive type* to a specific set of valid values e.g. ISO 3166 (ISO 2007) for valid country codes. In UPCC an *enumeration* is represented using the UML concept of an *enumeration*.

In contrast to an *enumeration* or a *primitive type* a *core data type* (CDT) can express a more meaningful type. A *core data type* is modeled using classes and consists of multiple attributes. Thereof exactly one attribute is stereotyped as *content component* (CON) and multiple attributes can be stereotyped as *supplementary components* (SUP). The *content component* represents an atomic value and *supplementary components* are used to provide meta information about the *content component*. An example *core data type* might for instance be *measurement*. The *content component* would be the number 12. Additional *supplementary components* could be *measurement type* (temperature) and *measurement unit* (Fahrenheit). Hence the three basic values are combined in order to form a more complex type - a *core data type*.

UML requires, that each attribute of a class has a certain type. In case of *content components* and *supplementary components* the valid type is either a *primitive type* (PRIM) or an *enumeration* (ENUM). As already outlined in the introduction to core components an *aggregate core component* (ACC) is modeled using UML classes. The attributes of an *aggregate core components* are stereotyped as *basic core components* (BCC). A *basic core component* attribute has a certain type - a so called *core data type* (CDT). In order to build a hierarchical structure between different *aggregate core components* it is possible to use the concept of a UML composition stereotyped as *association core component* (ASCC). By definition every *association core component* must have a *source* and a *target*.

Analogue to the concept of a *core data type* (CDT) a *qualified data type* (QDT) consists of exactly one *content component* (CON) and multiple *supplementary components* (SUP) which follow the same purpose as a *core data type*. Similar to the concept of *core components* an *aggregate business information entity* (ABIE) is modeled using a UML class. It consists of several attributes which are stereotyped as *basic business information entities* (BBIE). A BBIE attribute has a certain type - a so called *qualified data type* (QDT). In order to build a hierarchy of different *aggregate business information entities* UML compositions stereotyped as *association business information entities* (ASBIE) are used. Again it is required that there is exactly one *aggregate business informa-*

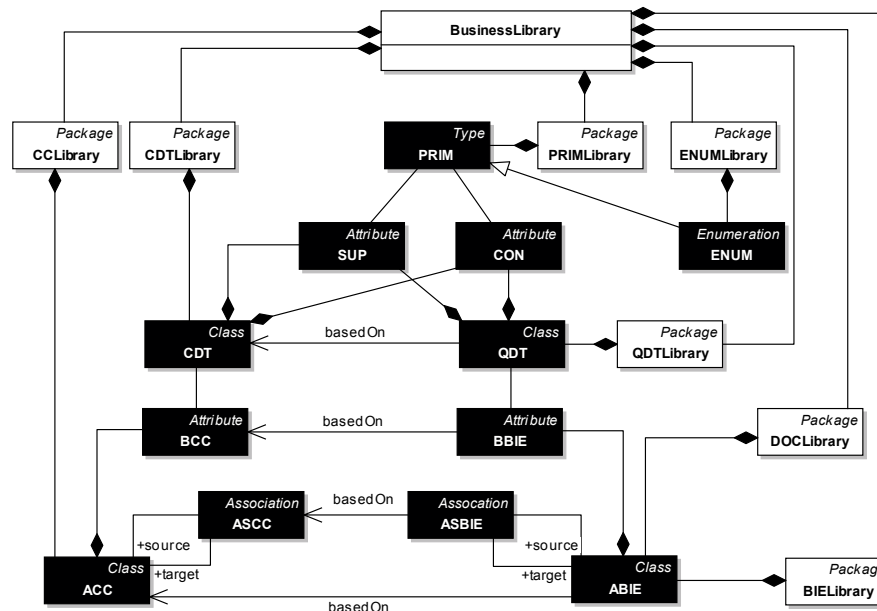


Figure 4: UPCC meta model

tion entity as `source` and one as `target`.

The different modeling artifacts are aggregated using packages. Indicated by its name each package aggregates a certain type of artifact or is itself aggregated by another package. Two packages have a particular role: *DOCLibrary* and *BusinessLibrary*. A *DOCLibrary*, shown on the lower right hand side of figure 4, is used to aggregate different *business information entities* forming a self contained business document. Each *DOCLibrary* therefore represents exactly one type of business document.

The different packages are eventually aggregated in the so called *BusinessLibrary*. The business document modeler constructs all necessary business documents of a given business collaboration in the business library which may be integrated in a business process model. The business process model specifies the exact exchange order of the different business documents. However the business process perspective and its methodology are not subject to this article. For an integration of a business document model in a business process model we would like to refer the interested reader to (Hofreiter et al. 2006). Having clarified the different stereotypes of the UML Profile for Core Components the following section will outline a holistic example from the sales domain.

## 5 Core Components by example

In figure 5 the example package structure of a business document model, using the UML Profile for Core Components is shown. The scenario is taken from a purchase order process taking place between a buyer and a seller where the buyer sends a purchase order to the seller. The example used in this article has been created using the UML modeling tool Enterprise Architect (Sparx 2008). The relevant diagrams contained in the different packages in figure 5 are shown in figure 6. Using the numbered dots the packages in figure 5 are connected to the pertaining diagrams in figure 6.

First the modeler searches in the existing *core component* library which is maintained by UN/CE-FACT for a *core component order*. The existence of a generic *aggregate core component order* as shown in (A) is assumed. The *aggregate core component order* consists of several *basic core components* and *associa-*

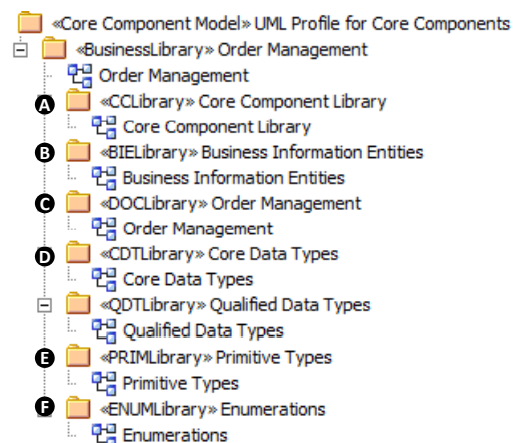


Figure 5: UPCC example package structure

tion core components. For the *purchase order* scenario not all of the *basic core components* and *association core components* are needed. Hence, the modeler restricts the generic *aggregate core component order* (A) to the business context specific *aggregate business information entity purchase\_order* (B).

As outlined in (B) the *aggregate core component charge* and several *basic core components* from the different *aggregate core components* in (A) have not become part of the final business document model (B). Furthermore qualifiers such as *purchase\_* or *purchaseorder\_* are used for the *aggregate business information entities* in (B) in order to allow for a better differentiation between core components and business information entities in the overall model.

In order to comply with the specific requirements of a *purchase order* the modeler derives a *qualified data type purchase order identifier* from the generic *core data type identifier* in (D). Similar to the relationship between a *core component* and a *business information entity* a *qualified data type* is always derived from a *core data type* by restriction.

The *core data type identifier* is used several times in the core component model (A) as indicated by an exemplary dotted line. For the specific needs of the *aggregate business information entity product unit-*

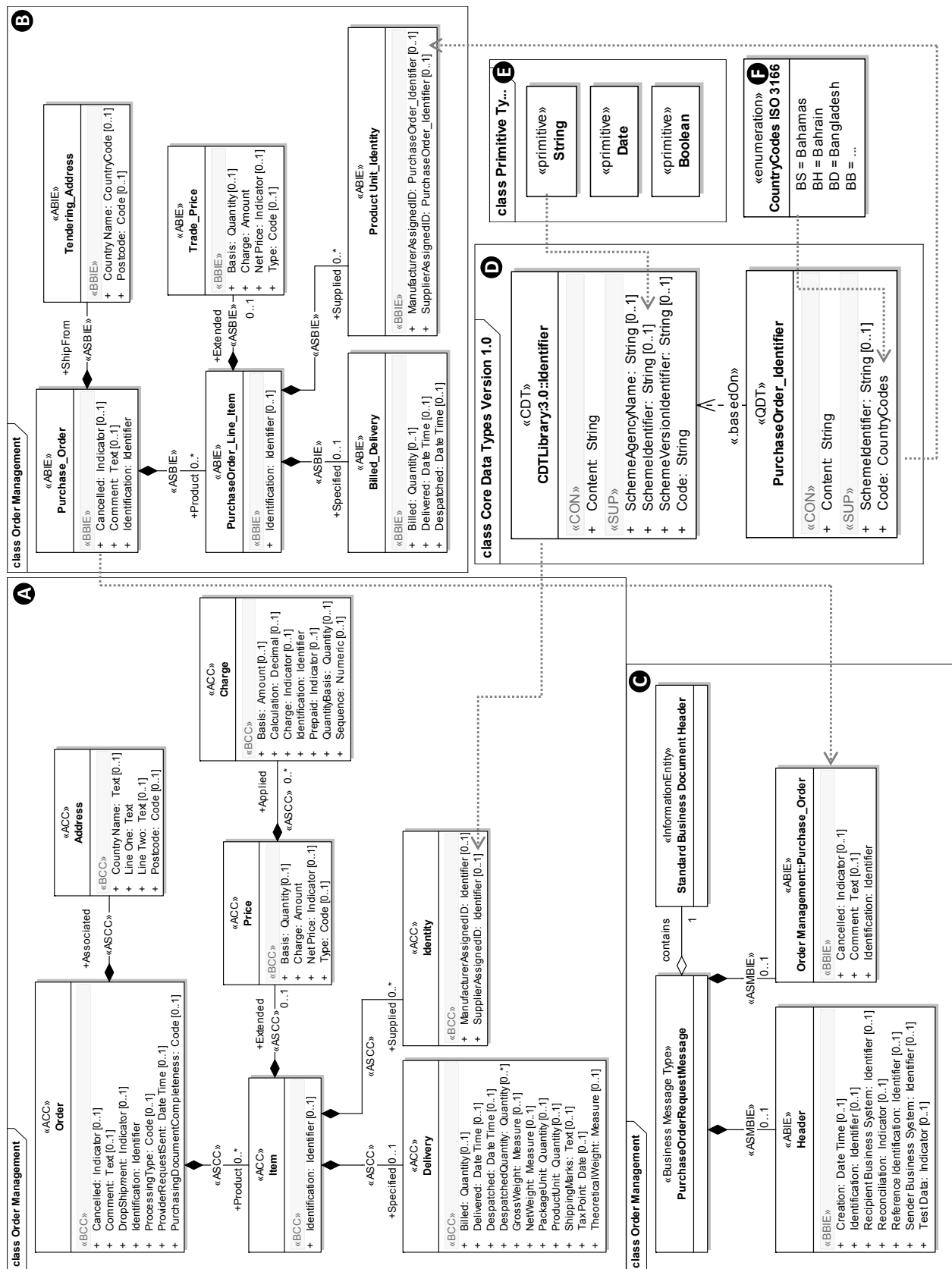


Figure 6: UPCC example model

identity the specialized *qualified data type* `purchase_order.identifier` is used instead of the generic type `identifier`. Hence the modeler can restrict the type of a *basic core component* when transferring it to a *basic business information entity*. Please note, that some of the *basic core components* and their types remain unchanged and are simply taken over from the core component model (A) to the business information entity model (B). Where no specialization of a data type is necessary, the *basic business information entities* simply use the *core data type* of the underlying *basic core component*. An *aggregate core component* `address` (A) becomes a `tendering_address` when used in the `purchase_order` context. The *aggregate business information entity* `tendering_address` restricts the generic *aggregate core component* to two *basic business information entities* namely `country_name` and `postcode`. While the type of `postcode` remains a *core data type* and therefore unchanged (`code`), the `country_name` becomes a *qualified data type* in (B) namely `country_code`. The *qualified data type* `country_code` is not shown in figure 6.

*Primitive types* (E) are used to set the type of *supplementary components* and *content components* in *core data types* (CDT) and *qualified data types* (QDT) as shown in (D). The *enumeration* `country_codes` (F) is used to restrict the *supplementary component* `code` of the *qualified data type* `purchase_order.identifier` as shown in (D). Finally the business document is assembled in (C). The business message is a `purchase_order_request_message` which has a `standard_business_document_header` and a regular `header`. Both header elements carry additional meta information about the actual business document `purchase_order_header`. `Header` and `purchase_order_header` are connected to the `purchase_order_request_message` using two *association messaging business information entities* (ASMBIE). The concept is the same as the one of an *association business information entity*, only the naming is different when used in the context of a business message.

The *business message type* `purchase_order_request_message` as well as the `header` and `standard_business_document_header` as shown in (C) in figure 6 are forming the embracing part of the actual business document. The actual payload of the `purchase_order_request_message` is the *aggregate business information entity* `purchase_order` and all its relating aggregate business information entities as shown in (B) in figure 6. The next section will introduce the usage of naming and design rules in order to uniquely derive XML schema artifacts from a core component model.

## 6 Deriving XML artifacts from Core Components

The previous sections introduced the core components methodology and the UML profile for core components. Using the core components methodology, the modeler can define a business document on a conceptual level in a unique and semantically precise manner. For the exchange of business documents between companies and B2B systems, however, a logical level representation of business documents is needed. This section will outline how the conceptual core component model can be used to derive XML schema artifacts. These XML artifacts form the logical level business document model to which every document instance exchanged between two B2B systems must conform to.

In order to allow for a unique representation of core components in XML, UN/CEFACT suggests the use of so called Naming and Design Rules (NDR) (UN/CEFACT 2006c). Along with each new release of the Core Components Technical Specification and

its UML Profile, UN/CEFACT delivers pertaining Naming and Design Rules.

Since a real world core component example can easily become extensive and complex a manual transformation of core components represented in UML to the appropriate XML schema artifacts is not efficient. In order to overcome the limitations of a manual transformation we have built an XML schema generator. The XML schema generator is part of a larger set of tools, supporting the modeler in inter-organizational business process and business data modeling, known as the UMM Add-In (RSA 2007). As already outlined in figure 5 a core component model is defined using stereotyped packages which follow a rigid structure. Using the UMM Add-In the user simply clicks on a package and initiates the transformation of core components to the appropriate XML schema representation. The XML schema generator automatically detects dependencies in the core component model and generates additional XML schema files, containing data type definitions, core component definitions etc.

Listing 1 shows the XML schema representation of the example model shown in (B) in figure 6. As outlined by the dotted arrow from (B) to (C) in figure 6 this code is attached to the final `purchase_order_request_message` in (C). The XML schema generator iterates over every *aggregate business information entity* in (B) and constructs a `complexType` with a `sequence` for each. The six complex types are shown in line 5, 14, 22, 30, 37 and 43 in listing 1. The root element of the business document is shown in line 4.

For each *basic business information entity* an element is created in the `sequence` of the embracing *aggregate business information entity's* `complexType`. Since every *basic business information entity* has a certain type (either *core data type* or *qualified data type*) the necessary `complexTypes` have to be referenced. As shown in figure 6 the *qualified data types* and *core data types* are defined in a separate library to the *business information entity library*. The generator automatically detects these dependencies, creates the necessary auxiliary schemes, and imports them into the final schema. In listing 1 the *core data type library* is imported in line 2 and the *qualified data type library* is imported in 3.

For each *association business information entity* an element in the `complexType's` `sequence` of the *aggregate business information entity* is created as well. As outlined in figure 6 the *aggregate business information entity* `purchase_order` has two *association business information entities* namely `ship_from` (`tendering_address`) and `product` (`purchase_order_line_item`). As shown in line 10 and 11 in listing 1 or each *association business information entity* the correct `complexType` is set.

Since the final *business information entity* schema uses *core data types* and *qualified data types*, auxiliary schema files have to be created for each data type library. Listing 2 shows a cut-out from the *core data type library* imported in line 2 in listing 1. As outlined in the introduction to the UML profile for core components every *core data type* and every *qualified data type* consists of exactly one *content component* and multiple *supplementary components*. As shown in listing 2 the XML schema generator maps the *content component* to an `extension` (line 53) and the *supplementary components* to `attributes` (line 54 to 56) for the *core data type* `code` type. The *core data type schema* is imported into the final *business information entity schema* as shown in listing 1 and its data types are used for several *basic business information entities* (line 27 and line 46 in listing 1). The specific data type of a *content component* or a *supplementary component* can either be a *primitive type* or an



enumeration.

Listing 1: Purchase\_Order XML schema

```

1 <xs:schema xmlns:cdt="sample:cdtlibrary" xmlns:bie="
  sample:bielibrary" xmlns:qdt="sample:qdtlibrary"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  targetNamespace="sample:bielibrary"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified" version="
  notSpecified">
2 <xs:import namespace="sample:cdtlibrary" schemaLocation
  ="CDTLibrary_3.0.xsd"/>
3 <xs:import namespace="sample:qdtlibrary" schemaLocation
  ="QDTLibrary_3.0.xsd"/>
4 <xs:element name="Purchase_Order" type="
  bie:Purchase_OrderType"/>
5 <xs:complexType name="Purchase_OrderType">
6 <xs:sequence>
7 <xs:element name="Cancelled" type="cdt:IndicatorType"
  minOccurs="0"/>
8 <xs:element name="Comment" type="cdt:TextType"
  minOccurs="0"/>
9 <xs:element name="Identification" type="
  cdt:IdentifierType"/>
10 <xs:element name="ProductPurchaseOrder_Line_Item"
  type="bie:PurchaseOrder_Line_ItemType" minOccurs
  ="0" maxOccurs="unbounded"/>
11 <xs:element name="ShipFromTendering_Address" type="
  bie:Tendering_AddressType"/>
12 </xs:sequence>
13 </xs:complexType>
14 <xs:complexType name="PurchaseOrder_Line_ItemType">
15 <xs:sequence>
16 <xs:element name="ExtendedTrade_Price" type="
  bie:Trade_PriceType" minOccurs="0"/>
17 <xs:element name="Identification" type="
  cdt:IdentifierType" minOccurs="0"/>
18 <xs:element name="SpecifiedBilled_Delivery" type="
  bie:Billed_DeliveryType" minOccurs="0"/>
19 <xs:element name="SuppliedProduct_Unit_Identity" type
  ="bie:Product_Unit_IdentityType" minOccurs="0"
  maxOccurs="unbounded"/>
20 </xs:sequence>
21 </xs:complexType>
22 <xs:complexType name="Trade_PriceType">
23 <xs:sequence>
24 <xs:element name="Basis" type="cdt:QuantityType"
  minOccurs="0"/>
25 <xs:element name="Charge" type="cdt:AmountType"/>
26 <xs:element name="Net_Price" type="cdt:IndicatorType"
  minOccurs="0"/>
27 <xs:element name="Type" type="cdt:CodeType" minOccurs
  ="0"/>
28 </xs:sequence>
29 </xs:complexType>
30 <xs:complexType name="Billed_DeliveryType">
31 <xs:sequence>
32 <xs:element name="Billed" type="cdt:QuantityType"
  minOccurs="0"/>
33 <xs:element name="Delivered" type="cdt:DateTimeType"
  minOccurs="0"/>
34 <xs:element name="Despatched" type="cdt:DateTimeType"
  minOccurs="0"/>
35 </xs:sequence>
36 </xs:complexType>
37 <xs:complexType name="Product_Unit_IdentityType">
38 <xs:sequence>
39 <xs:element name="ManufacturerAssignedID" type="
  qdt:PurchaseOrder_IdentifierType" minOccurs="0"/>
40 <xs:element name="SupplierAssignedID" type="
  qdt:PurchaseOrder_IdentifierType" minOccurs="0"/>
41 </xs:sequence>
42 </xs:complexType>
43 <xs:complexType name="Tendering_AddressType">
44 <xs:sequence>
45 <xs:element name="Country_Name" type="
  qdt:CountryCodeType" minOccurs="0"/>
46 <xs:element name="Postcode" type="cdt:CodeType"
  minOccurs="0"/>
47 </xs:sequence>
48 </xs:complexType>
49 </xs:schema>

```

Listing 2: Core Data Type Schema

```

50 <xs:schema xmlns:udt="sample:cdtlibrary" xmlns:xs="http:
  //www.w3.org/2001/XMLSchema" targetNamespace="
  sample:cdtlibrary" elementFormDefault="qualified"
  attributeFormDefault="unqualified" version="
  notSpecified">
51 <xs:complexType name="CodeType">
52 <xs:simpleContent>
53 <xs:extension base="xs:string">
54 <xs:attribute name="
  ListAgencyIdentifier
  " type="xs:string"
  use="optional"/>
55 <xs:attribute name="
  ListVersionIdentifier
  " type="xs:string"
  use="optional"/>
56 <xs:attribute name="
  ListIdentifier"
  type="xs:string"
  use="optional"/>
57 </xs:extension>
58 </xs:simpleContent>
59 </xs:complexType>
60 [...]
61 </xs:schema>

```

In listing 2 both, the *content component* and the *supplementary components*, used the *primitive type* string. In case a *primitive type* is set, the XML generator uses the built-in data type of the XML schema specification (W3C 2001). Sometimes the definition

of a *primitive type* is too weak e.g. in case the modeler wants to restrict the value of a *supplementary* or a *content component* to a specific set of values. In this case the concept of a so called *enumeration* is used. Listing 3 shows the XML representation of the *enumeration* shown in (F) in figure 6. In listing 3 the XML generator creates a *simpleType* for each *enumeration* (line 63) and uses the concept of a *restriction* (line 64) to define a set of valid values (line 65-67).

Listing 3: Enumeration Schema

```

62 <xs:schema xmlns:qdt="sample:qdtlibrary" xmlns:enum="
  sample:enumlibrary" xmlns:ccts="
  urn:un:unece:unefact:documentation:standard:CCTS:2
  " xmlns:xs="http://www.w3.org/2001/XMLSchema"
  targetNamespace="sample:enumlibrary"
  elementFormDefault="qualified" attributeFormDefault
  ="unqualified" version="notSpecified">
63 <xs:simpleType name="CountryCodesType">
64 <xs:restriction base="xs:token">
65 <xs:enumeration value="BS"/>
66 <xs:enumeration value="BH"/>
67 <xs:enumeration value="BD"/>
68 </xs:restriction>
69 </xs:simpleType>
70 [...]
71 </xs:schema>

```

*Qualified data types* are used to further restrict a *core data type* to the specific needs of a certain industry or business domain. Since *basic business information entities* can either have a *core data type* or a *qualified data type* as designated type, an auxiliary schema for *qualified data types* has to be created as well. Listing 4 shows a cut-out from the *qualified data type* schema which is imported in the final business document model in line 3 in listing 1.

As mentioned before *supplementary components* and *content components* can be restricted using the concept of enumerations. Line 74 in listing 4 shows how a *qualified data type* is restricted to a set of valid values. The *complexType* is restricted using the concept of a *simpleContent* which is based on an *enumeration* (line 76 in listing 4). The necessary *enumeration* is automatically detected by the XML schema generator and imported into the *qualified data type* schema (line 73 in listing 4).

In (D) in figure 6 the derivation of a *qualified data type* (purchase order.identifier) from a generic *core data type* (identifier) has been shown. The XML generator automatically detects the necessary dependencies and generates a *complexType* for each *qualified data type*. The *complexType* for the *qualified data type* purchase order.identifier type is shown in line 79 to 86 in listing 4. In listing 4 the content component of the qualified data type is depicted using an extension (line 81) and the supplementary components are specified using attributes (line 82-83).

Line 83 shows how the generic code attribute is restricted to a set of country codes.

Listing 4: Qualified Data Type Schema

```

72 <xs:schema xmlns:qdt="sample:qdtlibrary" xmlns:enum="
  sample:enumlibrary" xmlns:xs="http://www.w3.org
  /2001/XMLSchema" targetNamespace="sample:qdtlibrary"
  elementFormDefault="qualified"
  attributeFormDefault="unqualified" version="
  notSpecified">
73 <xs:import namespace="sample:enumlibrary"
  schemaLocation="ENUMLibrary_3.0.xsd"/>
74 <xs:complexType name="CountryCodeType">
75 <xs:simpleContent>
76 <xs:extension base="
  enum:CountryCodesType"/>
77 </xs:simpleContent>
78 </xs:complexType>
79 <xs:complexType name="
  PurchaseOrder_IdentifierType">
80 <xs:simpleContent>
81 <xs:extension base="xs:string">
82 <xs:attribute name="
  SchemeIdentifier"
  type="xs:string"
  use="optional"/>
83 <xs:attribute name="Code
  " type="
  enum:CountryCodesType
  " use="required"/>
84 </xs:extension>
85 </xs:simpleContent>
86 </xs:complexType>
87 [...]
88 </xs:schema>

```

The XML artifacts for a core component library are constructed similarly to a business information entity library. Due to space limitations however these generation artifacts are not shown in this article. This section has shown how the XML schema generator can be used to automatically generate a 1:1 representation of the conceptual level core component model in logical level XML schema syntax.

## 7 Conclusion and Future Trends

In this article we presented UN/CEFACT's core components technology. Using the concept of core components business documents can be defined on a conceptual and syntax independent level. In order to allow for an integration of the core components technology into a UML modeling tool, the UML Profile for Core Components has been developed. Using the UML profile the modeler can retrieve existing generic core components from a standardized library. The generic core components are further restricted in order to create industry and context specific business information entities.

We introduced our tool based XML generator which follows the guidelines specified in the Naming and Design Rules of UN/CEFACT. Using a single mouse click the XML generator automatically iterates over a complete core component model and generates XML schema files. The schema files are used to validate the exchanged business documents in an actual B2B collaboration. Furthermore they are needed for the definition of entry/exit points in a service oriented architecture e.g. the generated schema is imported into a WSDL file. If both B2B partners have the same business document definitions the entry/exit points of both business partners match.

The contribution of our approach in regard to the stated shortcomings of current business document modeling approaches is therefore as follows:

(1) *Standard incompatibilities.* Since core components define a business document on a conceptual level and not on the logical level (e.g. XML schema file) there cannot be any inconsistencies in regard to the representation of a specific business document. This is a major advantage over standards based on pure implementation logic (e.g. XML).

(2) *All-in-one approach.* Similar to EDI approaches UN/CEFACT defines a set of normative core components which are defined for a whole industry domain e.g. steel industry. Due to the derivation by restriction mechanism of core components no overhead occurs in a well defined business context. The generic industry specific core component is tailored to a business information entity for a given business domain. Since every business information entity has a *basedOn* dependency to its underlying generic core component the common semantic basis for every business information entity is given, hence allowing a matching even between business information entities from different business contexts.

(3) *Lack of conceptual document description.*

Core components are defined on a conceptual meta level, independent of any transfer syntax. With every release of the core components standard UN/CEFACT releases a set of well defined Naming and Design rules, allowing the mapping to a logical level implementation of core components. In case of a change in the transfer syntax only the mapping rules have to be altered instead of a complete standard re-engineering.

Furthermore the UML profile for Core Components can be used to easily assemble business documents using UML artifacts. Such models can be handed over to other business modelers, software de-

velopers and stakeholders in order to communicate the structure and semantics of specific business documents. Using such a conceptual business document representation greatly facilitates and enhances a software implementation process through increased legibility and apprehension.

Although the current implementation of the UML Profile for Core Components together with the XML generator provides a solid basis for business document modeling and schema artifact generation, several tasks are planned for the future. On the one hand the current UML Profile for Core Components (UPCC) is based on the Core Component Technical Specification (CCTS) version 2.01 (UN/CEFACT 2003). Since CCTS 3.0 will be released this year, the UML profile has to be updated to meet the requirements of the new core component version. Every core component release also includes updated Naming and Design Rules. Therefore the XML schema generator has to be updated accordingly.

In order to further foster the dissemination of the core component technology the usability of the UML Profile for Core Components must be enhanced. A first step allowing inexperienced users to use the core components technology is the implementation of a validation routine for the core component's UML profile.

Since the core component model of a business document is independent of any implementation syntax the derivation of other artifacts than XML schema could be possible as well. Following modified Naming and Design Rules the core component model may be used to derive Relax NG (OASIS 2001a), UBL (OASIS 2006) or EDIFACT (UN/CEFACT 2007) artifacts as well.

## References

- ACORD (2007), *ACORD Insurance Data Standards*.
- Berge, J. (1994), *The EDIFACT Standards*, 2 edn, Blackwell Publishers, Cambridge, MA, USA.
- Bernauer, M., Kappel, G. & Kramler, G. (2003), Approaches to implementing active semantics with XML schema, in 'Proceedings of the 14th International Workshop on Database and Expert Systems Applications DEXA03, Prague, Czech Republic, September 1-5', Springer, Berlin, pp. 559-565.
- Chen, P. P.-S. (1976), 'The Entity Relationship Model: Towards a unified view of data', *ACM Transactions on Database Systems* 1(1), 9-36.
- CIDX (2007), *Chemical Industry Data Exchange Standard*.
- Combi, C. & Oliboni, B. (2006), Conceptual modeling of XML data, in 'Proceedings of the ACM symposium on applied computing SAC06, Dijon, France, April 23-27', ACM, USA, pp. 467-473.
- Glushko, R. & McGrath, T. (2005), *Document Engineering*, 2 edn, Massachusetts Institute of Technology, USA.
- HL7 (2007), *Health Level Seven*.
- Hofreiter, B., Huemer, C., Liegl, P., Schuster, R. & Zapletal, M. (2006), UN/CEFACT's Modeling Methodology (UMM): A UML Profile for B2B e-Commerce, in 'Advances in Conceptual Modeling - Theory and Practice, ER 2006 Workshops BP-UML, Tucson, United States, November 6-9', Springer, Berlin, pp. 19-31.

- ISO (2007), *ISO 3166 List of country names and code elements*.
- Jung, J.-Y., Hur, W., Kang, S.-H. & Kim, H. (2004), 'Business process choreography for B2B collaboration', *IEEE Internet Computing Journal* 8, 37–45.
- Li, H. (2000), 'XML and Industrial Standards for Electronic Commerce', *Knowledge and Information Systems* 2(4), 487–497.
- Li, J. B. & Miller, J. (2005), Testing the semantics of W3C XML schema, in 'Proceedings of the 29th Annual International Computer Software and Applications Conference COMPSAC2006, Edinburgh, U.K., July 26-28', IEEE, USA, pp. 443–448.
- Malik, A. (2003), 'XML Schemas in an Object Oriented Framework', *XML Journal*.  
**URL:** <http://xml.sys-con.com/read/40580.htm>
- Mani, M., Lee, D. & Muntz, R. R. (2001), Semantic Data Modeling Using XML Schemas, in 'Proceedings of the 20th International Conference on Conceptual Modeling ER 2001, Yokohama, Japan, November 27-30', Vol. 2224, Springer, Berlin, pp. 149–163.
- OASIS (2001a), *RELAX NG Specification*, Organization for the Advancement of Structured Information Standards.
- OASIS (2006), *Universal Business Language v2.0*, Organization for the Advancement of Structured Information Standards.
- OASIS (2007), *Universal Description, Discovery, and Integration*, Organization for the Advancement of Structured Information Standards.
- OASIS, U. (2001b), *ebXML - Technical Architecture Specification*, Organization for the Advancement of Structured Information Standards, United Nations Center for Trade Facilitation and Electronic Business.
- OMG (2007), *Unified Modeling Language 2.1*, Object Management Group.
- papiNet (2007), *papiNet*.
- PIDX (2007), *Petroleum Industry Data Exchange*.
- Rinderle, S., Wombacher, A. & Reichert, M. (2006), On the Controlled Evolution of Process Choreographies, in 'Proceedings of the 22nd International Conference on Data Engineering ICDE06, Atlanta, Georgia, April 3-8', IEEE, USA, pp. 124–124.
- RosettaNet (2006), *RosettaNet Business Documents*.
- RSA (2007), *The UMM Add-In 1.0*, Research Studios Austria.  
**URL:** <http://ummmaddin.researchstudio.at>
- Salim, F., Price, R., Krishnaswamy, S. & Indrawan, M. (2004), UML Documentation Support for XML Schema, in 'Proceedings of the Australian Software Engineering Conference ASWEC2004, Melbourne, Australia, April 13-16', Vol. 2, Australia, pp. 211–220.
- Skogan, D. (1999), UML - A Schema Language for XML based Data Interchange, in 'Proceedings of the Second International Conference on the Unified Modeling Language UML99, Colorado, USA, October 28-30', Vol. 2, USA, pp. 211–220.
- Sparx (2008), *Enterprise Architect*, Sparx Systems.
- SWIFT (2007), *Society for Worldwide Interbank Financial Telecommunication*.
- UN/CEFACT (2003), *Core Components Technical Specification 2.01*, United Nations Center for Trade Facilitation and Electronic Business.
- UN/CEFACT (2006a), *UML Profile for Core Components 1.0*, United Nations Center for Trade Facilitation and Electronic Business.
- UN/CEFACT (2006b), *UN/CEFACT's Modeling Methodology 1.0*, United Nations Center for Trade Facilitation and Electronic Business.
- UN/CEFACT (2006c), *XML Naming and Design Rules 2.0*, United Nations Center for Trade Facilitation and Electronic Business.
- UN/CEFACT (2007), *UN/EDIFACT D.07B*, United Nations Center for Trade Facilitation and Electronic Business.
- UN/CEFACT (2008), *Core Components Technical Specification 3.0*, United Nations Center for Trade Facilitation and Electronic Business.
- W3C (2001), *XML Schema Language*, World Wide Web Consortium.
- W3C (2006), *Extensible Markup Language*, World Wide Web Consortium.
- W3C (2007a), *Simple Object Access Protocol*, World Wide Web Consortium.
- W3C (2007b), *Web Services Description Language*, World Wide Web Consortium.
- Yu, A. & Steele, R. (2005), An overview of research on reverse engineering XML schemas into UML diagrams, in 'Proceedings of the Third International Conference on Information Technology and Applications ICITA 2005, Sydney, Australia, July 4-7', Vol. 2, Australia, pp. 772–777.



# Contrasting Classification with Generalisation

Thomas Kühne

School of Mathematics, Statistics and Computer Science  
Victoria University of Wellington  
PO Box 600, Wellington, New Zealand,  
Email: Thomas.Kuehne@mcs.vuw.ac.nz

## Abstract

Classification and Generalisation are two of the most important abstraction mechanisms in modelling, and while they share a number of similarities, they are unmistakably different with respect to their properties. Recently, a number of (meta-) modelling language design approaches de-emphasised the differences between classification and generalisation in order to gain various advantages. This paper aims to demonstrate the loss in precision and the loss of sanity checks such approaches entail. After a careful comparison between classification and generalisation, I identify problems associated with the above mentioned approaches and offer alternatives that retain a strong distinction between classification and generalisation.

*Keywords:* classification, generalisation, strict meta-modelling

## 1 Introduction

The main purpose of modelling is to reduce complexity. Sometimes it suffices to simply reduce the information per element in the universe of discourse but otherwise retain a one-to-one correspondence between these elements and model elements (i.e., creating a token model (Kühne (2006))). However, often there is a need to use more abstract views, in particular, to disregard particularities of individual elements and only capture the relevant universal properties, creating a many-to-one correspondence between elements from the universe of discourse and modeling elements. One thus obtains a way to characterise many individuals by referring to one representative only. In particular, classification is used to create types from instances, giving rise to type models (Kühne (2006)), abstracting away from the identity of instances and their different property values. On the other hand, generalisation is used to create a *genus* (hypernym) from *species* (hyponyms) (Rayside & Campbell (2000)), i.e., to create super-types from subtypes, thus abstracting away from a number of subtypes and their respective differences.

Today, the differences between classification and generalisation are well understood, but this has not always been the case. Before Frege laid the foundation for a modern axiomatic logic with his “Begriffsschrift” (Concept Script) in 1879, there was no systematic way to avoid mistakes arising from confus-

ing classification with generalisation and the logical fallacies that inevitably follow (see section 3). The *arbor porphyriana* (aka, “tree of knowledge”, a concept tree as inspired by Porphyrios) in the version of Purchotius from 1730 relates a supertype (e.g., “Animal”) with a type (e.g., “Homo”) in the same way as a type (“Homo”) with its instances (e.g., “Petrus”) (Purchotius (1730)). This lack of proper distinction between classification and generalisation could still be observed in 1890 (see Frege (1892, page 193, footnote 4)). As such, this relatively modern attainment should not be given up lightly, even when apparent advantages seem to suggest this from a pragmatic point of view.

A number of (meta-)modelling language design approaches have been proposed that can be regarded as emphasising the similarities between classification and generalisation and de-emphasising their differences in order to yield

- a simplified language design (Jackson (2006)),
- more flexibility in metamodeling (Varró & Pataricza (2003), Gitzel & Merz (2004)), and
- an increased utility of a language specification (OMG (2004)).

In this paper, I argue that there is value in maintaining a clear distinction between classification and generalisation, and that alternatives to the above mentioned approaches exist that maintain a clear distinction.

Section 2 compares classification with generalisation pointing out similarities and fundamental differences. Section 3 analyses some of the aforementioned approaches and suggests alternative solutions. Section 4 concludes.

## 2 Comparison

The characterisation of classification and generalisation in the introduction, as typically using instances and types as their domains respectively, suggests that these abstraction mechanisms serve very different purposes and indeed this is the case for most common usage scenarios. However, note that classification may also be performed on types (metamodeling, see Kühne (2006)) and it is possible to generalise at the instance level as well, leading to so-called *abstract objects* (Mittelstraß (1995)). In the following comparison, I am hence careful to compare classification with generalisation by applying them to the same domain in order to avoid observing discrepancies that only exist due to the application to different domains.

## 2.1 Formalisation

For the following comparison it is useful to introduce the notion of a “concept” as conceived by Frege. Using the terminology of his pupil Carnap (Carnap (1947)), a concept  $C$  has an extension  $\varepsilon(C)$ , all instances falling under the concept  $C$ , and an intension  $\iota(C)$ , a predicate characterizing whether an element belongs to the concept or not, so that

$$\varepsilon(C) = \{x \mid \iota(C)(x)\} \quad (1)$$

We can now state whether a concept  $C$  *classifies* an instance  $e$ , i.e.  $e \Delta C$ , using an extensional or an intensional viewpoint. Here’s the extensional variant:

$$e \Delta C \iff e \in \varepsilon(C). \quad (2)$$

Using the usual interpretation of generalisation, a concept  $C_g$  is more general than another concept  $C_s$ , i.e.,  $C_s < C_g$ , if it includes all the instances falling under  $C_s$ :

$$C_s < C_g \iff \varepsilon(C_s) \subseteq \varepsilon(C_g). \quad (3)$$

Equations 2 & 3 make it obvious that a direct comparison between classification and generalisation is hindered by the fact that the former’s domain<sup>1</sup> typically consists of instances and the latter’s domain typically consists of concepts.

One way to enable an adequate comparison would be to look at concepts  $C_g$  and  $C_s$  as instances, i.e., instead of considering their *type facets* (e.g., their attributes), by which they define the shape of their instances, one could consider their *instance facets* (e.g., their property values), that is, properties that are associated with the concepts themselves (Atkinson & Kühne (2003)). For example, for the purpose of modelling a pet store, the instance facet of the concept “Dog” could have a tax rate property with the value “16%” whereas the instance facet of concept “DogFood” could have the value “7%” for the same property.

However, this way of looking at concepts is unfamiliar to most and would also imply that we had to use meta-concepts to classify concepts. Therefore, I perform the comparison at the instance level and use generalisation at the instance level by using the notion of an *abstract object* (Kamlah & Lorenzen (1996)). An abstract object represents all instances that are considered to be equivalent to each other for a certain purpose, e.g.,

$$P(|x|_{\sim}) \iff \forall y : y \sim x \rightarrow P(y)$$

The abstraction operator  $|\cdot|_{\sim}$  gives us a way to make a statement about all instances that are considered equivalent to each other. For example, while

$$\text{HasFourLegs}(\text{Lassie})$$

is true if the instance “Lassie” has four legs, the expression

$$\text{HasFourLegs}(|\text{Lassie}|_{\sim_{\text{dog}}}) \quad (4)$$

is only true if *all* instances considered equivalent to “Lassie” have four legs (by means of  $\sim_{\text{dog}}$ , which here is meant to regard all instances of subspecies “Dog” to be equivalent with each other).

Note that  $|\text{Lassie}|_{\sim_{\text{dog}}}$  is not a type/concept. What we assert of  $|\text{Lassie}|_{\sim_{\text{dog}}}$  is asserted of an instance (and all other instances that are equivalent to it). The expression

$$\text{NamedByLinnaeusIn1758}(|\text{Lassie}|_{\sim_{\text{dog}}}) \quad (5)$$

<sup>1</sup>The type of the left hand side element in a relation.

is false since Linnaeus did not name all dogs, but the concept “Dog” (Canis lupus familiaris), i.e., the subspecies subordinate to species “Canis lupus”.

It is of course true, though, that  $|\text{Lassie}|_{\sim_{\text{dog}}}$  implicitly defines a set of instances (an equivalence class), which could be the extension of a concept—in particular, if the equivalence relation  $\sim_{\text{dog}}$  is defined by referring to a predicate  $\text{Dog}(X)$ , i.e., instances are considered equivalent with each other if they satisfy predicate  $\text{Dog}(X)$ . Yet,  $|\text{Lassie}|_{\sim_{\text{dog}}}$  refers to *all instances* of that set, not the *set* itself. In other words, the concept “Dog” is not a collection of dogs.

If one wants to introduce an alternative name to the notion of an abstract object like  $|\text{Lassie}|_{\sim_{\text{dog}}}$  then *prototype* would best describe its nature. An abstract object captures what is universal about a set of instances but resides at the same logical level as the instances, much like an object in a prototype-based language from which other objects can be cloned (Ungar & Smith (1987)). It thus has the quality of a type but is not (yet) a type, hence “*prototype*”.

Finally, note that the way we generalise from *Lassie* to  $|\text{Lassie}|_{\sim_{\text{dog}}}$  conforms to how a number of special concepts may be generalised to a general one (see equation 3). A general concept can be regarded as capturing what is universal about its subconcepts. This is true with respect to instance facet properties, e.g., tax rate values, but also with respect to type facet properties, e.g., attributes that require certain properties for instances, such as “age : Integer”. If an equivalence relation  $\sim_{\text{named}}$  considers all subconcepts to be equivalent that feature a “name” attribute and is used on a number of subconcepts, such as “Collie”, “Poodle”, and “Beagle”, then  $|\text{Collie}|_{\sim_{\text{named}}}$  is the generalisation of these subconcepts and could be labelled “NamedDog”.

Because a general concept  $C_g = |C_i|_{\sim}$  is derived from more specialised concepts  $C_i$  ( $i \in [1..n]$ ) by disregarding differences in the  $C_i$ , every  $C_i$  will at least have the requirements on instances that  $C_g$  has, which means that if an instance satisfies the requirements of a  $C_i$ , it will also satisfy the requirements of  $C_g$ :

$$\forall i, x : \iota(C_i)(x) \rightarrow \iota(C_g)(x) \quad (6)$$

Here, I am referring to the intensions of concepts since I want to emphasise the fact that a concept can be viewed as being independent from the instances it describes. A concept resides at a higher logical language level than the instances within its extension, and its intension can be used as a judge with respect to instances that may or may not fall under the concept. This view of concepts is particularly important if one wants to deal with dynamic extensions that may shrink/grow over time and it is the prevalent one in modelling languages such as the UML ((OMG (2007))) where a class / type is regarded as an intensional description of its instances.

From equations 1 & 6 it follows that the extension of the superconcept is a superset of the union of the extensions of its subconcepts.

$$\varepsilon(C_g) = \varepsilon(|C_1|_{\sim}) \supseteq \bigcup_i \varepsilon(C_i). \quad (7)$$

In equation 7, “ $\supseteq$ ” can be replaced with “ $=$ ”, if

$$\forall e : e \in C_g \rightarrow \exists i : e \in C_i,$$

i.e., if there are no elements which are classified by  $C_g$  but not by any  $C_i$ . The latter holds for all natural objects which always have proper type which is more specific than a generalised type, but may not be true in modelling or programming, where instances of generalised types may be created unless they are declared as being “abstract”.

## 2.2 Similarities

When taking a sufficiently broad view, it is indeed possible to identify a number of similarities between classification and generalisation.

### 2.2.1 Abstraction

Classification and generalisation can both be regarded as abstraction mechanisms. By abstracting away from individual detail they give rise to relationships that are typically many-to-one,<sup>2</sup> i.e., many elements are abstracted from to yield one representative. By using the representative, i.e. the type or the generalisation,<sup>3</sup> one is able to assert facts about a large number of elements, e.g. what relationships they may engage in, without referring to individual elements. Hence, both classification and generalisation help to reduce the complexity of specifications.

### 2.2.2 Membership

Instances and subtypes can both be viewed as belonging to or being members of their respective representative. Each instance is a member of the set characterised by its type and each subtype is a member of the subtype hierarchy of which the generalisation forms the top.

### 2.2.3 Description

Obviously, the representatives can be regarded as *describing* their members, i.e., the members are at least partially defined by virtue of their membership. Consider aggregation, as an example for another many-to-one relationship, that does not have such a descriptive flavour. The *whole* can be used as a representative of its *parts*, but does not describe the latter.

### 2.2.4 Reuse

Finally, both types and generalisations can be usefully kept in libraries as they allow modellers to derive new elements from existing ones, as instances or subtypes respectively. They, therefore, both support reuse and incremental development in the sense that a modeller may reuse such library elements and only needs to specify what is different about the new element.

## 2.3 Differences

While the previous section appears to suggest that classification and generalisation have a lot in common, it actually refers to rather superficial similarities which distract from the fundamental differences between them.

### 2.3.1 Abstraction

Everything that has been stated in section 2.2.1 regarding the nature of classification and generalisation as abstraction mechanisms regarding the reduction of complexity can also be stated about aggregation, leaving only their descriptive nature and utility in libraries as differences. The vast differences between, say generalisation and aggregation, highlight what little significance a commonality in terms of “supporting abstraction” actually has.

<sup>2</sup>Many-to-many forms, known as multiple inheritance and multiple classification, exist but are not very commonly supported.

<sup>3</sup>I refrain from using “supertype” as the latter term implies that the element obtained by generalising has a type role.

### 2.3.2 Membership

While types and generalisations may both be regarded as representatives, they are in fact at different logical language levels with respect to each other. In section 2.1, I used expressions 4 & 5 to demonstrate the difference between properties at the instance level and the type level. Note that when using an abstract object (a generalisation) we can directly assert a property as in expression 4. To achieve the same with a type, we actually need to use universal quantification as in

$$\forall x : \iota(Dog)(x) \rightarrow HasFourLegs(x).$$

This universal quantification is often left implicit, using the pragmatic assumption that assertions are made about instances of a type, rather than the type itself (expression 5 being an example for the latter).

Yet, the above must not detract from that fact that  $|C|_{\sim}$  refers to *all elements* within the equivalence class implied by  $\sim$ , whereas  $\varepsilon(C)$  refers to the *set* of all elements, i.e., the equivalence class *itself* (given a corresponding  $\iota(C)$ ). Hence, when associating meaning to “ $|C|_{\sim}$ ” and “ $C$ ” by using a mapping “ $\mu$ ”, “ $\mu(|C|_{\sim})$ ” is multi-valued, i.e., here “ $\mu$ ” is a relation, whereas “ $\mu(C)$ ” has a single result, i.e., here “ $\mu$ ” is functional. Assuming a stratification of values in which sets of objects rank higher than the objects they contain (see Russell’s Theory of Types (Whitehead & Russell (1910))) clearly, types are at a higher logical level than their instances and hence also at a higher level than generalisations of their instances.

From the fact that the result of classification is a set (rather than all the members of the set), it follows that classification gives rise to a relation which is not transitive. While we have equation 6 for generalisation, and hence transitivity, i.e.,

$$C_1 < C_2 \wedge C_2 < C_3 \rightarrow C_1 < C_3,$$

for classification, obviously an element  $C_1$  with  $C_1 \in \varepsilon(C_2)$ , need not be in an element in a set  $C_3$ , even if  $\varepsilon(C_2) \in C_3$ . Figure 1 uses a 3D variant of a Venn diagram to illustrate the fact that an element (Lassie) is automatically also an (indirect) member of the super-type of its type, but is not automatically a member of the type of its type.

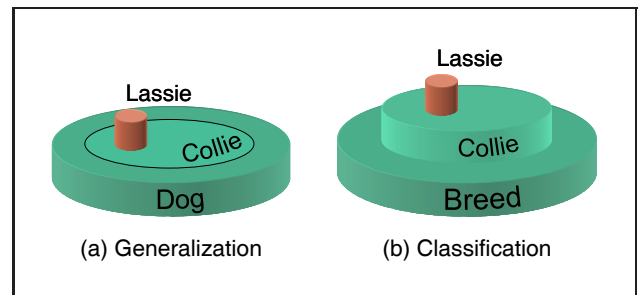


Figure 1: Differences in transitivity

In (Kühne (2006)), I argued that in contrast to classification, generalisation cannot be used to erect a *metalevel* hierarchy because of the transitivity of the relation it implies.

### 2.3.3 Description

Referring to the previous section 2.2 again, it is true that types and generalisations both have a descriptive role. However, note that while a type typically only shapes the *instance facet* of its instances, i.e.,

controls its instances' properties, a generalisation is typically only used to shape the *type facet* of its subordinated elements, i.e., supertypes are typically used to make subtypes inherit type facet features (such as the attribute "age : Integer"). It is, however, possible to influence the type facet with types (see "deep instantiation" (Kühne & Schreiber (2007))) and influence instance facets with specialisation (see Smalltalk "class variables" (Goldberg & Robson (1983))).

### 2.3.4 Reuse

As a result of their different descriptive roles, types and generalisations have rather different purposes when used as library elements. Types provide a vocabulary that is used without being refined. There is no specification of a "difference" to the derivable element, but one simply provides values for the schema made available through the type. Generalisations, on the other hand, are refined when used by specifying what has to be added to derive a specialised element from a general one.

In summary, although classification and generalisation share some superficial properties, they are intrinsically and unmistakably different.

## 3 Analysis of Approaches

In the following I examine a number of approaches that see benefits in de-emphasising the differences between classification and generalisation in one way or the other.

### 3.1 In the Name of Simplicity

Alloy is a language for the specification of software systems (Jackson (2006)). One of the tutorials on Alloy contains the following statement:

*"Set membership and subsets are both denoted in."* (Seater & Dennis (2008)).

In other words, Alloy uses one "in" operator for both " $\in$ " and " $\subseteq$ ". This is surprising at first because of the fundamental differences between these two relations. As discussed earlier on, " $\subseteq$ " (corresponding to generalisation) is transitive, whereas " $\in$ " (corresponding to classification) is not.

This apparent puzzle is easily resolved by observing that Alloy does not fully support modelling at the instance level. The modeller is rather required to model instances as singleton sets, as in

```
one sig Lassie extends Collie {}
```

Here the set of all collies is specialised by a singleton set<sup>4</sup> `Lassie` which is used to uniquely reference the intended instance "Lassie". This way the "in" operator can be considered to only support the " $\subseteq$ " interpretation. Checking

```
Lassie in Collie
```

yields true, because the set `Lassie` indeed contains a (unique) element which is also a member of the set `Collie`.

The good news is that, hence, "in" does not really confound the set membership and subset relations as quoted above. This confusion may still be claimed regarding a modeller's intention but technically "in" always corresponds to " $\subseteq$ ".

The bad news is, however, that representing instances as singleton sets

- can be very confusing for novices, and

<sup>4</sup>A set with exactly one and thus unique instance.

- denies the modeller the ability to distinguish between a singleton set and its instance.

Novices will read "in" to mean " $\in$ " in situations like this one:

```
Lassie in House
```

when trying to check whether Lassie is at home, i.e., test whether "Lassie" is among the elements of the set "House" while they are actually checking whether the (singleton) set `Lassie` is a subset of the set `House`.

Strangely, the below expression, using `some` (" $\exists$ ") to extract an element `x` from the set `Lassie`,

```
some x : Lassie | x in House
```

also yields true, although there is no element `x` in `Lassie` which is a subset of `House`. Although this may suggest, that here "in" is interpreted as " $\in$ " after all, Alloy interprets the unique instance within `Lassie` as the singleton set containing "Lassie". This also takes place when referring to generated instances, such as "Collie\$0", which are converted into singleton sets, e.g., "{Collie\$0}".

Furthermore,

```
some x : House | x in House
```

yields true, if the house is not empty, again strongly suggesting that "in" is interpreted as " $\in$ ". Yet again, however elements in `House` are converted to their singleton sets before the `in` test, so that the actual " $\subseteq$ " test yields the expected result.

In total, even experienced modellers need to be very wary in order to avoid misreading Alloy specifications that involve instances. Sometimes an Alloy expression (e.g., `Lassie`) appears to denote an instance, as it is used to uniquely reference a certain element, and sometimes it clearly is used as a set (as in `some x : Lassie | ...`). While there is always a consistent technical reading of "in" as " $\subseteq$ ", some of its usages are highly suggestive of an " $\in$ " interpretation. Understanding Alloy's results thus requires an understanding of its implicit conversion of elements to their respective singleton sets. Of course, it could be argued that the latter is not necessary and that Alloy manages to always associate the intended meaning of either " $\in$ " or " $\subseteq$ " to `in`, but this implies that one concedes to a blurring between classification and generalisation which is potentially dangerous (the intention could deviate from the actual meaning) and inappropriate for novices that have not yet been sufficiently exposed to a proper distinction between classification and generalisation. The latter is a problem when using Alloy in first year courses such as devised by Noble et al. (Noble et al. (2008)).

In section 2.3, I already pointed out the difference between properties at the instance level and at the type level (see expression 4 versus expression 5). Due to Alloy's approach to representing instances with singleton sets, the modeller loses the ability to separate these two levels of properties. Technically, it is an error to ask an instance for its member count since it is not a container type like a set, but an Alloy representation of an instance will happily answer "1". This may be regarded as a feature rather than a bug since it enables specifications which are agnostic as to whether they are dealing with instances or sets. However, this convenience comes with a price because one loses the ability to detect (i.e. type check for) erroneous data flows which lead instances to appear in places where only sets should occur and vice versa.

The rationale given for Alloy's treatment of instances as singleton sets, and the corresponding apparent unification of " $\in$ " and " $\subseteq$ " to "in", is the desire to uniformly allow the application of a single operator



to both scalars (instances) and sets (Jackson (2006)). Overall, Alloy represents a highly elegant language design which enables very concise and readable specifications. However, I believe that prioritising simplicity (just one “in” operator) and uniformity (applicable to both instances and sets) does not justify the loss in expressiveness and clarity as discussed above.

One could maintain the uniformity requirement of having just one “in” operator by overloading it, i.e., using the same syntax for “ $\in$ ” and “ $\subseteq$ ” but retaining their different meaning depending on the types of the arguments. Yet, this would exclude the possibility of detecting type errors resulting from an unintended usage of a set in place of an instance and vice versa.

I claim that the difference between an element and a set, including the set that only contains said element, is big enough in order to abandon simplicity and uniformity in favour of improved type checking capabilities. With respect to Alloy’s treatment of instances, I therefore propose to allow the direct modelling of instances and to use two different operators for “ $\in$ ” and “ $\subseteq$ ”.

### 3.2 In the Name of Flexibility

Motivated by the necessity to define the meaning of metalevel boundaries in metalevel hierarchies and in order to support sanity checks for the integrity of such hierarchies, Atkinson and Kühne have proposed a *strict metamodelling* doctrine. According to the latter it is possible to fully understand each level in a metamodelling hierarchy as being instantiated from only the level above it (Atkinson & Kühne (2001)). In order to enforce this property, the only relationships allowed to cross metalevel hierarchy boundaries are “instance-of” relationships.

Subsequently, this doctrine has sometimes been criticised as leading to inflexible infrastructures, and approaches have been developed that relax the strictness requirement in order to provide more flexible metamodelling infrastructures (Gitzel & Merz (2004), Varró & Pataricza (2003)). In particular, Varró and Pataricza take issue with the strict four-layer architecture of the OMG in that it leads to scenarios in which “...*concepts are replicated both on meta-level and model-level*...” (Varró & Pataricza (2003, p. 191)). As a remedy they advocate the introduction of “refinement” as a unification of the notions of instantiation and specialisation, regarding the latter as being highly compatible with each other:

*“As a result, two model elements can simultaneously be in subtype and instance-of relations...”* (Varró & Pataricza (2003, p. 194)).

Varró and Pataricza even provide a proof for this proposition (Varró & Pataricza (2003, p.195–196)). Their proof relies on the fact that classification and generalisation both give rise to many-to-one relations and that there are pairs of models which *can* be viewed as being in a classification *or* a generalisation relation. However, while Varró and Pataricza would read the aforementioned “or” as a logical “or”, I maintain that it must be read as a logical “xor”. Note that their example using “Graph” and “BipartiteGraph” (Varró & Pataricza (2003, Fig. 6)) excludes attributes. If elements of “Graph” defined attributes—e.g., “Node” could have the attribute “outDegree”—one could clearly see that in the instantiation case the nodes of “BipartiteGraph” would have *values* for “outDegree”, whereas in the specialisation case the nodes of “BipartiteGraph” would inherit the “outDegree” attribute. An obvious solution to the “attribute” dilemma is that nodes of “BipartiteGraph” have both

“outDegree” values and attributes, but this is most certainly not the intended structure.

Furthermore, if one considered not only model pairs, but deeper derivation structures, the difference in transitivity between classification and generalisation would become apparent.

For these reasons, I consider it inappropriate to view instantiation and specialisation as incarnations of a unified refinement notion that may occur simultaneously between two models.

Gitzel and Merz also aim to reduce the number of concepts in metamodelling hierarchies (Gitzel & Merz (2004)). They model “JavaAccount” as an instance of “Account” using a new form of “instance-of” (classification) relationship which “...*is used in a similar fashion to inheritance relationships*...” (Gitzel & Korthaus (2004, p. 72)). In essence, they are relaxing the strictness doctrine (Atkinson & Kühne (2001)) to allow instantiation across several metalevel boundaries. However, as is apparent from the “Account” / “JavaAccount” example, their new form of “instance-of” relationship in fact has specialisation semantics as opposed to instantiation semantics. Intuitively, every instance of “JavaAccount” should also (indirectly) be an instance of “Account”. Also, having elements at one metamodelling level that are instantiated from several different metamodelling levels higher up is incompatible with the requirements for a metamodel hierarchy erecting relation (Kühne (2006)), and as a matter of fact, with Russell’s Theory of Types (Whitehead & Russell (1910)).

In ontological metalevel hierarchies (Atkinson & Kühne (2003)), it is obvious that instantiation has to be anti-transitive and instantiation may only occur from one level to an adjacent one. Here is an ill-formed syllogism that violates this rule, representing a logical fallacy:

Man is a species
Socrates is a man
∴ Socrates is a species

If “species” is replaced with “mammal” then the syllogism works as intended because then the first “is a” corresponds to generalisation as opposed to classification. The above ill-formed syllogism illustrates the inappropriateness of assuming that an instance (Socrates) could be classified by an element that is two levels higher up in the metalevel hierarchy (species).

Gitzel et al. appear to require certain concepts at more than one metamodelling level since there are metamodelling hierarchies which cannot be aligned with each other (Atkinson & Kühne (2001)). In such cases, which include primitive types like “Integer”, strictness can be maintained for the hierarchies individually, and a single hierarchy may be used multiple times in conjunction with another one. I believe that this is an acceptable form of replication since it corresponds to “multiple usage” rather than “multiple definition”.

Gonzalez-Perez and Henderson-Sellers also appear to treat classification and generalisation as closely related relationships because they let both cross metalayer boundaries in parallel (Gonzalez-Perez & Henderson-Sellers (2006, p. 88, Fig. 20)), but note that their *metalayers* are not aligned with *metalevels*, the latter being defined by classification relationships only. Although their usage-oriented *layering* appears to blur the differences between classification and inheritance, the underlying *level hierarchy* does not suffer from any such conceptual difficulties.

Summarising, with respect to attempts to relax the strictness doctrine, I argue that there is no value

in weakening the significance of metalevel boundaries. On the contrary, abandoning strictness opens the door for errors that no longer can be detected as such. If there is value in partitioning a metamodeling hierarchy along boundaries other than the metalevel boundaries, such structures should be overlaid, or offered as alternative views, but not undermine the integrity of the metalevel boundaries themselves.

### 3.3 In the Name of Utility

In order to promote consistency and parsimony, the OMG introduced a “Core” model from which both the Meta-Object Facility (OMG (2006)) and the UML definition (OMG (2007)) are derived. Regarding the UML definition, note that it is both specialised<sup>5</sup> from the Core (OMG (2004, p. 12, Fig. 7.2)) and also instantiated from the Core<sup>6</sup> (OMG (2004, p. 14, Fig. 7.4)).

Formally, we both have UMLA Core (Core classifies the UML definition) and UML < Core (Core generalises the UML definition). In section 3.2, I have already argued that this is inappropriate, i.e., impossible to maintain in a sound manner. However, I was making the assumption that both models are used with an ontological interpretation, which was appropriate regarding Varró and Pataricza’s examples. As observed by Atkinson and Kühne, however, the OMG uses the Meta-Object Facility (MOF) and hence by implication the Core, both as an ontological type model and as a linguistic type model (Atkinson & Kühne (2005, p. 409, Fig. 15(b))). In the following, I will investigate under which circumstances it is possible for a linguistic type model to be both the type model and a supermodel for another model.

Formally, we are looking for elements  $M$  (UML) and  $MM$  (Core) which fulfil the following constraint:

$$M \in \varepsilon(MM) \wedge \varepsilon(M) \subseteq \varepsilon(MM). \quad (8)$$

Assuming an element  $m$  (a UML model), the above constraint with  $\varepsilon(M) = \{m\}$  and  $\varepsilon(MM) = \{m, M\}$ , yields

$$M \in \{m, M\} \wedge \{m\} \subseteq \{m, M\} \quad (9)$$

which fulfils the constraint. The following observations are noteworthy:

- The only way in which constraint 8 may (non-trivially) be true is for an  $M$  that has a type role. If  $M$  were an instance without a type role, i.e.,  $\varepsilon(M) = \emptyset$ , it could not meaningfully take the place of  $M$  in constraint 8 since it would not have an extension whose elements may also appear in the extension of  $MM$ ; its extension could not be non-trivially considered to be a subset of  $MM$ ’s extension.
- From constraint 8 it follows that  $\varepsilon(MM)$  must contain two elements which can be considered as being in a classification relation. Formally,

$$\begin{aligned} M \in \varepsilon(MM) \wedge \varepsilon(M) \subseteq \varepsilon(MM) \wedge \varepsilon(M) \neq \emptyset \rightarrow \\ \exists m : M \in \varepsilon(MM) \wedge m \in \varepsilon(MM) \wedge m \in \varepsilon(M). \end{aligned}$$

Hence,  $MM$  must (at least implicitly) define a notion of instantiation (some of) its elements. Indeed, the Core/MOF defines instantiation between its elements.

<sup>5</sup> Actually, the term “dependent on” is used which refers to package usage which can be appropriately regarded as specialisation.

<sup>6</sup> Actually, it is instantiated from the MOF which contains the Core.

- The above point implies that  $MM$  cannot only instantiate  $M$ —as well as being  $M$ ’s supermodel—but also  $m$ , an instance of  $M$ . In fact, this is the case in our example, as the MOF/Core can (linguistically) classify UML models. This ability is used to provide a common repository/interchange format for UML models.
- If  $M$  and  $MM$  are chosen to be equivalent, i.e.,  $\iota(M) \sim \iota(MM)$ , and hence describe the same models, ignoring the implied different logical levels in a stratified scheme, then  $MM$  can be said to be *self-describing*. This is useful to obtain a self-terminating metamodeling hierarchy, such as the OMG’s four layer architecture.

With respect to the last point, note that choosing  $M = MM$ , would lead to a set that contains itself, i.e. an infinite regression, by expressions 8 & 9. This is not possible assuming ZFC set theory (Kunen (1980)), but operationally one could of course construct a metamodeling hierarchy which features as many copies of the top-level as desired. One could use an *MM-quine*, i.e., an  $MM$  that contains a quoted version of itself. Upon instantiation one would unquote this version and supplement it with a quoted version again. This is the principle with which one can write programs that replicate/output themselves.

Note that Core is used as a linguistic type model and can instantiate itself. This makes it possible to avoid the logical stratification problems which were mentioned in section 3.2 as follows: The linguistic Core type model can instantiate both the UML definition and another Core instance, the latter acting as a supermodel for the UML definition. This way the UML definition can simultaneously be an instance and a subtype of Core, thus being shaped in two ways, albeit referring to two (identical) instances of Core. In contrast to the “BipartiteGraph” example of section 3.2, it does make sense for the UML definition to have features for its elements—such as “name”—that exist both as properties (with values) and as attributes (shaping UML instances, i.e., user models).

As a (surprising!) result, it is indeed possible to increase the utility of Core/MOF by using it both as a repository format (linguistic type) and a language definition supermodel (language definition supertype) while maintaining a clean structure free of inconsistencies. I propose to use the above developed interpretation of the dual role of the Core/MOF as an underpinning to its intended dual purpose.

## 4 Conclusion

Classification and generalisation can be regarded as sharing a number of properties. The aim of this paper was to demonstrate that claiming any resemblance between the two abstraction mechanisms is only meaningful when taking a very broad view. Since Frege’s pioneering work in 1879, there should be no doubt regarding the fundamental differences between classification and generalisation and the resulting incompatibility regarding a simultaneous usage.

Apart from recalling this important lesson learned, this paper makes a number of contributions:

First, I used the notion of an abstract object to define generalisation on instances, i.e., avoiding the usual view of specialisation as extending type facets. This provided an unusual interpretation of supertypes as generalising the instance facets of their subtypes as well as their type facets. First and foremost, however, it allowed an adequate comparison between classification and generalisation which focussed on intrinsic differences rather than on discrepancies that result from different usage scenarios.

Second, I pointed out some conceptual problems and inconsistencies of approaches which use a unifying view on classification and generalisation, de-emphasising their differences. Whenever needed and possible, I offered a logically consistent view, restoring an adequate separation of classification versus generalisation. To the best of my knowledge, in particular the formal investigation into whether the OMG's view of the Core/MOF as both a repository format and a language definition supermodel has a sound interpretation, is novel.

Third, I argued that acknowledging the differences between classification and generalisation—e.g., by complying with the “strict metamodeling” doctrine—one gains an opportunity for sanity checks regarding the integrity of metamodeling hierarchies that otherwise would not exist. Detecting incorrect dataflow in specifications becomes possible if instances and types are not substitutable for each other. Logical fallacies that are introduced by crossing multiple metalevel boundaries at once or introducing circular definitions are impossible, if metalevels are stratified according to Russell's Theory of Types. Finally, a metamodel can be defined to have a dual purpose in a sound manner, if it features elements that can classify each other.

Ultimately, there is no single correct way of designing languages and in particular Alloy's prioritisation of flexibility over safety can certainly be defended. Also, I am aware that the authors of the work which I subjected to some critical remarks may take different definitions for classification and generalisation as a basis and hence arrive at different conclusions regarding their compatibility, maintaining internal consistency.

However, I hope that the observations made in this paper may be of use for future language designers. While they may not choose to subscribe to the most rigorous treatment I have suggested, they will at least be in a position to consciously deviate from it, explicitly rationalising as to why a non-strict treatment should be preferred and whether it is worth accepting the resulting loss in precision and loss of sanity checks.

## Acknowledgements

I would like to thank Lindsay Groves for his very helpful comments on a draft of this paper.

## References

- Atkinson, C. & Kühne, T. (2001), ‘Processes and products in a multi-level metamodeling architecture’, *International Journal of Software Engineering and Knowledge Engineering* 11(6), 761–783.
- Atkinson, C. & Kühne, T. (2003), ‘Rearchitecting the UML infrastructure’, *ACM Transactions on Modeling and Computer Simulation* 12(4), 290–321.
- Atkinson, C. & Kühne, T. (2005), Concepts for comparing modeling tool architectures, in L. Briand, ed., ‘Proceedings of the ACM/IEEE 8<sup>th</sup> International Conference on Model Driven Engineering Languages and Systems, MoDELS / UML’, Springer Verlag, pp. 398–413.
- Carnap, R. (1947), *Meaning and Necessity: A Study in Semantics and Modal Logic*, University of Chicago Press.
- Frege, G. (1892), Über Begriff und Gegenstand (On Concept and Object), in ‘Vierteljahrsschrift für wissenschaftliche Philosophie’, Vol. XVI, Fues's Verlag, pp. 192–205.
- Gitzel, R. & Korthaus, A. (2004), The role of meta-modeling in model-driven development, in ‘Proceedings of the 8<sup>th</sup> World MultiConference on Systemics, Cybernetics and Informatics’, Vol. IV, pp. 68–73.
- Gitzel, R. & Merz, M. (2004), How a relaxation of the strictness definition can benefit MDD approaches with meta model hierarchies, in ‘Proceedings of the 8<sup>th</sup> World Multi-Conference on Systemics, Cybernetics and Informatics’, Vol. IV, pp. 62–67.
- Goldberg, A. & Robson, D. (1983), *Smalltalk-80: The Language and its Implementation*, Addison-Wesley, Reading, MA.
- Gonzalez-Perez, C. & Henderson-Sellers, B. (2006), ‘A powertype-based metamodeling framework’, *Software and Systems Modeling* 5(1), 72–90.
- Jackson, D. (2006), *Software Abstractions: Logic, Language, and Analysis*, The MIT Press, Cambridge, Mass.
- Kamlah, W. & Lorenzen, P. (1996), *Logische Propädeutik*, Metzler.
- Kühne, T. (2006), ‘Matters of (meta-) modeling’, *Software and Systems Modeling* 5(4), 369–385.
- Kühne, T. & Schreiber, D. (2007), Can programming be liberated from the two-level style? – Multi-level programming with DeepJava, in ‘Proceedings of the 22<sup>nd</sup> annual ACM SIGPLAN conference on Object oriented programming systems and applications’, ACM, NY, USA, pp. 229–244.
- Kunen, K. (1980), *Set Theory: An Introduction to Independence Proofs*, Elsevier. ISBN 0-444-86839-9.
- Mittelstraß, J., ed. (1995), *Enzyklopädie Philosophie und Wissenschaftstheorie*, Verlag J. B. Metzler.
- Noble, J., Pearce, D. J. & Groves, L. (2008), Introducing Alloy in a software modelling course, in ‘ETAPS 2008 Workshop on Formal Methods in Computer Science Education (FORMED)’.
- OMG (2004), *Unified Modeling Language Infrastructure, Version 2.1.2*. OMG document formal/2007-11-04.
- OMG (2006), *Meta Object Facility (MOF) 2.0 Core Specification*. OMG document formal/2006-01-01.
- OMG (2007), *Unified Modeling Language Superstructure Specification, Version 2.1.1*. OMG document formal/07-02-05.
- Purchotius, E. (1730), Institutiones philosophicae I, in ‘Tomus primus, Complectens Logicam & Metaphysicam’, Apud Joannem Manfrè.
- Rayside, D. & Campbell, G. T. (2000), An Aristotelian understanding of object-oriented programming, in ‘Proceedings of the Conference on Object-Oriented Programming, Systems, Languages, and Applications’, ACM Press, pp. 337–353.
- Seater, R. & Dennis, G. (2008), ‘Tutorial for Alloy analyzer 4.0’, <http://alloy.mit.edu/alloy4/tutorial4/>.
- Ungar, D. & Smith, R. B. (1987), Self: The power of simplicity, in ‘Proceedings of the Conference on Object-Oriented Programming, Systems, Languages, and Applications’, ACM press, pp. 227–242.

- Varró, D. & Pataricza, A. (2003), 'VPM: A visual, precise and multilevel metamodeling framework for describing mathematical domains and UML', *Journal of Software and Systems Modelling* **2**(3), 1–24.
- Whitehead, A. N. & Russell, B. (1910), *Principia Mathematica*, Suhrkamp, Frankfurt.

# Extracting and Modeling the Semantic Information Content of Web Documents to Support Semantic Document Retrieval

Shahrul Azman Noah<sup>1</sup>, Lailatulqadri Zakaria<sup>1</sup> & Arifah Che Alhadi<sup>2</sup>

<sup>1</sup>Faculty of Information Science & Technology  
Universiti Kebangsaan Malaysia  
43600 UKM Bangi Selangor MALAYSIA

<sup>2</sup>Department of Computer Science  
Universiti Malaysia Terengganu  
21030 Kuala Terengganu, Terengganu, MALAYSIA

samn@ftsm.ukm.my, laila@ftsm.ukm.my, arifah\_hadi@umt.edu.my

## Abstract

Existing HTML mark-up is used only to indicate the structure and lay-out of documents, but not the document semantics. As a result web documents are difficult to be semantically processed, retrieved and explored by computer applications. Existing information extraction system mainly concerns with extracting important keywords or key phrases that represent the content of the documents. The semantic aspects of such keywords have not been explored extensively. In this paper we propose an approach meant to assist in extracting and modeling the semantic information content of web documents using natural language analysis technique and a domain specific ontology. Together with the user's participation, the tool gradually extracts and constructs the semantic document model which is represented as XML. The semantic models representing each document are then being integrated to form a global semantic model. Such a model provides users with a global knowledge model of some domains.

**Keywords:** ontology, information retrieval, semantic document retrieval, semantic information extraction.

## 1 Introduction

Accessing and extracting semantic information from web documents is beneficial to both humans and machines. Humans can browse and retrieve documents in a semantically manner whereas machine can easily process such structured representations. Furthermore integrating extracted information from multiple documents can provide users with a global knowledge model of some domains. Due to the structure of human knowledge, the tasks of extracting semantic information in web documents, however, proved to be difficult. The vision of Semantic Web (Berners-Lee et al, 2001) offers the possibility of providing the meanings or semantics of web

documents in a machine readable manner. However, the vast majority of 1.5 billion web documents are still in human readable format, and it is expected that this form of representation will still be the choice among content creators and developers due to its simplicity. Due to this phenomenon and the desire to make the Semantic Web vision a reality, two approaches have been proposed (van Harmelen & Fensel, 1999): either furnish information sources with annotations that provide their semantics in a machine accessible manner or write programs that extract such semantics of Web sources.

This research falls into the latter category, whereby the intention is to develop a semi-automated tool meant to assist in extracting and modeling the semantic information content of web documents using the natural language analysis (NLA) technique and a domain specific ontology. In this approach a set of candidate concepts (key phrases or keywords) is automatically extracted from web documents using heuristic rules. Sentences which relate with these concepts are then analyzed and compared with the domain ontology to construct the semantic information content. This process might be performed with the user's participation depending on the domain ontology. Each semantic domain model of a domain is then integrated together to form the global semantic document model. The approach discussed here might be very much similar to another of our work on semantic document retrieval (Noah et al, 2005). However, the focus of this paper is more on the extraction aspect to represent the semantic information of a web document.

This paper is organized into the following sections. The next section provides the background and related research. Section 3 explains the approach employed in extracting and modeling of the semantic information content of web documents. Section 4 and 5 respectively present the testing results and the conclusion that can drawn from our work.

## 2 Background and Related Research

The aim of information extraction (IE) is to collect the information from large volumes of unrestricted text. IE isolates relevant text fragments, extracts relevant information from the fragments, and pieces together the targeted information in a coherent framework (Cowie & Lehnert, 1996). IE problems have been popularly dealt with NLA techniques. However, a few research have

considered using domain ontology (Uren et al., 2006; Villa et al., 2003). We provide some background knowledge of NLA and ontology; and then proceed with some related works.

## 2.1 NLA and the Semantic Web

NLA is the study of understanding human natural language such that it can be understood and correctly processed by machines. Within the vision of Semantic Web, although, NLA contributions are not directly explicated (Berners-Lee et al., 2006), the technology can do play an important in this very slow but progressing semantic technology. NLA for instance can automatically create annotations from unstructured text that provides data which semantic web applications require (Pell, 2007). Research has also been done on providing an NLA type of interface for describing a semantic content which is then translated into one of the Semantic Web enabling technology i.e. Resource Description Framework (RDF) and Web Ontology Language (OWL) (Schwitter, 2005). Another potential application of NLA to Semantic Web is in terms of annotation. Interestingly there are two very different types of annotation process involving NLA, one is annotating natural language document such HTML documents with a pre-specified domain ontology (Vargas-Vera et al., 2002) and the other is annotating document (can be natural language documents or semantic web documents) with natural language (Katz & Lin, 2002). The work by Vergas-Vera et al. (2002) involves pre-processing of HTML documents and semi-automatically annotate the identified concepts with domain ontology. They developed a tool called MnM. The work by Katz and Lin (2002) on the other hand allow users to augment RDF schema with natural language annotations to in order to make RDF more friendly to human instead of machine alone. The work reported in this paper falls into the earlier approach.

## 2.2 Ontology

Ontologies are widely used in knowledge engineering, artificial intelligence; as well as applications related to knowledge management, information retrieval and the semantic web. Among the first definition of ontology was provided by Neches et al. (1991); which said that “*an ontology defines the basic terms and relations comprising the vocabulary of a topic areas as well as the rules for combining terms and relations to define extensions of the vocabulary*”. However, the most quoted definition of ontology is based from Gruber (1993); which defined it as “*an explicit specification of a conceptualization*”. Although, there has been no universal consensus for the definition of ontology; the aim of ontology is very clear as put forward by Gomez-Perez et al. (2004) that is “*to capture consensual knowledge in a generic way, and that they may be reused and shared across software applications and by groups of people*”.

Ontology is considered as the backbone for the Semantic Web and received great attentions from researchers working in this area. Ontology has also been gradually seen as an alternative to enhance information retrieval task. However, the majority of efforts in information retrieval are limited to query expansion and

relevance feedback by exploiting the so-called linguistic ontology such as the WordNet (Miller, 1995). In this paper, we extend the use of ontology into a mediator for mapping concepts extracted from documents and to establish the semantic relationships among the concepts.

## 2.3 Related Work

We briefly discuss three research works which are very related to ours, which are the work by Embley et al. Embley (2004) and Embley et al. (1999); Brasethvik and Gulla (2001, 2002) and Alani et al. (2003).

The work by Embley (2004) use object relationship model, data frames and lexicon (which forms an ontology) to assist in data extraction from web documents. Brasethvik and Gulla (2001, 2002) employ NLA technique and a conceptual model to support the task of document classification and retrieval. In this case, the conceptual model is constructed by a committee from a set of sample documents by identifying the concepts and relationships. This model is then used for classification and retrieval. Alani et al. (2002) on the other hand develop Artequakt which is an information extraction system for extracting information about artist and artifacts using a domain ontology.

Our work is towards the development of a framework for extracting and modeling the semantic information content of web documents. Our work differs from those of Embley which mainly concerns on extracting ‘data’ that can be queried similar to SQL-like statement. Our work also differs to the work of Alani et al. (2003) which basically concerns with semantic annotation of web documents. Our work, however, share some similar ‘motivation’ with the work Brasethvik and Gulla (2001, 2002). The difference is that instead of using a conceptual model, we used existing domain ontology and allow interactive user-tool refinement in constructing the semantic model.

## 3 The Approach

Our approach to semantic information extraction of web documents involves constructing a semantic document model representing each processed documents. To support this process, the approach employs the natural language analysis (NLA) technique and a set of domain specific ontology. Both are used to perform the task of textual analysis which results not only in the identification of important concepts represented by the documents but also the relationships between these concepts. This approach, therefore, follows the general approach to building semantic index as illustrated by Desmontils and Jacquin (2001).

Figure 1 illustrates the overall process involved in constructing the semantic document model. As compared to the Brasethvik and Gulla (2001, 2002) approach which relies on the conceptual model previously constructed by a selective of personnel, our approach utilise existing ontology of the chosen domain, i.e. the medical domain. A detail discussion of the process therefore follows.

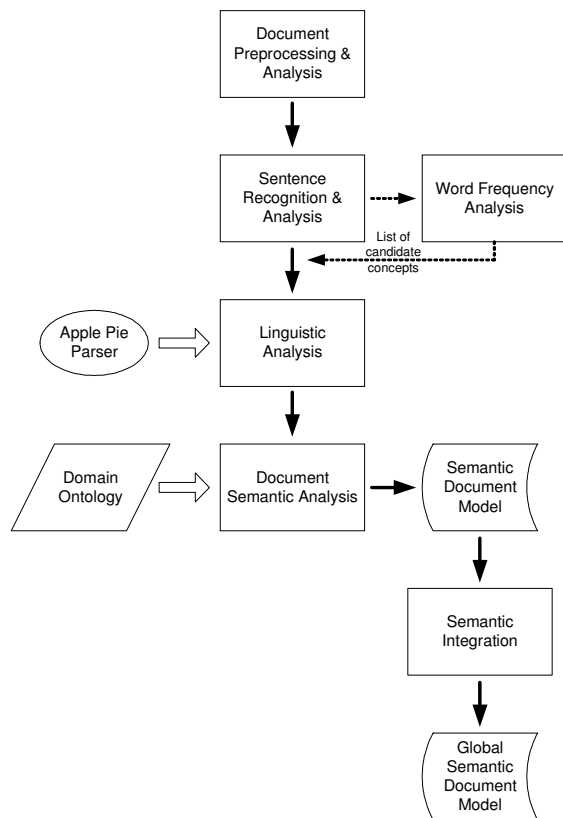


Figure 1. The construction of semantic document model

### 3.1 Document Pre-processing and Analysis

Every HTML documents sent for processing will firstly be decoded to generate ASCII document files type that are free from any HTML tags. The documents' content extracted from this process are the document's metadata such as the document's title, URL, description and keywords. The textual content of the documents is also extracted.

These documents will then undergo a word analysis process which involved document's filtering and words frequency calculation. In document's filtering, all stop words will be eliminated and selected concepts will be stemmed to their root words. These concepts or words will be sorted according to the frequency of appearance within the document. The sentence analysis and recognition on the other hand will divide the documents into paragraphs, which are in turn broken down into sentences and stored in the document sentences repository. A set of concepts with high frequency previously obtained from the word analysis process will be matched with the sentences stored in the repository in order to select the candidate sentences to be used in the next NLA process. According to Luhn (1958), extracted words with high frequency can represent document's content. The selection of sentences that contains high frequency concepts is entirely based on the heuristic which suggest that such sentences are best describe the content of documents. This heuristic also removes the needs to analyse all possible sentences found in the document which can jeopardize the processing performance of the system.

An example of a document pre-processing and analysis process is as illustrated in Figure 2. As can be seen, the results of this process is a list of potential candidate concepts (for building the semantic document model) sorted according to the number of occurrences – as well as a list of potentially rich information sentences in which the candidate concepts were found.

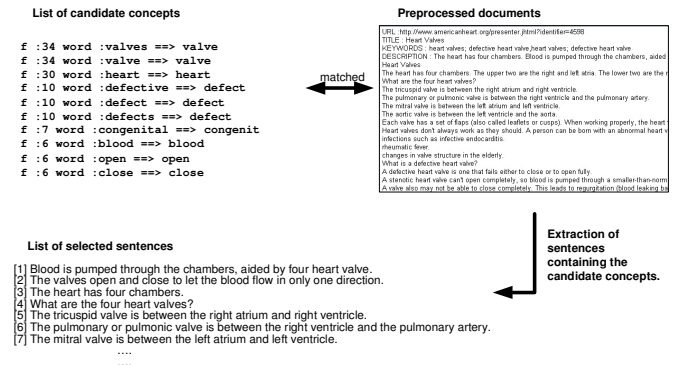


Figure 2. An example of the document analysis process

### 3.2 Natural Language Analysis

The natural language analysis process can be divided into two subsequent stages: the morphology and syntactic analysis; and the semantic analysis. The main aim of this process is to generate a local semantic document model representing each processed document. The morphology and syntactic analysis process will analyse the input sentences previously stored (sentences that contains candidate concepts) in the sentences repository into a parse tree using the Apple Pie Parser (Sekine, 2006). The parser is a bottom-up probabilistic chart parser which finds the parse tree with the best score by best-first search algorithm. The grammar used is a semi context sensitive grammar with two non-terminals and was automatically extracted from Penn Tree Bank, syntactically tagged corpus made at the University of Pennsylvania (Sekine, 2006).

The process of morphology and syntactic analysis is considered to be domain independent. For example the input sentence of "Blood is pumped through the chambers, aided by four heart valves", is being parse to the following parse tree.

```
(S
  (NPL Blood)
  (VP is
    (VP pumped
      (PP through
        (NP
          (NPL the chambers) -COMMA-
          (VP aided
            (PP by
              (NP
                (NPL four heart) valves))))))
          -PERIOD-))
```

The semantic analysis on the other hand performs the task of extracting the semantic relationships between the selected concepts. This is performed either by the use of domain specific ontology or by exploiting the semantic structure of the analysed sentences with the help of the



user. The interactions from user at this stage is seen acceptable as fully-automated approach to semantic analysis is not possible due to the requirement for deep understanding of the domain in concern (Snoussi et al., 2002). Figure 3 illustrates the overall process of this stage which indicates that the two main activities involved are the identification of concepts and the relationships between these concepts. Detail discussion of these activities therefore follows.

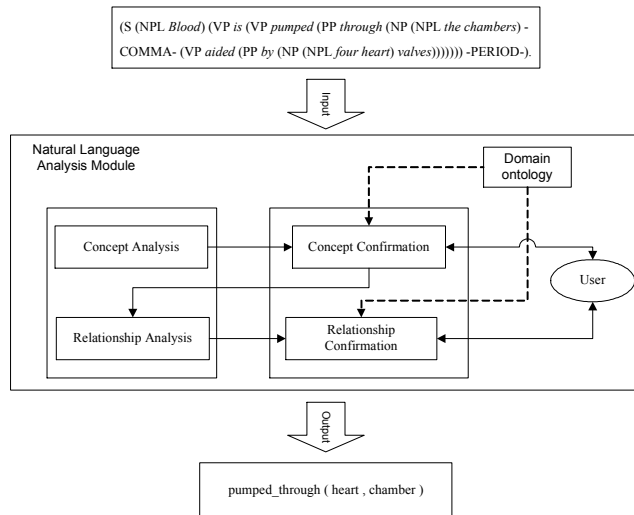


Figure 3. The natural language analysis process

Noun phrases and verb phrases are good indications of concepts to be included in the semantic documents model. Therefore, every noun phrases and verb phrases extracted from the analysed sentences are represented as concepts. These noun phrases will be analysed to filter determiners (such as *the*, *a* and *and*) that usually occur in word phrases.

For example, the parsed sentences of “*Blood is pumped through chambers aided by four heart valves*” in the form of (S (NPL Blood) (VP is (VP pumped (PP through (NP (NPL the chambers) -COMMA- (VP aided (PP by (NP (NPL four heart) valves)))))) -PERIOD-), will resulted in the extraction of the concepts: ‘*blood*’, ‘*the chambers*’ and ‘*four heart*’. The determiner ‘*the*’ and the stopword ‘*four*’ will be removed from the identified concepts.

The confirmation (in terms of the correctness) of the filtered concepts is performed in two ways; either automatically endorsed by referring to the domain ontology if the mapping between the concepts and the domain ontology existed; or from user intervention in the case where no such mapping found existed. Figure 4 illustrates a portion of the domain ontology used by the tool.

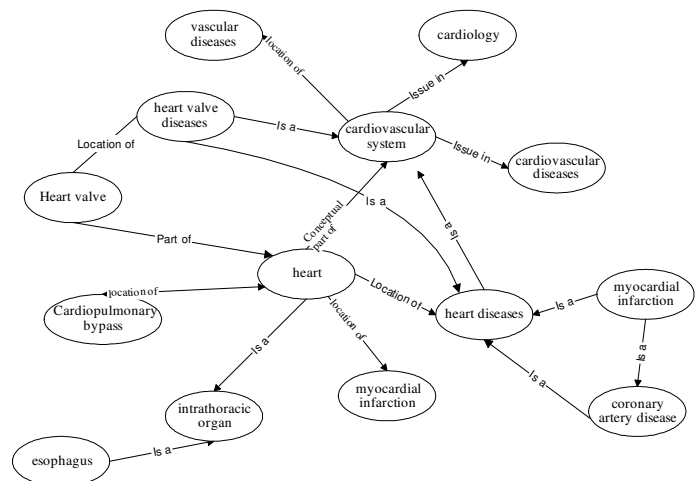


Figure 4. A segment of the heart domain ontology

Relationship recognition identified during the previous phase (concept recognition and confirmation) against concepts within the is done by comparing candidate concepts and concepts which were domain ontology. If a pair of concepts found matched with the domain ontology, the relation of these concepts is automatically defined by referring to the domain ontology if such a relation existed. If a relation does not exist, a suggestion is provided based upon the syntactic sentence structure of the associated concepts, of which the user will define it manually. Similarly, for those concepts not presented in the domain ontology, the tool will first provide a list of concept candidates which can best be linked based upon the analysis of the chosen sentences. Once the desired concepts have been selected, the tool will provide the suggestion of possible relationships between these concepts.

Figure 5 illustrates an example of a HTML document, a fraction of medical domain ontology and the output generated by the semantic document modeling tool. As can be seen from this example, the semantic relationships of “*mitral valve part-of heart*”, “*heart valve part-of heart*” and “*mitral valve is-a heart*” are all extracted from the domain ontology whereas the other concepts and relationships are extracted by means of text analysis with the user’s participation. The generated semantic document model is an XML representation of the concepts, relationships as well as the URL of the selected documents. Example below is part of the generated XML representation. This model is then stored in the Semantic Document Model.

```
<?xml version="1.0" encoding="UTF-8" ?>
<DocumentInfo>
<MetadataInfo>
<Title>Heart Valves</Title>
<Url>http://www.americanheart.org/presenter
.jhtml?</Url>
<Keywords>heart valves , heart , mitral
valve , aorta , blood , chambers , valves ,
blood flow , valve , flaps ,</Keywords>
</MetadataInfo>
<Semantic_Content>
<Concept>
<ConceptDescription>
```



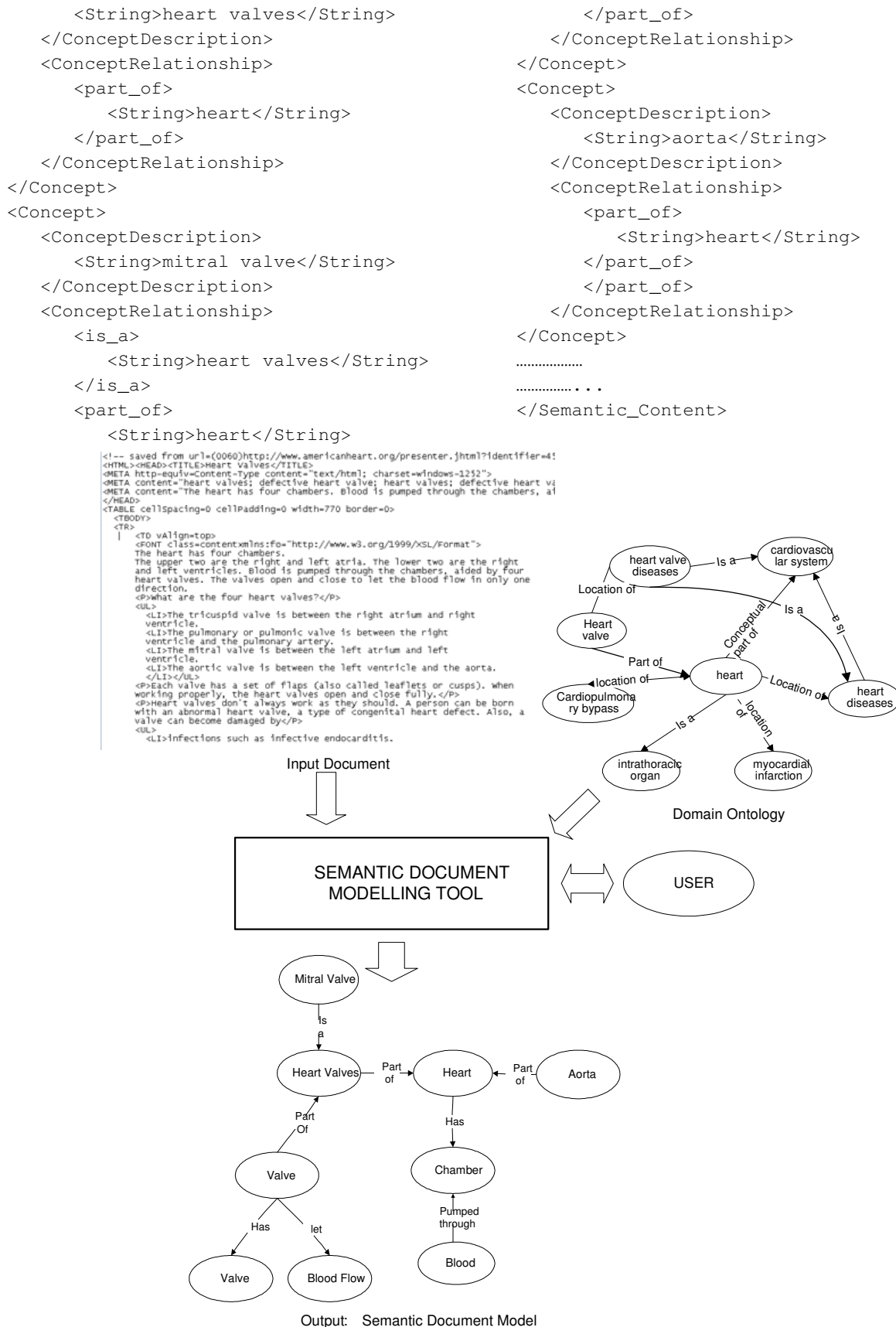


Figure 5. Constructing the semantic document model

### 3.3 Integration of Semantic Document Model

The stored semantic document models will then undergo an integration process which results in the creation of a global semantic document model. The global semantic model is meant to be used for semantic retrieval and browsing.

The semantic integration is an uncomplicated process requiring insertion of a new semantic document model to the existing global semantic document model. The process will remove aspects of redundancies and documents that belong to the same semantic concept are clustered together. A set of simple object type and mismatch rules which was mainly derived from the theory and technique of automatic conceptual modeling integration process (Batini et al., 1986); Noah & Williams, 2004) have been used. The rules are mainly for binary relationships as the semantic document model represented are binary in nature. At the moment aspects pertaining to the concepts of generalization, aggregations and associations of a semantic model are being considered. Naming conflicts such as synonyms and homonyms however are not being considered by the rules.

## 4 Results

Evaluation was done by comparing the extracted concepts in semantic document model with keywords in <META> tag provided by authors. Our assumption was that <META> tags provide key information or phrase reflecting document content. A similar method of evaluation has been conducted by Witten et al. (2000) and Song et al. (2004) which compare the extracted concepts with those human-generated keywords or key phrase in order to evaluate the performance of KEA and KPSPotter respectively.

The main inherent problem with this evaluation approach is the lack of web pages that provide <META> tags keywords which resulted in the limited number of available testing document collection. We have performed hundreds of document analysis but only 50 web documents were acceptable and sufficient enough for testing. Table 1 shows the result of average 'correct concepts' corresponding to the concepts assign by author (extracted from <META> tags).

Table 1 lists the average number of matched candidate concepts, system extracted concepts and concepts from the generated semantic document model for the 50 test web documents. Candidate concepts referred to concepts used for selecting potentially rich information sentences during the document analysis process. Ontological-based extracted concepts are concepts extracted from the domain ontology and used to generate portion of the semantic document model. The semantic document model concepts are concepts presented in the final generated semantic document model. In other words the semantic model concepts are the composition of matched concepts derived from the domain ontology and concepts confirmed from the activities of user interactions.

Concepts extracted	Average number of concepts matched with <META> tags		
	Candidate Concepts	Ontology-based extracted concepts	Concepts in generated semantic document model
1	0.75	0.52	0.56
2	1.6	1.14	1.18
3	2.32	1.48	1.64
4	3.14	1.66	1.96
5	4.18	1.84	2.26
6	4.93	1.92	2.48
7	5.93	1.98	2.76
8	6.72	2.06	3.12

**Table 1:** Overall Performance.

As can be seen, the average numbers of system extracted concepts with user interactions (that corresponds to the concepts assigned by authors) is 3.12. Therefore, system extracted concepts with user interventions capable of extracting one to three 'correct concepts'. System-extracted concepts achieved one to two correct concepts which is 2.06 in average. System extracted concepts depends fully on domain ontology for concepts identification and extraction limits to the stored information. Our implementation of domain ontology only stores 24 related concepts which actually represent a small fraction of the 22,997 terms listed in the Medical Subject Heading (MeSH) Concepts assigned by authors cover a wider range of concepts in domain ontology and sometimes go beyond the domain ontology itself. Adding more information/concepts into domain ontology may increase system efficiency in performing concepts identifications. System extracted concepts with user intervention achieved a better result because it allows user to describe web document content based on their understanding of the domain.

Based on the testing, our approach capable of extracting at between one to six correct candidate concepts. In other words for the first eight concepts an average of six concepts matched with those of authors' concepts (i.e. meta tags). However, for every eight concepts extracted of which six were selected as candidate concepts, only about three concepts were presented in the generated semantic document model. This difference is resulted by the following possibilities.

- The candidate concepts do not matched with any of the ontological domain concepts.
- The candidate concepts and the ontological domain concepts were not used to construct the semantic document model because their presence are not inherent in the filtered sentences.

Having one to three 'correct concepts' in the generated model does not indicate that other concepts are not representative of the domain. Our testing result, however, is higher than those reported by KEA and KPSPotter that respectively shows 1.8 and 2.6 extracted key phrases in average that match with author key

phrases. This technique of evaluation, however, does not consider semantic relationships between extracted concepts. Such ‘correctness’ of relationship is best judge independently by human beings.

The results of our testing might also be influenced by the way authors describe their respective documents with <META> tags, which may be summarize as follows:

- Authors do not always choose the best keywords or key phrases to reflect the content of their documents. In some cases, authors apply the same set of keywords for different documents.
- Authors, instead of using the same keyword or key phrases that they use in their documents, they replace them with other concepts which are similar or synonyms. As a result, some of those keywords are not available within their document.

#### 4 Conclusions and Future Works

Domain ontology plays an important role in supporting the tasks of document classification and organization. In this paper, we have presented how a domain ontology combine with a natural language analysis technique can be exploited not only to extract important concepts from documents but also to construct the semantic content of web documents.

Although, controlled vocabulary has been used in information retrieval systems (Embley, 1999), the vocabulary tends to be a list of terms that are syntactically matched with terms in documents. The inherent meanings or structures of the terms in the vocabulary are not used to represent the semantic meanings of documents, and users are still left with a syntactic approach to information retrieval. While ontologies for Semantic Web have been focus to support machines looking for information instead of human, semantic document model is intended to support human communication, which requires a human readable notation. In our case the constructed semantic document model is rather meant for later retrieval by human instead of machines or software agents. Our current research work is to enhance aspects of global semantic document integration by considering further integration aspects such as synonyms, homonyms and inheritance mechanisms which are very well established within the context of database conceptual modeling. Testing and evaluation of the approach presented in this study are also currently being carried out.

#### 4 References

- T. Berners-Lee, J. Hendler, and O. Lassila, The Semantic Web. *Scientific American*, May, pp. 35-43, 2001.
- F. van Harmelen, and D. Fensel, Practical knowledge representation for the Web. *IJCAI Workshop on Intelligent Information Integration*, 1999.
- S. A. Noah, A. C. Alhadi. and L. Zakaria, A semantic retrieval of web documents using domain ontology. *International Journal of Web Grid and Services*, pp. 151–164, 2005.
- E. Desmontils and C. Jacquin, Indexing a web site with terminology oriented ontology. *International Semantic Web Working Symposiums (SWWS)*, Stanford University, California, 2001.
- J. Cowie and W. Lehnert, Information Extraction. *Communications of the ACM*, 39(1), pp. 80-91, 1996.
- V. Uren, P. Cimiano, J. Iria, S. Handschuh, M. Vargas-Vera, E. Motta and F. Ciravegna, Semantic annotation for knowledge management: Requirements and a survey of the state of the art. *Journal of Web Semantics*, 4, pp. 14-28, 2006.
- R. Villa, R. Wilson, and F. Crestani, Ontology mapping by concept similarity. *International Conference on Digital Libraries*, pp. 666-674, 2003.
- T. Berners-Lee, W. Hall, J.A. Hendler, K. O'Hara, N. Shadbolt, N. and D.J. Weitzner, A Framework of Web Science. *Foundations and Trends in Web Science*, 1(1), pp 1-25, 2006.
- B. Pell, POWERSET - Natural Language and the Semantic Web. *The 6th International Semantic Web Conference and the 2nd Asian Semantic Web Conference, 2007*.
- R. Schwitter, A Controlled Natural Language Layer for the Semantic Web. *AI 2005: Advances in Artificial Intelligence*, pp. 425-434, 2005.
- M. Vargas-Vera, E. Motta, J. Domingue, M. Lanzoni, A. Stutt, and F. Ciravegna, MnM: Ontology Driven Semi-Automatic and Automatic Support for Semantic Markup. *Proc. of EKAW 2002*, pp. 379-391, 2002.
- B. Katz, and J. Lin, Annotating the Semantic Web Using Natural Language. *Proceedings of the 2<sup>nd</sup> Workshop on NLP and XML, Taipei, September 200*, pp. 1-8, 2002.
- R. Neches, R.E. Fikes, T. Finin, T.R. Gruber, T. Senator and W>R. Swartout, Enabling Technology for Knowledge Sharing. *AI Magazine* 12(3), pp. 36-56, 1991.
- T.R. Gruber, A Translation Approach to Portable Ontology Specification. *Knowledge Acquisition* 5(2), pp. 199-220, 1993.
- A. Gomez-Perez, M. Fernandez-Lopez and O. Corcho, Ontological Engineering. Berlin: Springer-Verlag, 2004.
- G. Miller, WordNet: A Lexical Database for English. *Communications of the ACM* 38(11): pp. 39-41, 1995.
- D.W. Embley, Toward Semantic Understanding – An Approach Based On Information Extraction Ontologie. *Proceedings of the 15th Australasian Database Conference, 2004, (ADC'04)*, Dunedin, New Zealand, pp. 3-12, 2004.
- D.W. Embley, D.M. Campbell, Y.S. Jiang, S.W. Liddle, D.W. Lonsdale, Y.K. Ng and R.D. Smith, Conceptual-model-based data extraction from multiple-record Web pages. *Data and Knowledge Engineering*, pp. 227-251, 1999.
- T. Brasethvik, and J.A. Gulla, A Conceptual Modelling Approach to Semantic Document Retrieval. *Advanced Information Systems Engineering, 14th International Conference*, pp.167-182, 2002.
- T. Brasethvik and J.A. Gulla Natural language analysis for semantic document modeling. *Data and Knowledge Engineering*, pp. 45-62, 2001.

- H. Alani, S. Kim, D. Millard, M. Weal, W. Hall., P. Lewis, P. and N. Shadbolt, Ontology knowledge extraction from web documents. *IEEE Intelligent Systems*, 18(1), pp. 14-21, 2003.
- H.P. Luhn, The automatic creations of literature abstracts. I.B.M. Journal of Research and Development, 2(2), pp. 159-165, 1958.
- S. Sekine *Proteus Project - Apple Pie Parser (Corpus based Parser)*. <http://nlp.cs.nyu.edu/app> (accessed on 15 September 2006)
- S. Sekine and R.A. Grishman, Corpus-based Probabilistic Grammar with Only Two Non-terminals, *Fourth International Workshop on Parsing Technology*, pp. 216-223, 1995.
- H. Snoussi, L. Magnin and J.Y. Nie, Towards an ontology-based web data extraction. *The AI-2002 Workshop on Business Agents and the Semantic Web (BASeWEB)*, pp. 26-33, 2002.
- C. Batini, M. Lenzerini and S.B. Navathe, A comparative analysis of methodologies for database schema integration, *ACM Computing Surveys*, 18(4), pp. 323-364, 1986.
- S.A. Noah and M. Williams, Intelligent object analyzer for conceptual database design, *Jurnal Teknologi*, 3, pp. 27-44, 2004.
- I.H. Witten, G.W. Paynter, E. Frank, C. Gutwin and C.G. Nevill-Manning, KEA: Practical automatic keyphrase extraction, *Working Paper 00/5*, Department of Computer Science, The University of Waikato, 2000
- M. Song, I.Y. Song and X. Hu, An efficient keyphrase extraction system using data mining and natural language processing techniques. *First International Workshop on Semantic Web Mining and Reasoning*, pp. 58-65, 2004.

# Extracting Conceptual Graphs from Japanese Documents for Software Requirements Modeling

Ryo Hasegawa<sup>1</sup>

Motohiro Kitamura<sup>1</sup>

Haruhiko Kaiya<sup>2</sup>

Motoshi Saeki<sup>1</sup>

<sup>1</sup>Dept. of Computer Science, Tokyo Institute of Technology  
Ookayama 2-12-1, Meguro-ku, Tokyo 152, Japan  
Email: saeki@se.cs.titech.ac.jp

<sup>2</sup>Dept. of Computer Science, Shinshu University  
Wakasato 4-17-1, Nagano 380-8553, Japan  
Email: kaiya@cs.shinshu-u.ac.jp

## Abstract

A requirements analysis step plays a significant role on the development of information systems, and in this step we produce various kinds of abstract models of the systems (called requirements models) according to the adopted development processes, e.g. class diagrams in the case of adopting object-oriented development. However, constructing these models of sufficient quality requires highest intellectual tasks and skills of human requirements analysts. In this paper, we develop a computerized tool to extract from a set of Japanese text documents conceptual information, called *conceptual graph*, which can be used as intermediate representation to generate software requirements models. More concretely, by applying the variation of text-mining techniques that we have developed, we extract significant words from text documents referring to the same problem domain and identify relevant relationships among them. The extracted words can be considered as concepts and they are constituents of a conceptual graph in the domain. This constructed graph can be used for generating requirements models, e.g. object oriented models, feature model, and even as a domain ontology that can be utilized during requirements analysis activities. We have made experimental analyses of our tool. This paper also includes the discussion on how the extracted conceptual graph can act as an object-oriented model, a feature model and a domain ontology, in order to show its wide applicability.

**Keywords:** Conceptual Graph, Requirements Modeling, Text mining, NL processing

## 1 Introduction

Since a requirements analysis step is the first one in information systems development processes, the quality of the artifacts that are produced in this step greatly affects on the quality of a final artifact, i.e. an information system. If we constructed an artifact of lower quality in this step, for example an incomplete and/or inconsistent one, we might re-do our activities after completing the final artifact and as a result we might spend much effort and the development cost might exceed an estimated budget.

In this requirement analysis step, we produce abstract models of the information system according to the adopted development process style. For example, when we use object-oriented (OO) development process, we produce a class diagram as an object-oriented model. If we develop a product belonging to a certain family and adopt

Feature-Oriented Analysis technique, we should produce a feature oriented model. Thus we can produce various kinds of model in a requirements analysis step according to the adopted development process. We call these models, i.e. abstract models of the system that produced in a requirements analysis step, *requirements models*. We should construct a requirements model of high quality as early as possible to reduce development costs and efforts. However, human engineers are required to perform highly intellectual and complicated activities and to have distinguished skills in order to construct a requirements model of high quality. In addition, they should be experts to various modeling techniques that can be adopted. A current status is that a limited number of domain experts are involved in requirements modeling in their domains, spending their large efforts. We need some supporting techniques to assist human engineers in constructing various types of requirements models of higher quality with less effort.

On the other hand, it is a rare case that we construct a requirements model whose domain is quite new and does not appear before. If we had reusable assets helpful for requirements modeling, we could get the model efficiently. However, we have not accumulated sufficient reusable assets of requirements models in a certain domain yet. Rather, we can get many text documents referring to the domain, including the electronic texts lying over Internet. In fact, the experts to modeling frequently use the documents regarding to the topics relevant to the problem domain so as to get important information. Thus, it can be considered as a promising support technique to extract from the documents information necessary for requirements modeling. These documents are written in natural language, and the constituents that a requirements model should have, e.g. concepts and their relationships appear in the documents as words and their co-occurrences in a suitable abstraction level, because of the abstractness of natural language descriptions. The words that commonly appear in the documents of a domain, except for general words such as be-verbs, prepositions, particle, etc., can be considered as the representation of significant concepts in the domain. In addition, the usages of these words such as co-occurrence and modification relationships suggest the relationships between the concepts that the words denote. Thus we focus on the extraction of these words and relationships from the documents.

To generate various kind of requirements model, we extract an intermediate representation from a set of text documents by using the combination of natural-language (NL) processing and text-mining techniques so that it can be (semi-)automatically transformed into various requirements models. Our intermediate representation is called *conceptual graph*, which includes concepts and their relationships extracted from the documents. Furthermore requirements analysts can use this graph to make up for their lack of domain knowledge during their requirements elicitation activities. Figure 1 shows the overview of our

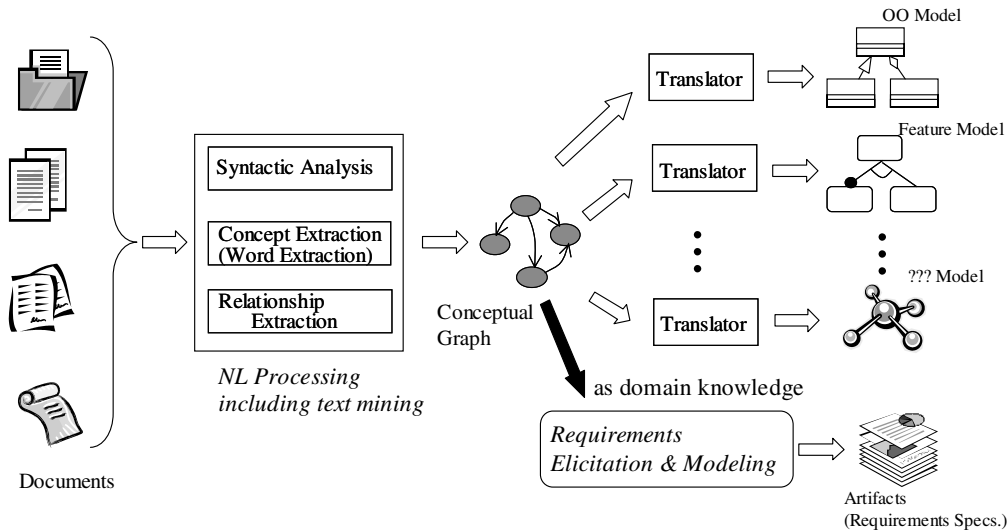


Figure 1: Overview of Our Approach

approach. In this paper, we have developed a computerized tool for extracting conceptual graphs from a set of documents. As will be discussed in the section of Related Work, we can find many studies to extract specific requirements models such as OO models from a *single* document. Unlike these studies, we use a set of documents as inputs so that we can get stable and reusable conceptual information. It is very significant which information we should extract from documents. Since our target is information systems, we adopt the concepts and their relationships that we frequently use in modeling them, e.g. Object, Function, Is-a relationship (generalization) and Has-a (aggregation), etc., and construct from them a meta model of the conceptual graphs.

The rest of this paper is organized as follows. In the next section, we explain the basic idea and show the logical structure of the conceptual graphs, i.e. meta model. We extract from Japanese documents information based on this meta model. Section 3 presents the process for extracting conceptual graphs and the computerized tool using NL processing and the text mining technique that we have proposed. Since our conceptual graphs have more specific conceptual types and relationship ones rather than usual thesauruses, we should develop newly a text mining technique. Section 4 includes experimental results on the effectiveness of our developed tool. In section 5, we discuss how to get software requirements models from the constructed conceptual graph in order to show its wide applicability. In sections 6 and 7, we discuss related work and our current conclusions together with future work, respectively.

## 2 Meta Model of Conceptual Graphs

### 2.1 Requirements to a Meta Model

As mentioned in section 1, we have a variety of notations for requirements models such as Entity Relationship Diagram and UML (Class Diagram etc.), and they have different meta concepts for description. In the case of Class Diagram, it has meta concepts Class, Attribute, Operation, Association, etc. Therefore, we need to clarify the structure of conceptual graphs, i.e. a meta model of conceptual graphs so that we can extract Classes, Attributes, Operations, Associations etc. from the conceptual graph afterward. Our meta model should 1) have extensive meta concepts so that we can derive various requirements models, even reusable assets such as feature model of FODA [1], from a conceptual graph that is its instance, 2) have useful

meta concepts specific to the area of information system, and 3) be based on the information that can be automatically gathered from text documents.

### 2.2 Meta Model of Conceptual Models

In order to satisfy the requirements to the meta model mentioned in section 2.1, we analyzed the existing software requirements modeling methods, referring to UML's meta model [2], Method Engineering meta model [3], Method Common Meta Model [4], etc. and have got the meta model shown in Figure 2. Our meta model consists of concepts and relationships among the concepts, and it has several subclasses of "concept" class and "relationship". In the figure, "object" is a subclass of a concept class and a relationship "apply" can connect two concepts. Concepts and relationships in Figure 2 are adopted so as to easily represent the semantics in information systems. Intuitively speaking, the concepts "object", "function", "environment" and their subclasses are used to represent functional aspects of the systems. On the other hand, the concepts "constraint" and "quality" are used to represent non-functional aspects. The concept "constraint" is useful to represent numerical ranges, e.g., speed, distance, time expiration, weight and so on.

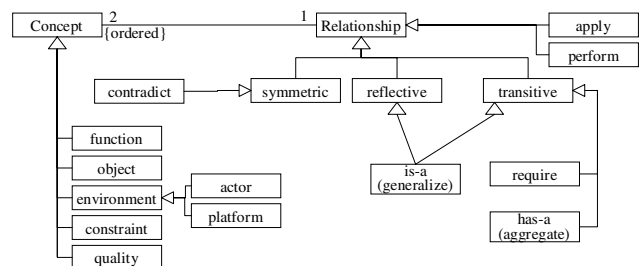


Figure 2: Meta Model of Conceptual Graphs

Figure 3 shows an example of a conceptual graph of the problem domain of "making estimates", an instance of the meta model of Figure 2, which is depicted in the form of Class Diagram, and it is a screenshot of our tool. Note that our tool is for Japanese only and the figures of tool screens have been produced by translating Japanese words into English directly. A concept and its type are depicted as Class and a stereo type respectively in the figure. Readers can find the concepts of type object, "esti-

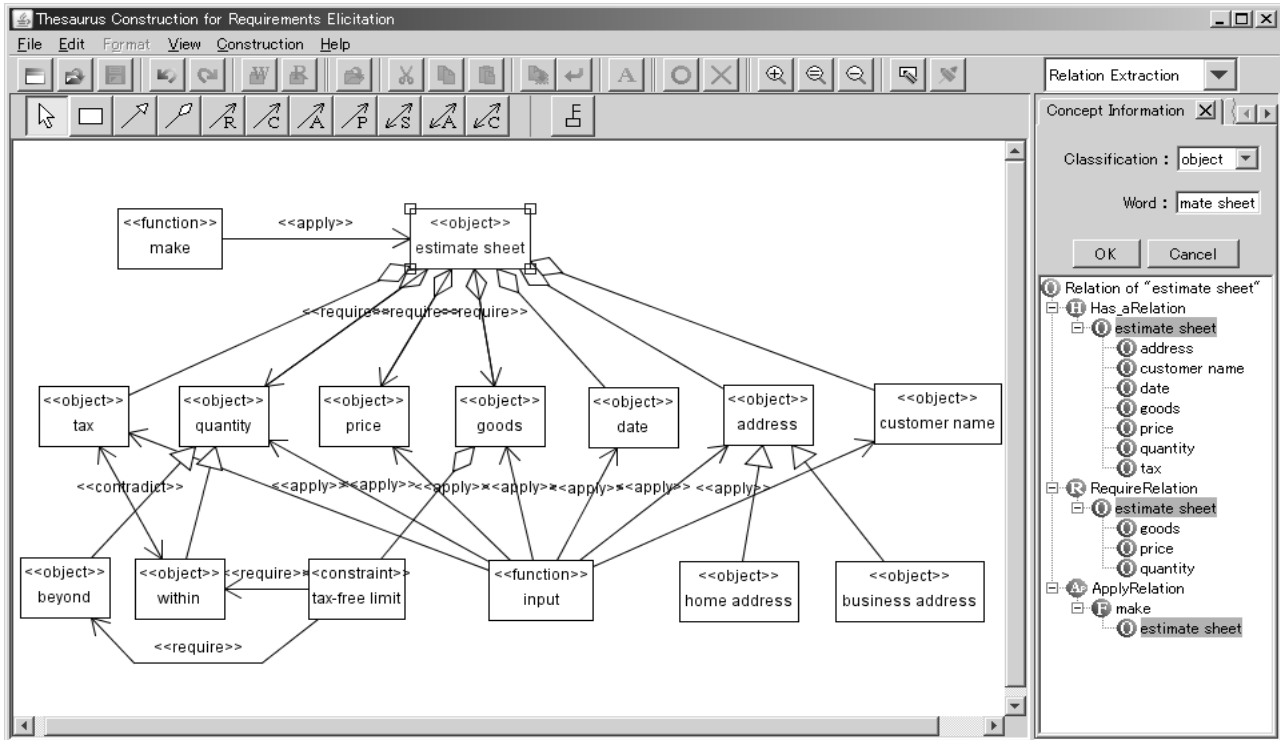


Figure 3: A Tool Screenshot of Relationship Extraction: An Example of a Conceptual Graph

mate sheet”, “goods”, “tax”, etc, and “input” and “make” of type function. There are two relationships between “estimate sheet” and “goods”; one is the relationship of type “require” and another is of “has-a”<sup>1</sup>. Since an estimate sheet should have some columns on goods, their prices and their quantity information, we use the combination of these two types (require and has-a) of relationships between the estimate sheet and them. The concept “input” of type “function” is applied to “goods”, “quantity”, “prices” etc. in order to input these data, and thus we can have the relationships of type “apply” to them.

### 3 A Supporting Tool for Extracting Conceptual Graphs

In this section, we focus on the technique to extract constituents of a conceptual graph from Japanese text documents. The quality of a conceptual graph greatly depends on the quality of used text documents. If we use a document of lower quality, we also get a graph of lower quality. There are no formal techniques to validate the quality of the extracted conceptual graph. We consider that the quality of the conceptual graph can be validated by *social consensus* of domain experts and by its usability to our applications. We can consider that concepts and relationships commonly appearing in many documents on a domain have established social consensus. The larger the number of documents is, the higher the quality of the extracted graph can be, because the concepts appearing in the many documents are widely accepted in this domain. As for the usability to our applications, we will discuss it in section 5.

Basically, nouns and verbs included in the documents correspond to the object concepts and functions of Figure 2 respectively, and adjectives and adverbs modifying objects or functions represent the concepts of quality. Thus the essential parts of our process for extracting a conceptual graph are 1) Word extraction for extracting from doc-

uments the important words that can be considered as useful concepts and 2) Relationship extraction for discovering the relationships among the extracted words, as shown in Figure 4.

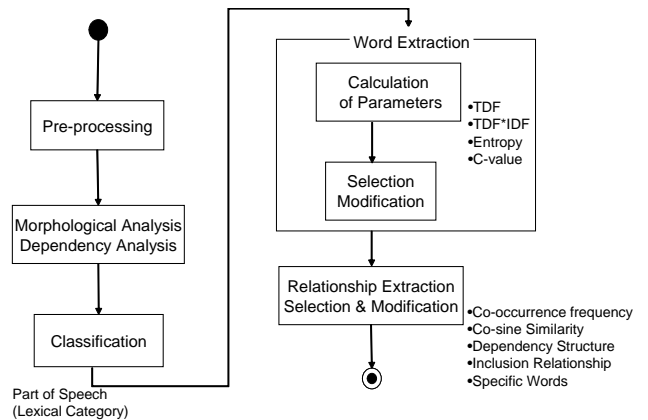


Figure 4: Process for Extracting a Conceptual Graph

#### 3.1 Word Extraction

After morphological analysis and dependency analysis, we identify part-of-speech categories of the meaningful words appearing in the documents such as noun, verb, adjective etc. These steps can be performed automatically using the natural-language processing tool called Cabocha (dependency structure analyzer for Japanese)<sup>2</sup>. By using part-of-speech information of words, we classify the words into the types of the concepts shown in Figure 2 such as object, function and quality. For example, “estimate sheet” is a noun and is classified into an object concept. In the next step, our tool calculates various measure

<sup>1</sup>We use the same notation as UML Class diagram to represent “has-a”, i.e. aggregation relationship.

<sup>2</sup><http://chasen.org/taku/software/cabocha/>

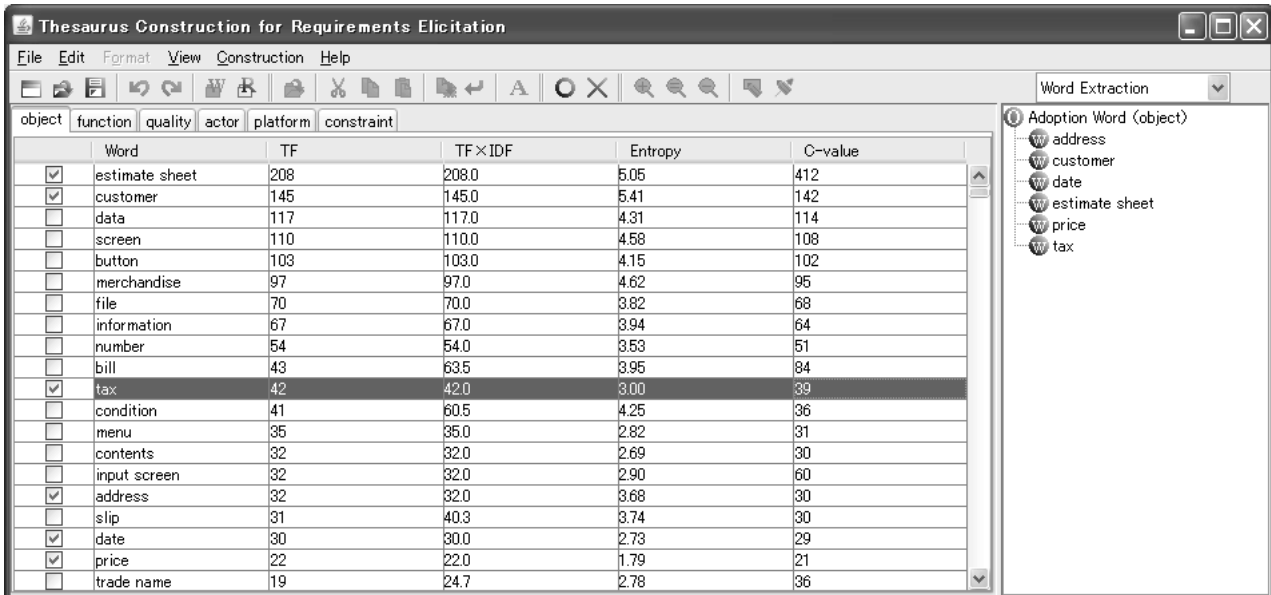


Figure 5: A Tool Screenshot of Word Extraction

parameters of the words so that we can filter out unimportant words from the classified words. The parameters that we use are based on word frequency, i.e. the number of times a word appears in documents, and are shown below.

1. TF (term frequency): the number of times a word appears in the documents.
2. TF × IDF (term frequency × inverse document frequency): the term frequency of a word weighted with its importance degree. The importance degree results from the number of the documents the word appears.
3. Entropy: logarithmic value of the term frequency of a word weighted with its entropy [5]. Intuitively speaking, an entropy value of the word A comes to be lower if A appears uniformly throughout all documents.
4. C-value: the term frequency of a word weighted with its length and its occurrences as a part of multi-words. This value is for the characteristic of Japanese texts that they frequently include many occurrences of multi-words. A multi-word is a combination of several words.

Figure 5 shows an example of the result of word extraction. As shown in the figure, the words are measured and sorted in descending order of the measure values. A user of the tool can select the important words denoting concepts in a conceptual graph of a problem domain, by checking boxes on the sorted list of the measured words. In the example of the figure, the user has manually selected the words “estimate sheet”, “customer”, “tax”, “address”, “date” and “price”.

### 3.2 Relationship Extraction

After selecting the words, the user proceeds to the step of relationship extraction. As shown in Figure 4, the tool calculates the number of times a pair of words included in a sentence in the documents, i.e. co-occurrence frequency (CF) of two words and cosine similarity (CS) of the frequency of co-occurrence vectors, in order to find the semantically relevant word pairs. If the two words co-occur frequently, we can consider the two concepts denoted by them are semantically related to each other.

The calculation method of cosine similarity of co-occurrence vectors is as follows. Suppose that the words

$y_1, \dots, y_n$  frequently co-occur with the word  $x$  in the documents. We can define a co-occurrence vector  $\mathcal{V}_x$  of the word  $x$  as  $(c(x, y_1), \dots, c(x, y_n))$  where  $c(x, y_i)$  is the number of times in which the words  $x$  and  $y_i$  co-occur. Thus we can calculate cosine similarity (CS) of the co-occurrence vectors of the words  $u$  and  $w$  as  $(\mathcal{V}_u \cdot \mathcal{V}_w) / (|\mathcal{V}_u| \cdot |\mathcal{V}_w|)$ . If the cosine similarity is sufficiently higher, we can consider that the words  $u$  and  $w$  are used in a similar way in the documents and that they have a certain semantic relationship.

After calculating CFs and CSs, pairs of words whose CF and CS are higher than certain thresholds are basically selected as candidates of the relationships to be included in a conceptual graph. Based on types of words (e.g. object, function and quality) and dependency structures in the sentences, the tool suggests the types of the concept relationships. Figure 6 shows the detailed process to extract relationships and to identify their types. Suppose that we focus on the words A and B, as shown in the figure. If A and B co-occur in a sentence and they also co-occur with a specific word such as “require”, “contradict”, “such as”, etc., we decide that A and B has a relationship of types “require”, “contradict” or “is-a”, respectively. If A is a precisely right-side substring of B, we consider the relationship as “is-a”. For example, the word “new estimate” (shinki-mitsumori-sho in Japanese) is an “estimate” (mitsumori-sho), because the word mitsumori-sho appears in shinki-mitsumori-sho as its right-side substring. If the CF value of A and B is higher, we check the dependency structure of the sentences where they co-occur and by their types and their syntactic roles such as subject and object etc., we decide the type of their relationship. For example, suppose that the types of A and B are “object” and “function” respectively. In addition, if A is an object in grammatical sense and B is its verb in a sentence, the tool suggests an “apply” relationship between A and B. “Apply” relationship between object A and function B means that the function B is applied to the object A. CS values are used to detect require and has-a relationships.

Figure 3 shows an example of the detected concepts and their relationships in a class diagram-like form. The tool users can modify the detected relationships and edit the diagram to make it more complete and precise as a conceptual graph.



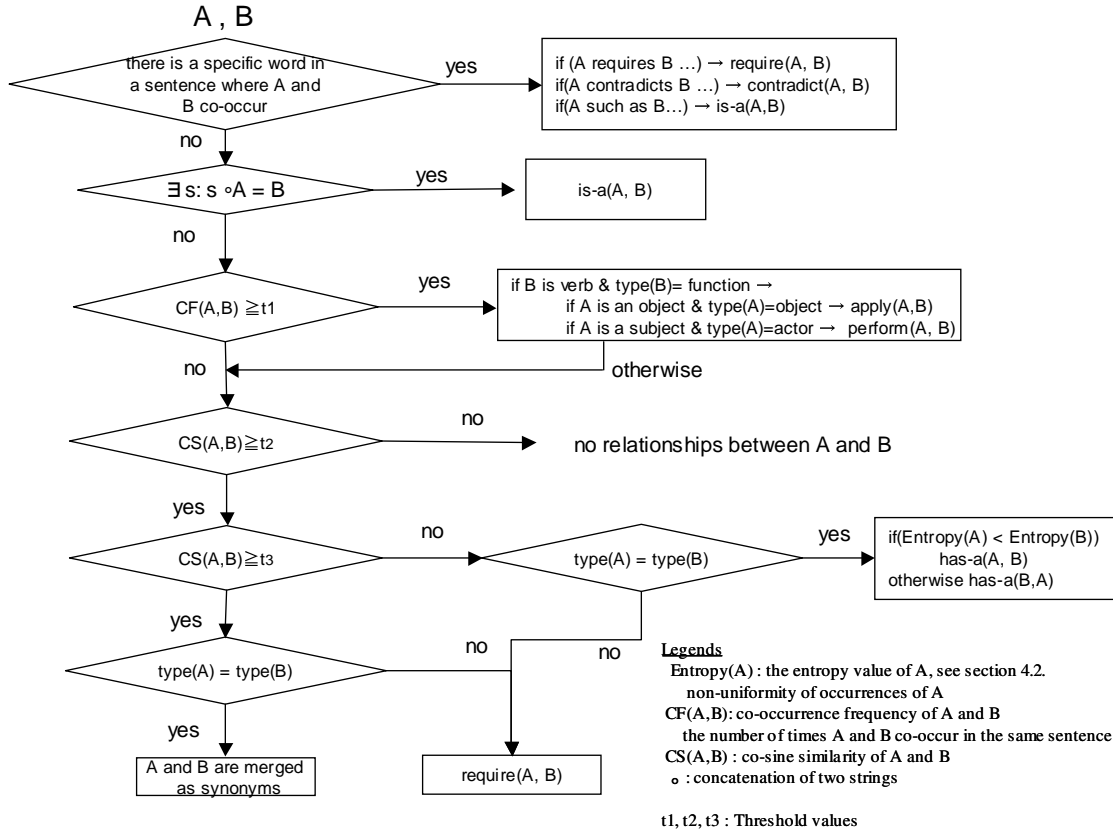


Figure 6: Relationship Extraction Process

## 4 Experimental Results

To assess our tool, we made several experiments, and in this section we discuss these experimental results.

### 4.1 Aims of Our Experiments

The essential aim of our experiments is to show that our tool allows any requirements analyst, who has a skill and domain knowledge in a certain level, to derive efficiently conceptual graphs of high quality. For these experiments, we had several subjects who had experiences in software development of more than 5 years including requirements analysis and software design. They had also actually developed software of the problem domains that we adopted in the experiments. Thus we can consider all of them had skills of requirements modeling and domain knowledge in a certain level. And we set threshold values of  $t_1$ ,  $t_2$  and  $t_3$  of Figure 6 to 3, 0.75 and 0.9 respectively.

We can decompose the above aim into the following items;

1. Any analyst<sup>3</sup> can construct conceptual graphs efficiently. Basically, we observe how long it took our subjects to complete their conceptual graphs.
2. Any analyst can get the same results, i.e. the same conceptual graphs, if they use the same documents as inputs to the tool. We have two subjects having the same skills and knowledge, more concretely having similar experiences, and make them construct conceptual graphs from the same documents by using our tool. After their constructing graphs, we compare their results and check how many parts of their constructed conceptual graphs are the same.

Table 1: Results on Feed Readers and POS systems

Problem Domain	Spent Time	Concepts	Relationships
Feed Reader	180 min.	178	270
POS System	160 min.	226	252

3. Any analyst can construct conceptual graphs of high quality. In fact, it is difficult to measure the quality of a conceptual graph. Thus we pay attention to the following two points to estimate the quality of conceptual graphs;

- (a) how many constituents of her graph the subject should modify so as to get to the graph at the quality level that she could be satisfied.
- (b) whether the conceptual graph that the subject constructed could be transformed to requirements models and be used as domain knowledge in requirements elicitation processes.

The second point is related to the application of conceptual graphs and it is very significant to show that they graph can be used for requirements analysis tasks. This point will be discussed in the section 5.

### 4.2 Spent Time for Constructing Conceptual Graphs

We picked up specific problem domains and investigated how long and how large our subjects constructed conceptual graphs, in order to show that they could do efficiently. We selected the two domain Feed Reader and POS (Point of Sales) systems. Their results are shown in Table 1. For example, the subject of Feed Reader finally constructed the graph having 178 concepts and 270 relationships in 180 minutes. In this experiment, we gave 17 documents

<sup>3</sup>In the context of this section, as mentioned above, “analysts” have sufficient skills, experience and domain knowledge like our subjects.

Table 2: Results on Making Estimates

Spent Time	Concepts	Relationships
260 min.	218	432
180 min.	201	363

for the subject of Feed Reader and 14 documents for the subject of POS system. The lengths of the documents that we used were from 3 to 23 pages of A4 paper size. Although their tasks included manual activities to modify the graphs, we consider that our tool is helpful to construct conceptual graphs of practical size within reasonable labor time. In addition, our subjects pointed out that they could know which parts they had to concentrate on for their understanding because our tool suggested significant words in these domains.

#### 4.3 Similarity of the Constructed Graphs

We had two subjects and each of them developed a conceptual graph using our tool from the 8 documents referring to “making estimates” domain. The result is shown in Table 2. Although our two subjects spent different time (260 and 180 minutes respectively) in constructing their conceptual graphs, the sizes of the graphs were similar. They extracted 218 and 201 concepts respectively as shown in the table, and 147 of them were quite the same. Thus about 70% of the extracted concepts were commonly included in the graphs that different persons constructed, and we consider that any analyst can reasonably construct a conceptual graph at a certain level.

#### 4.4 Quality of Conceptual Graphs: Modification Efforts

In the third experiment, we investigated the quality of the constructed conceptual graphs by measuring how much effort the subjects should modify manually the graphs that the tool derived. We selected a domain of “a record management system in a school (for storing and managing records of students’ scores and credits)”. We used 8 manuals for existing software for record management systems. Our subject, a skilled domain expert created a conceptual graph for “a record management system in a school” by using our tool. As shown in Table 3, he finally got 74 concepts and 202 relationships and these can be considered as the graph of high quality, because the distinguished expert manually modified and completed the graph. The tool automatically extracted 68 (64 + 4) and 64 of 68 were used without any modifications. 4 of 68 extracted concepts were modified and 6 concepts were newly added by the expert. As for the relationships, the tool automatically recognized 76 + 62 relationships and 76 were used without any modifications. From this table, our tool could create totally more than 60% of concepts/relationships in the graph. In almost of 62 modifications of the relationships, the expert manually modified has-a and is-a relationships to require relationship, because our technique of Figure 6 could not distinguish correctly require relationship from has-a and is-a relationships. From this experiment, although the tool could not necessarily extract the conceptual relationships accurately, it could do concepts satisfactorily. Human efforts were necessary to get more complete relationships. However, the time spent in modifying and adding concepts and their relationships was less than 2 hours and thus in a tolerable range.

Note that the goal of this tool is not to automate completely the creation of a precisely correct conceptual graph, but to support human activities and produce a useful graph for our application. In the application of modeling the requirements of an information system, it is more important to include concepts and relationships as many

Table 3: Results on a Record Management System

	Concepts	Relationships
Used without any modifications	64 (86.4%)	76 (36.7%)
Modified	4 (5.4%)	62 (30.6%)
Added	6 (8.1%)	64 (31.6%)
Total number	74 (100%)	202 (100%)

as possible, in order to avoid lacking requirements. Thus our tool tries to show many candidates of concepts and relationships. By using our tool, a requirements analyst selects appropriate ones out of the candidates, and replaces their types into correct ones if their types are inappropriate.

### 5 Applications of a Conceptual Graph

In this section, to show the wide applicability of the conceptual graphs constructed using our approach, we explain how to derive an object oriented model and a feature model from an extracted graph. And we show another application where the conceptual graph can be used as domain knowledge for software requirements elicitation processes. By showing the wider applicability of the constructed graph, we can estimate its quality.

#### 5.1 Transforming a Graph into an Object Oriented Model

One of the most popular modeling techniques in software engineering is object oriented modeling and we use class diagrams to represent them. As shown in Figure 2, our conceptual graphs are based on an object oriented modeling technique. Therefore, an object oriented model can be derived from our conceptual graph straightforward. The outline of derivation rules is as follows. For simplicity, we call each subclass of a concept or a relationship in Figure 2 as XX-concept or YY-relationship. For example, we call “function” subclass of a concept “function-concept”. An object-concept in our conceptual graph corresponds to a class in an object oriented model, and function-concepts related to the object-concept with an apply-relationship become methods of the class. Constraint-concepts related to the object-concept become attributes of the class. Is-a-relationships and has-a-relationships in our conceptual graph simply correspond to inheritance and aggregation relationships in the object oriented model.

Figure 7 shows a class diagram (an object oriented model) derived from a conceptual graph in Figure 3 by using the above rules. Object-concepts such as “estimate sheet”, “goods” and “price” become classes in the class diagram, and has-a-relationships in the conceptual graph become aggregation relationships. An aggregation relationship between “estimate sheet” and “goods” is a typical example in Figure 7. A function-concept “input” in the conceptual graph has apply-relationships to several object-concepts as shown in Figure 3. Therefore, classes corresponding to the object-concepts in the class diagram has a “input” method as shown in Figure 7. For example, a class “goods” has a method “input”.

#### 5.2 Transforming a Graph into a Feature Model

Feature modeling was developed by Kang et al, and reusable assets in a product line development can be represented in the model. A definition of a feature is given in [1] as “a prominent or distinctive user-visible aspect, quality or characteristic of a software system or a system”. A feature model has a hierarchical (normally tree) structure among features, which are inherent concepts of a product family, and it is normally depicted in the tree-like diagram

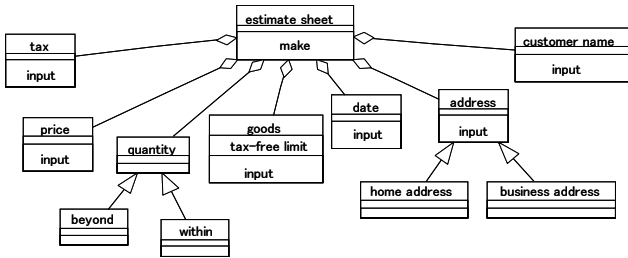


Figure 7: Deriving a Class Diagram

called feature diagram. To specify a model of a product in a product family, features in a feature diagram are chosen in a top down manner, i.e., sub-features are chosen after their super-feature was chosen. When a sub-feature has a mandatory relationship to its super-feature, this sub-feature should be chosen, i.e. the product should have this sub-feature. There are other kinds of relationships among features such as optional, alternative, exclusive and so on.

A conceptual graph can be derived from documents about a product family, and concepts in the graph correspond to features in a feature model. Has-a-, is-a-, apply- and perform-relationships in the graph correspond to relationships between super- and sub-feature relationships. In the case of is-a-relationship, the derived relationship between a super- and a sub-feature should be alternative relationships. In addition, since the properties of a super concept are inherited to its sub concepts in our conceptual graph, we consider that the concepts related to the super-concept would be also related to all of its sub concepts.

In Figure 8, we show a feature diagram derived from the conceptual graph in Figure 3. All concepts in the graph are transformed into features at first. Since the features such as “make”, “tax” and “goods” have “has-a”, “is-a” or “apply” relationships with a feature “estimate sheet”, these features become sub-features of the feature “estimate sheet”. A sub-feature “goods” is a mandatory feature of its super-feature “estimate sheet” because “goods” has a “require” relationship to its super-feature “estimate sheet” as well as a “has-a” relationship. Suppose that an apply-relationship between a function-concept and an object-concept is included in our conceptual model. The function-concept corresponds to a sub-feature of a feature corresponding to the object-concept. In addition, this sub-feature becomes a mandatory feature if the apply-relationship is only one between the function and the object, because it is the only one function that can manipulate the object. A sub-feature “make” in Figure 8 is a typical example of this kind of mandatory features and “make” is the only one that can manipulate “estimate sheet” according to the conceptual graph shown in Figure 3. As shown in the figure, object-concepts “home address” and “business address” are the sub-classes of an object-concept “address” because these concepts have is-a relationships to “address”. These two concepts become alternative sub-features of a feature “address” as shown in Figure 8. In addition, these two features has a sub-feature “input”, because a function-concept “input” has an apply-relationship to the object-concept “address” and two concepts “home address” and “business address” are the sub-classes of the object-concept. By applying these kinds of transformations, the feature diagram of Figure 8 can be derived from Figure 3.

In a feature diagram, several kinds of relationships among features, e.g., a dependency relationship among features and an exclusive relationship between features, are allowed in addition to the tree-like hierarchy of features. When there is only a require-relationship between two concepts in our conceptual graph, we have a dependency relationship between the two features corresponding to these concepts. When there is a contradict-

relationship among concepts in a conceptual graph, there is an exclusive relationship between the features corresponding to the concepts. An example of exclusive relationships appears between features “tax” and “within” in Figure 8. An example of dependency relationships appears between features “tax-free limit” and “within”.

In deriving object oriented models and feature models, their derivation rules can be formally defined, and these derivations can be automatically achieved. However, the quality of derived models cannot be guaranteed without the inspection of human experts. Thus such derivation rules play a role of guidelines only. This kind of derivation should be achieved interactively, and the finally derived models should be improved by manual.

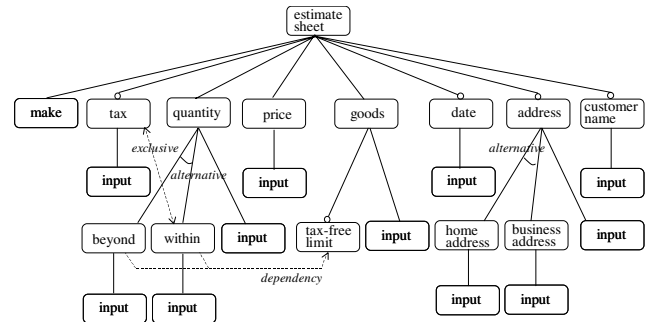


Figure 8: Deriving a Feature Diagram

### 5.3 Using as a Domain Ontology

Knowledge on a problem domain where an information system is operated (simply, domain knowledge) plays an important role on eliciting system requirements of high quality. For example, to develop e-commerce systems, the knowledge on marketing business processes, supply chain management, commercial laws, etc. is required as well as internet technology. Although requirements analysts have much knowledge of software technology, they may have less domain knowledge. As a result, lack of domain knowledge allows the analysts to produce requirements specification of low quality, e.g. an incomplete requirements specification where mandatory requirements are lacking. Although interviews with domain experts are one of the solutions to avoid this problematic situation, communication gaps between the analysts and the domain experts resulted from their knowledge gaps [6]. Thus, the techniques to provide domain knowledge for the analysts during their requirements elicitation and computerized tools based on these techniques to support the analysts are necessary.

We have proposed how to use domain ontologies for requirements elicitation [7] where domain ontologies are used to make up domain knowledge to requirements analysts during requirement elicitation processes. In this framework, how to create domain ontologies of high quality efficiently is a crucial issue. Our tool for extracting conceptual graphs can be used to create domain ontologies for supporting requirements elicitation processes.

In this section, we present the basic idea how to use our conceptual graph as domain knowledge to detect lacking requirements and inconsistent requirements. Below, let's consider how a requirements analyst uses a conceptual graph of a certain domain for completing requirements elicitation. Suppose that a requirements document initially submitted by a customer is itemized as a list. At first, an analyst should map a requirement item (statement) in a requirement document into concepts of the conceptual graph as shown in Figure 9. For example, the item “bbb” is mapped into the concepts A and B and an aggregation relationship between them. The requirements document

may be improved incrementally through the interactions between a requirements analyst and stakeholders. In this process, logical inference on the graph suggests to the analyst what part she should incrementally describe. In the figure, although the document S includes the concept A at the item bbb, it does not have the concept C, which has a require-relationship to A in the conceptual graph G. The inference resulted from “C has a require-relationship to A (i.e. C is required by A)” and “A is included” suggests to the analyst that a statement having C should be added to the document S. The details of this technique are out of scope of this paper, and the readers who have a great interest to it can see [7].

To assess this technique, we used the conceptual graph of Feed Reader in section 4.1 and made comparative experiments of requirements elicitation of a specific feed reader system. As a result, subjects with less domain knowledge could get the same results as a domain expert, more concretely they could elicit requirement of the same quality as the domain expert did. The details of the experiments and their results are shown in [8]. This result means that our conceptual graph is applicable as domain knowledge for requirements elicitation processes.

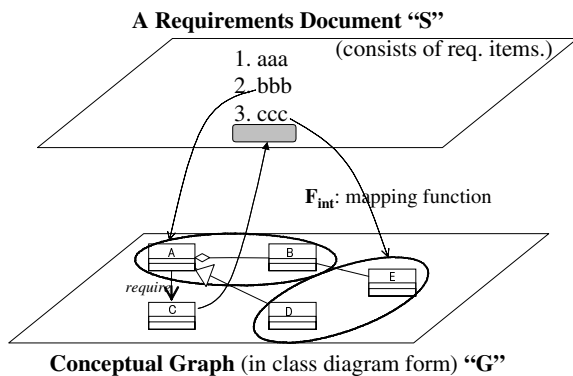


Figure 9: Mapping from Requirements to a Conceptual Graph

## 6 Related Work

In the area of requirements analysis and software specification, some studies to extract requirements models by applying NLP techniques to natural-language documents exist [9]. In particular, many of them derive OO models, e.g. class diagrams [10, 11, 12, 13] for software systems. Their techniques are basically to focus on nouns and verbs that are indicators of classes and of operations or relationships respectively, and their success greatly depends on the quality of an input document. For example, if mandatory descriptions are lacking from the document, the corresponding part of the model cannot be extracted. Since our approach uses multiple documents as inputs, our approach can mitigate these shortcomings. Furthermore they did not consider the extracted models as reusable assets like feature models. And, since we have adopted a variety of types of concepts and their relationships in our meta model of conceptual graphs so as to have wide applications for requirements analysis, we have developed a newly devised text-mining technique fit to our meta model in order to achieve the construction of the graphs from documents. In the area of database systems, a lot of work has also been done to derive a family of Entity Relationship (ER) models from natural-language documents and their major aims are designing a data schema [14, 15, 16, 17, 18]. They focused on the extraction of entities, attributes, relationships and inheritance ones, but did not consider the other constructs such as require re-

lationships, which are necessary for requirements elicitation. Furthermore, an ER model can be derived from our conceptual graph in the same way as section 5.1, and in this sense, our resulting conceptual graph includes rich information for requirements modeling. CM builder, developed by Harmain et. al. [19], uses the domain knowledge, that has been made ready beforehand, to analyze semantically documents. More precisely, in their approach, the domain knowledge is extended to a more specific model by means of adding the extracted classes to it. Although our conceptual graph plays the same role on their domain knowledge of CM builder's technique, they did not discuss the technique how to construct the domain knowledge, i.e. conceptual graphs.

In research community of Ontology, many computerized tools for supporting ontology creation using text-mining techniques have been developed. Text2Onto of KAON [20, 21] is a computerized tool having a text-mining functions based on  $TF \times IDF$  measure so that words frequently appearing can be extracted from text documents. In fact, our tool uses the same quantification techniques for word extraction. In [22], the author applied to software documents of a certain domain the technique similar to Text2Onto to extract the terminology that software developers, domain experts and other stakeholders could commonly use during software development processes. OntoLearn [23] adopted a kind of pattern matching technique to disambiguate words in the semantic analysis for word extraction. DODDLE [24] is also a tool to mining English texts for concept extraction based on term frequency, and it uses WordNet [25] and EDR dictionary [26] as a general-purpose ontologies. Although these tools developed by ontology communities have some functions to make our tool more elaborated, all of them cannot classify the extracted concepts and relationships into the types specific to requirements models as shown in Figure 2, e.g. "Class", "Function", etc. for concepts and "apply", "require", "perform", etc. for conceptual relationships. They are just for extracting concepts with no types and too general relationships such as "is-a", "has-a" and "synonym" etc. as general-purpose ontology or thesauruses, not necessarily suitable for requirements analysis. Their aim is different from ours and they are not immediate supports to requirements modeling. Our conceptual graphs have a variety of types of concepts and of relationships in order to apply to requirements modeling and elicitation, and these existing techniques could not classify the extracted concepts and relationships into these types. To support seamlessly requirements modeling, these techniques should extract not only concepts and their relationships but also their types that lead to the elements of requirements models.

As for the quality of input documents, [27] suggested several guidelines of writing natural-language sentences that could be used for extracting requirements models. Although they are for German, some of them could be useful to improve the quality of input documents for our tool.

## 7 Conclusion

In this paper, in order to support requirements modeling, we presented a computerized tool for extracting conceptual information from Japanese documents, and made several experiments to show the usefulness of our tool. Although our experiments mentioned in section 5 were too small in the sense of practical setting to argue the generality of the experimental findings, we could find the possibility of supporting the construction of useful conceptual graphs. According to the results of interviews to our subjects, the user interface of our tool should be improved.

None of conceptual graphs that our tool suggested included contradiction relationships, and our subjects added them by manual. The reason was that the documents we used did not contain any specific words denoting contra-

diction. We should explore more elaborated mining techniques together with good samples of documents.

Although our current approach is based on the frequency of words in documents, frequent words are not always important in general. Comparing different documents [28, 29] is one of the ways to complement this frequency based approach. Another way to create a conceptual graph of higher quality is the integration of many existing ontologies, including WordNet and EDR dictionary.

In sections 5.1 and 5.2, we illustrated how to derive two types of requirements models from our conceptual graphs. Formalization of these derivation rules using a graph rewriting system [30] and its automation are also a future work.

In section 5.3, we used our conceptual graph as domain knowledge. There are several excellent techniques and Meta CASE tools to generate domain specific modeling languages such as MetaEdit+ [31], Metaview [32] and GME [33]. Our conceptual graph can be an input to these Meta CASE tools to produce domain specific modeling environments. This is one of the most interesting applications of our technique.

## References

- [1] K. Kang, S. Cohen, J. Hess, W. Novak, and A. Peterson. Feature Oriented Domain Analysis (FODA): Feasibility Study. Technical Report CMU/SEI-90-TR-21, 1990.
- [2] OMG. Unified Modeling Language Specification, Version 1.4. <http://www.omg.org/cgi-bin/doc?formal/01-09-67>.
- [3] F. Harmsen. *Situational Method Engineering*. Moret Ernst & Young Management Consultants, 1997.
- [4] M. Saeki. A Meta-Model for Method Integration. *Information and Software Technology*, 39:925 – 932, 1998.
- [5] T. Tokunaga. *Information Retrieval and Natural Language Processing (in Japanese)*. University of Tokyo Press, 1999.
- [6] Haruhiko Kaiya, Daisuke Shinbara, Jinichi Kawano, and Motoshi Saeki. Improving the detection of requirements discordances among stakeholders. *Requirements Engineering*, 10(4):289 – 303, Dec. 2005.
- [7] H. Kaiya and M. Saeki. Using domain ontology as domain knowledge for requirements elicitation. In *Proc. of 14th IEEE International Requirements Engineering Conference (RE'06)*, pages 189–198, 2006.
- [8] M. Kitamura, R. Hasegawa, H. Kaiya, and M. Saeki. An Integrated Tool for Supporting Ontology Driven Requirements Elicitation. In *Proc. of 2nd International Conference on Software and Data Technologies (ICSOFT 2007)*, pages 73–80, 2007.
- [9] L. Goldin and D. Berry. AbstFinder, A Prototype Natural Language Text Abstraction Finder for Use in Requirements Elicitation. *Automated Software Engineering Journal*, 4(4):375 – 412, 1997.
- [10] R. Abbott. Program Design by Informal English Descriptions. *Commun. ACM*, 26(11):882–894, 1983.
- [11] M. Saeki, H. Horai, and H. Enomoto. Software Development Process from Natural Language Specification. In *Proc. of 11th International Conference on Software Engineering*, pages 64–73, 1989.
- [12] S. Overmyer, B. Lavoie, and O. Rambow. Conceptual Modeling through Linguistic Analysis Using LIDA. In *Proc. of 23rd International Conference on Software Engineering (ICSE'01)*, pages 401–410, 2001.
- [13] A. Montes, H. Pacheco, H. Estrada, and O. Pastor. Conceptual Model Generation from Requirements Model: A Natural Language Processing Approach. In *Lecture Notes in Computer Science (NLDB 2008)*, volume 5039, pages 325–326, 2008.
- [14] R. Hausser. Database Semantics for Natural Language. *Artificial Intelligence*, 130(1), 2001.
- [15] A. Min Tjoa and L. Berger. Transformation of Requirement Specifications Expressed in Natural Language into an EER Model, 1994.
- [16] P. Chen. English Sentence Structure and Entity-Relationship Diagrams. *Information Science*, 29(2-3):127–149, 1983.
- [17] S. Hartmann and S. Link. English Sentence Structures and EER Modeling. In *Proc. of 4th Asia-Pacific Conference on Conceptual Modelling (APCCM2007)*, pages 27–35, 2007.
- [18] E. Buchholz, H. Cyriaks, A. Dusterhoft, H. Mehlan, and B. Thalheim. Acquiring Complex Information from Natural Language for EER Database Design. In *1st International Workshop on Applications of Natural Language to Data Bases (NLDB'95)*, 1995.
- [19] H. Harmain and R. Gaizauskas. CM-Builder: An Automated NL-based CASE Tool. In *Proc. of 15th IEEE International Conference on Automated Software Engineering (ASE'00)*, pages 45–53, 2000.
- [20] P. Cimiano and J. Volker. Text2onto : A framework for ontology learning and data-driven change discovery. In *Lecture Notes in Computer Science*, volume 3513, pages 227–238, 2005.
- [21] KAON Tool Suite. <http://kaon.semanticweb.org/>.
- [22] L. Kof. Natural Language Processing for Requirements Engineering: Applicability to Large Requirements Documents. In *Proc. of the Workshops, 19th International Conference on Automated Software Engineering*, 2004.
- [23] R. Navigli, P. Velardi, and A. Gangemi. Ontology learning and its application to automated terminology translation. *IEEE Intelligent Systems*, 18(1):22–31, 2003.
- [24] T. Morita, N. Fukuta, N. Izumi, and T. Yamaguchi. DODDLE-OWL: A Domain Ontology Construction Tool with OWL. In *Lecture Notes on Computer Science (ASWC2006)*, volume 4185, pages 537–551, 2006.
- [25] WordNet: A Lexical Database for the English Language. <http://wordnet.princeton.edu/>.
- [26] Japan Electronic Dictionary Research Institute. EDR Home Page. <http://www.jsa.co.jp/EDR/index.html?>
- [27] G. Fliedl, C. Kop, W. Mayerthaler, H. Mayr, and C. Winkler. Guidelines for NL-Based Requirements Specifications in NIBA. In *Lecture Notes in Computer Science (NLDB 2000)*, volume 1959, pages 251–264, 2000.
- [28] Renaud Lecceuche. Finding Comparatively Important Concepts between Texts. In *The Fifteenth IEEE International Conference on Automated Software Engineering (ASE'00)*, pages 55–60, Grenoble, France, Sep. 2000.

- [29] Akira Osada, Daigo Ozawa, Haruhiko Kaiya, and Kenji Kaijiri. Modeling Software Characteristics and Their Correlations in A Specific Domain by Comparing Existing Similar Systems. In Kai-Yuan Cai, Atsushi Ohnishi, and M. F. Lau, editors, *QSIC 2005, Proceedings of The 5th International Conference on Quality Software*, pages 215–222, Melbourne, Australia, Sep. 2005. IEEE Computer Society.
- [30] G. Taentzer, O. Runge, B. Melamed, M. Rudolf, T. Schultzke, and S. Gruner. AGG : The Attributed Graph Grammar System. <http://tfs.cs.tu-berlin.de/agg/>, 2001.
- [31] S. Kelly, K. Lyytinen, and M. Rossi. MetaEdit+ : A Fully Configurable Multi-User and Multi-Tool CASE and CAME Environment. In *Lecture Notes in Computer Science (CAiSE'96)*, volume 1080, pages 1–21, 1996.
- [32] P. Sorenson, J. Tremblay, and A. McAllister. The Metaview System for Many Specification Environments. *IEEE Software*, 2(5):30–38, 1988.
- [33] A. Ledeczi, M. Maroti, A. Bakay, G. Karsai, J. Garrett, C. Thomason, G. Nordstrom, J. Sprinkle, and P. Volgyesi. The Generic Modeling Environment. In *Proc. of WISP'2001*, 2001.

# Modelling Web-Oriented Architectures

Gunnar Thies

Gottfried Vossen

European Research Center for Information Systems (ERCIS)

University of Muenster

Leonardo-Campus 3, 48149 Muenster, Germany

Email: {guth|vossen}@wi.uni-muenster.de

## Abstract

Service-oriented architectures (SOAs) provide the basis of distributed application frameworks where software components are provided as modular and reusable services. Until today there is no generally accepted method for conceptual modelling of a SOA. Rather, there exist several procedural methods which are used in practice. On the other hand, recent developments in the context of what is commonly termed “Web 2.0” show how easy it can be to link or compose (“mesh”) IT components dynamically, so that original SOA goals like flexibility, reusability, or reduction of complexity can indeed be achieved by relatively simple means. An interesting concept in this context is the *Web-oriented architecture* (WOA), which represents a specialization of SOAs obtained by using simple Web 2.0 technologies and standards (e.g., HTTP, SSL, XML). This paper introduces a methodology for designing WOAs, where the big picture follows existing SOA models. In particular, this WOA methodology comprises conceptual as well as realization issues and breaks WOA design down into three distinct phases.

**Keywords:** Web-oriented architectures, conceptual modelling, design methodology

## 1 Introduction

Service-oriented architectures (SOAs) provide the basis of distributed application frameworks (W3C 2004b) where software components are provided as modular and reusable services. The benefits of a SOA are seen in the flexibility of business processes which consist of loosely coupled services, and the resulting potential cost decrease, complexity reduction, reusability potential, and high flexibility. Conceptual modelling is an important factor here, as it not only refers to data modelling, but also needs to take process design into consideration. Indeed, having to deal with all kinds of (legacy) systems and databases makes the development of a complex and business-ready SOA a major challenge. Until today there is no generally accepted method for the conceptual modelling of a SOA or for converting other concepts into a SOA. Rather, there exist several procedural methods, including those from IBM (Ganci 2006) and SAP (Woods & Mattern 2006), which are used in practice. Moreover, the various standards for “easing” the creation of a SOA (e.g., Web services, UDDI, SOAP)

typically make the realization of a SOA more complicated (Vossen 2006). Indeed, the large number of standards reflects an “over-standardization” which makes SOAs difficult and complex to implement. Numerous individual aspects of Web Services are defined by over 70 distinct specifications, yet some of them (like UDDI) are barely used (Hagemann et al. 2007).

On the other hand, recent developments in the context of what is commonly termed “Web 2.0” show how easy it can be to link or compose (“mesh”) IT components dynamically, so that original SOA goals like flexibility, reusability, or reduction of complexity can indeed be reached by relatively simple means. Examples include mashups based on Google Maps (like [www.housingmaps.com](http://www.housingmaps.com)) or applications like Yahoo!Pipes ([pipes.yahoo.com](http://pipes.yahoo.com)); these are based on Web Application Programming Interfaces (Web APIs), which allow using the functionality of a Web application by a simple (most commonly REST-based) service layer. An interesting and emerging concept in this context is the *Web-oriented architecture* (WOA), which represents a specialization of a SOA obtained by emphasizing the use of simple Web 2.0 technologies and standards. Its important aspect is the fact that no additional standards have been defined, but existing ones such as HTTP, SSL, or XML are employed. Since, as mentioned, there are no generally accepted modelling techniques available for a WOA, this paper introduces a methodology for conceptual WOA design which builds upon existing SOA models. In this context the *Business Process Modelling Notation* (BPMN) is used for the definition of relevant processes in every phase. To this end, the BPMN specification<sup>1</sup> is extended by several additional artifacts.

To motivate our approach, a case study originally presented in previous work (Thies & Vossen 2008) is briefly reviewed here; we will later use it for an illustration of our methodology. Our case considers an enterprise that has so far only run stationary shops and now plans to extend its sales operation beyond regional borders. To this end, it wants to move its operation to the Web, but in such a way that available legacy systems can be incorporated into the new IT landscape. Moreover, core business processes need to be captured and implemented, where some parts can be processed automatically, others only semi-automatically.

The core business of the enterprise is the imprinting of textiles and other merchandise with user-defined templates. Customers are offered a wide variety of imprintable articles, which can be decorated with writings or graphics. Orders handled can range from small (e.g., for individuals or clubs) to large quantities (e.g., for companies). Up to now the en-

Copyright ©2009, Australian Computer Society, Inc. This paper appeared at the Sixth Asia-Pacific Conference on Conceptual Modelling (APCCM 2009), Wellington, New Zealand, January 2009. Conferences in Research and Practice in Information Technology, Vol. 96. Markus Kirchberg and Sebastian Link, Eds. Reproduction for academic, not-for profit purposes permitted provided this text is included.

<sup>1</sup>Since no standard metamodel is defined in the specification of BPMN, a coarse overview of the elements is shown later.



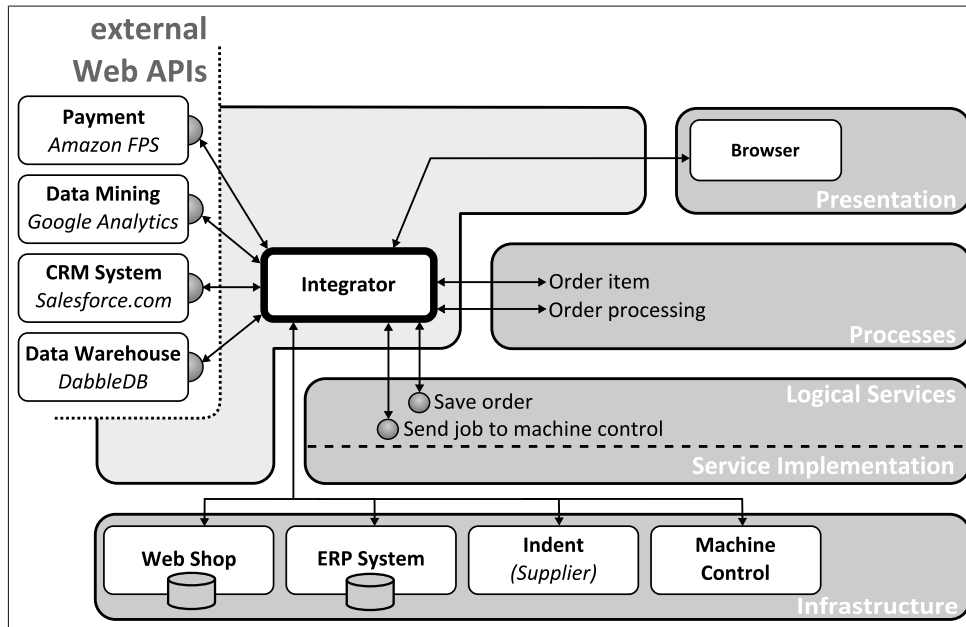


Figure 1: Case study using WOA solution.

terprise runs an ERP system for warehouse and product management. Lithographs are developed together with the customer within a vector graphics application. Moreover, the enterprise runs several machines for printing as well as for flock coating, which are controlled by a central server. The server software is able to accept print commands via a Web service call over the company's intranet. Since machine capacity is not fully used, but the company wants to expand, the business model is to be extended to the Internet and in particular to the Web; consequently, it is to be supported by a new IT system.

The target system is expected to process customer product designs and orders entirely over the Web, so that the customer base can be enlarged considerably. After a transition period, it is even planned to abandon the stationary business completely. Thus, the goal is to implement and run a Web shop alongside the existing ERP system. A possible WOA solution is shown in Figure 1.

As will be seen, the enterprise intends to use as much service offerings from the Web as possible. While this may amount to the design and implementation of a SOA, it has been decided to take a different route: A SOA would typically require a design at several layers of abstraction (Vossen 2006), also indicated in Figure 1, where the infrastructure is lowest, individual services are next, which are topped by service compositions; ultimately business processes as seen by end users are built from these compositions. As Figure 1 shows, a WOA will typically break up this strict division of layers, as individual services obtained over the Web may be employed either as individual service, as a replacement or provisioning of a complex service, or as part of a process. These occurrences may need the help of an integrator component, yet they show that a different design methodology is needed. While SOA design can either follow a bottom-up approach, a top-down approach, or a combination of both ("meet in the middle"), a WOA design can no longer simply follow either of them. A methodology that can be used in WOA design is the subject of this paper.

It should be noted that the fact that a WOA typically breaks up the layer division of a SOA has the

consequence that a certain amount of programming will be needed during the development of a WOA. In other words, the model of a WOA that reflects its architecture and composition and that results from a development process will not be entirely conceptual; instead, it will actually be a hybrid model that comprises both conceptual as well as physical issues. We consider this a kind of "price to pay" for the fact that a WOA is much easier to develop and deploy than a SOA.

The remainder of this paper is organized as follows: In Section 2 we give an overview of related work in the field of conceptual SOA modelling. In Section 3 we then present our methodology which extends and derives from already known methods, in order to meet the requirements for an appropriate WOA methodology. This motivates Section 4, where parts of the case study will be realized using the novel WOA methodology. Finally, in Section 5 we discuss open challenges and future work.

## 2 Related Work

Service-oriented architectures (SOAs) have been discussed intensively in science and industry since their first appearance in the mid-90s. However, even today there is no generally accepted method for modelling a SOA. The various approaches that have been described in the literature aim at methodologies for planning and implementing a SOA, but none has become a "standard" yet. We review available methods next.

First of all there is the *Web Service Architecture* (WSA) of W3C (2004b), where a system of Web Services and their relations to each other are conceptually described. Here, specific aspects are considered: the Service Model, the Message Oriented Model, the Resource Oriented Model and the Policy Model. This approach mainly aims at the usage of Web Services and therefore is less general than our approach. Another approach inspired by WSA comes from the EU project *Service Centric Systems Engineering* (SeCSE) where Colombo et al. (2005) are focusing more on some clarification in addition to the WSA model e.g., relationships between the concepts



of service description, semantics, and service interface. This approach is also focused on Web Services and not as generic as our approach.

More into a semantic approach is OWL-S as outlined in W3C (2004a), which describes a service and its metadata in detail. Here the focus lies more on the semantic description of a single Service within an ontology than on a whole system. Also used for the semantic description of a service is the *Web Service Modeling Ontology* which currently exists as a final draft only, see WSMO Working Group (2006). Since we are focusing on the whole architecture, these approaches are only relevant as far as functional descriptions are concerned.

Another important approach is the *Service Oriented Modelling and Architecture* proposed by IBM (cf. Arsanjani & Allam (2006)) where modelling of a SOA is divided into three phases. The first phase deals with *Component Business Modelling* (CBM) for describing all business processes within an enterprise. The second phase is used to connect processes with services, and the third phase deals with the implementation of the found services within a SOA. This conceptual view is, among others, the basis of the approach described in this paper. Furthermore, we provide a brief description of a conceptual process model which deals with the creation of a WOA from scratch.

More specific is the approach by Mos et al. (2008), which is a method for designing a SOA with the goal of using an *Enterprise Service Bus* (ESB) and therefore Java Business Integration. This method is restricted to a Java environment, but the three phases of the (top-down) method are a good starting point and close to the IBM approach: First discover and describe the processes via *Business Process Execution Language* (BPEL) or BPMN by a *Business Process Designer*, then create a Service Component Architecture descriptions by a *Software Architect* and finally implement and deploy the various services within the ESB by a *Technology Team*.

Another approach by Quartel et al. (2007) named *COncceptual Service MOdelling* (COSMO) focuses on basic concepts to represent essential, elementary, and generic service properties. Here processes are described within three levels of abstraction: as a single interaction, as choreography, and finally as orchestration. For the description of services and their interactions the *Interaction System Design Language* and other languages like OWL, SPARQL, and Java are used. This in a sense contradicts the general idea behind a WOA, which is to use simple and commonly known Web techniques. On the other hand, the division of the provider on the one side and the consumer on the other side and their connection pattern within a process is close to the differentiation of services by a “pool” in our approach.

A more programmatic approach uses the Unified Modeling Language (UML) for modelling a SOA (López-Sanza et al. 2008). It defines a UML profile for the design of PIM-level SOA-based architecture models. In contrast to our approach, processes are less important; not only service providers and services are defined within the SOA metamodel, but also so-called *Active Components*, which handle service request and responses and interact with the frontend. These components are implicitly visible in our approach as well, namely as tasks within the pool named “integrator” of a process model.

### 3 The Approach

In this section we describe our approach to conceptual WOA design. We start out by collecting various requirements and setting several goal. After that

we present the methodology, which is comprised of three phases: process modelling, refinement, and implementation. Each phase will be described in necessary detail.

#### 3.1 Requirements and Goals

The existing methodologies for designing a SOA mostly focus on Web Services as the core connection element. In contrast, a WOA uses Web services as well as RESTful services, i.e., services that can be invoked via REST. As a consequence, a WOA approach is not bound to a specific technology. While Web services can be described by a WSDL document (cf. W3C (2007)) for RESTful services (cf. Fielding (2000)) there is no explicit standard for their description. However, a reasonable solution for the time being is the *Web Application Description Language* (WADL) which is much simpler than WSDL, yet allows the description of important features of a RESTful service such as resource, input, output, and even a textual description, see Hadley (2006). Most services today come with textual descriptions only (as can be seen, for example, at [www.programmableweb.com](http://www.programmableweb.com)), so no matching algorithm can be specified. Therefore, most services have to be “discovered” by the software architects themselves. This can be done in different ways, but our approach currently assumes that the search for suitable services is done by hand in the second phase. As long as no accepted semantic service description is available and used for a large portion of the services offered on the Web, according to experience no reasonable registry mechanism will work.

One of the important points of our approach is the usage of well-known and well-understood formal languages in every phase, including BPMN for process descriptions and WSDL as well as WADL for the functional description of individual services.<sup>2</sup> The advantage of BPMN is that it is easy to read and still allows for a powerful visual design of processes.

The following simplified steps are identified to model, construct, and implement a WOA:

- Development a set of inter-connected business process models;
- refinement of these process models with functional details of each service task;
- provisioning of the process models with explicit data flow (especially a specification of which data is used as input for a service request);
- implementation of the defined process models (in a development environment or workflow engine).

Similar to approaches that have been described in the literature (e.g., the SOMA method of IBM Arsanjani & Allam (2006) or Mos et al. (2008)) the basis of our approach is built upon three different roles and has three main phases (see Figure 2). These roles are the *Process Designer*, the *Software Architect* and the *Technology Team*. In a small enterprise these roles will often be overlapping and assigned to the same person. Otherwise, communication is obviously needed between those with one of these roles assigned.

#### 3.2 The Methodology

We go through the three phases of our design methodology next.

<sup>2</sup>Plans for the near future contain the implementation of a *WSDL2WADL* tool which is able to convert service descriptions between these two description languages without loss of information.

### 3.2.1 Process Modelling Phase

In the first phase the Process Designer has to define the process models for a specific domain. We propose the usage of BPMN because of its readability (even for non-technologists), its widespread usage, and the good tool support. A standard-conforming BPMN model basically consist of the following components (compare Object Management Group (2008)):

- Flow Objects: Event, Activity (Task), and Gateway
- Connecting Objects: Message Flow, Sequence Flow, and Association
- Artifacts: Data Object, Group, and Annotation

A process contains at least one “pool” representing an organization or a role in the process. “Lanes” can be defined within a pool to sub-divide a pool (e.g., according to different roles or sub-systems).

In this first phase the Process Designer has to model each business process of the enterprise with non-functional descriptions. The following steps should be carried out for each particular business case:

1. Identify the systems comprised (e.g., ERM or CRM system) and add a new pool for each.
2. Analyze the business case, whether there are different roles or parts of a system and add as many lanes as needed within the specific pool.
3. Construct the process following the BPMN specification.
4. Mark each task which requests a service as “service task”. These tasks will later be refined by a Software Architect.

After the first phase, all process models will be defined and give an overview of all systems and roles used within the enterprise. The result can now be compared, for example, to a *Component Business Matrix* of the IBM methodology. The lane *Process Designer* in Figure 2 shows this step as the first task of the methodology process (also designed in BPMN).

### 3.2.2 Refinement Phase

The second phase is carried out by a Software Architect, since now the tasks of a process, which have been identified as service tasks in the first phase, will be described in detail. This phase is divided into two main actions:

1. First, the appropriate services for a service task are searched for, and their functional specification and behavior is described in an adequate way. For a Web Service WSDL will be used, while a RESTful service will be described using WADL. In this phase the resource URL, the input and output parameters as well as the service description are defined.
2. The data flow within the process models is described for each service. Any (specific) mapping of data from one service to another will be described with regard to the functional description of the service (meaning an exact overview of in- and outflow of data).

In the first part of this phase some kind of service repository can be employed. This could be a repository like [www.programmableweb.com](http://www.programmableweb.com) or any other internal or external repository of the given enterprise

(see Hagemann et al. (2007) for an overview). A certain difficulty obviously lies in searching for an adequate service which optimally fits the purpose of the business process. Notice that in SOA models, often a repository based on UDDI is proposed, which has once been designed for an automatic matching of services to a given description. The various problems that arise while detecting a suitable service for a SOA (not limited to UDDI) have been investigated in Letz (2007), and as shown in Hagemann et al. (2007), repositories based on UDDI are no really in use anymore.

So far the most complete service directory for RESTbased services is [www.programmableweb.com](http://www.programmableweb.com). If that is not sufficient, either a formal description needs to be connected with a service task (done by the Process Designer) which can be understood by a Software Architect, or the Process as well as the Software Designer need to communicate while refining the processes. In some cases, these two roles will be combined in one person. Also, the Software Architect should have a broad knowledge of the internal IT systems and hence be able to specify any service parameter. If a service task can be linked to a real service, the process model needs to be adapted. Therefore, adding or changing a pool or a lane within the process is needed to show the sequence flow from system to system. A pool called “Integrator” will always exist, from which all requests made by services will be started and their responses computed. To support the addition of functional descriptions to a task, the underlying BPMN metamodel needs to be extended with an artifact element called “functional description”. An instance of this element can then be connected to a task.

In the second part of the refinement phase, the data flow within a service is described. Through the information of the given WSDL and WADL descriptions, the parameters needed to call a service are outlined. Data objects can be used to define data that is saved or available throughout the process. Besides the sequence flow of a BPMN process, which defines the deterministic flow of the process, the message flow shows when and between which tasks of different pools messages are exchanged. Therefore, to define which parameter is used for a service request additional information is defined for the message flow connector. Even if our approach is designed to be technology independent, here the BPEL construct of assigning data to an activity will be used (see OASIS (2007)). We do so because on the one hand this construct supports the possibility to convert BPMN process models to BPEL processes and on the other hand because of the small footprint in contrast to other alternatives (like the *Web Service Flow Language* which builds heavily onto WSDL (Leymann 2001)).

Listing 1 shows how the assignment of parameters is defined for a message flow connector in XML notation. The **container** attribute in the **from** tag refers to a data object of the underlying process, where the **part** points to attributes within a data object. The **to** tag, in contrast, refers to a service task.

```
<assign>
  <copy>
    <from
      container="Session Data"
      part="customerId"/>
    <to
      container="Check Id"
      part="id"/>
    </copy>
  </copy> ... </copy>
</assign>
```

Listing 1: Assignment of data.

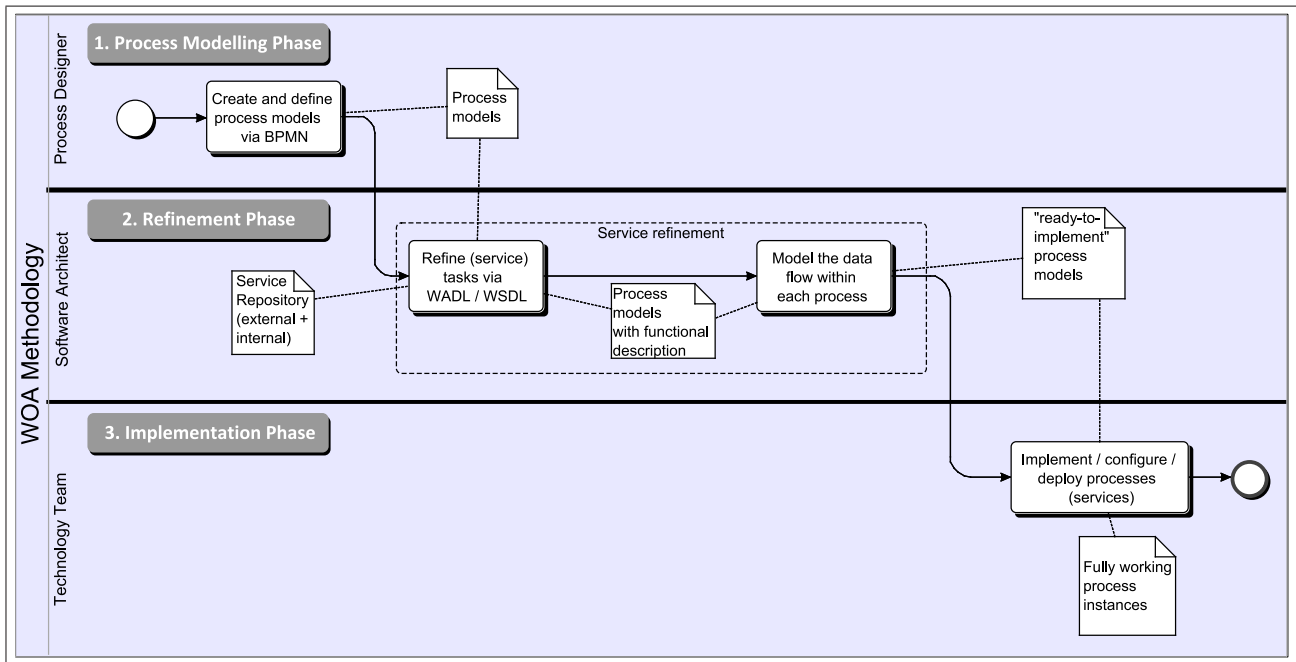


Figure 2: The three phases of our WOA methodology.

We mention that this phase is equivalent to the second step of the IBM method where services are bound to processes of the CBM matrix. The result of this phase are process models with a functional description of the specific services and their data flow.

So far there is no specific graphical representation for defining a functional description or a data flow element in a BPMN process, besides the normal artifact description. Therefore Figure 3 shows a prototypical construction of a process part, where a service task is refined with a WADL description and an in-going data flow from another task.

Since the BPMN specification explicitly allows the extension of BPMN models by artifacts, the necessary data flow element as well as the functional description are added as an artifact type. Additionally, the data flow artifact can only be bound to a message flow. In Figure 4 an overview of all the BPMN elements is shown including the gray elements for use in the second phase as extensions to the standard.

### 3.2.3 Implementation Phase

The third phase finally brings the WOA to life. The result of the two preceding phases are fully defined processes with the specific services and the integrator as a coordinating platform. One of the great hopes in service computing has always been a fully automatic code generation where a process is “designed, not programmed”. In some cases this is indeed possible, e.g., for a process which has no manual task in it and which uses mainly external services where the logic inside the integrator is minimal. Therefore, the BPMN process should be either directly run by an engine which can handle data in data objects, can send requests to any Web-based service (described by WADL or WSDL), and understands the flow logic of the integrator, or the process should be translated into a common workflow format like BPEL and then run in a workflow engine.

Since BPMN has no underlying metamodel, there are no generic workflow engines which could run a BPMN process. However, some tools are able to simulate a process. The latter approach has been inves-

tigated by Ouyang et al. (2006), and even the specification of BPMN addresses a conversion of BPMN patterns into BPEL4WS. Unfortunately, a standard BPEL process is not able to handle RESTful services out of the box, so the “RESTful BPEL” approach propagated by Overdick (2007) can be helpful in this respect. In most cases, however, the technology team has to integrate external services into the process flow, either by programming a wrapper for a service (in case that there is only something like a JavaScript API) or by taking a look at the *Service Level Agreements* (SLAs) of the service. Our approach aims at a WOA, so the integrator component should be a Web-based platform. This could be providers of Platform-as-a-Service solutions like *bungeeConnect* or *force.com*, or other integrating components.

Basically, the following steps are required for each process which is not automatically processable:

- Check for every pool that is not the integrator pool whether the services described are reachable and also whether or not the functional description is correct.
- Check the given SLAs and verify the guarantees if possible.
- Test the services and consider alternatives (think about load balancing and redundancy scenarios).
- Write the code needed for the integrator component and bind the services to the process.
- Conduct test runs of the finished processes.

These steps are closely related to a standard software engineering process, where the business cases are implemented, tested, and then enabled for production use. There is also a similarity to the third step of the IBM method, where the services are deployed in a SOA environment, or to the model of Mos et al. (2008), where the software components are finally deployed in an ESB.

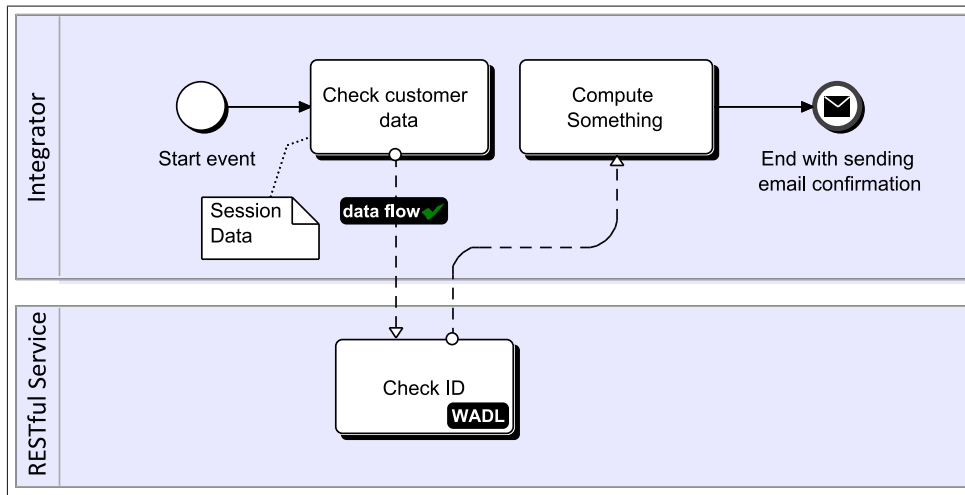


Figure 3: Refined process model after second phase.

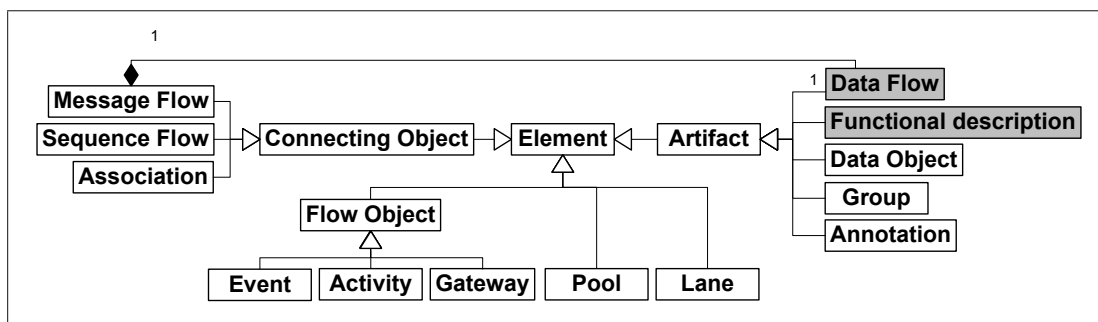


Figure 4: Extended BPMN object model.

#### 4 Applications

We now employ our WOA design methodology using the payment for items in the *Web Shop* which has been presented in detail in Thies & Vossen (2008) as a case study. The starting point of the methodology is the creation of a process model in the *Process Modelling Phase*. So the Process Designer models the Order process (at a coarse level). Only the “integrator” pool is modelled in the first place, where all the process steps take place. The process covers the following action:

“A customer chooses items from the web shop (so there is a filled shopping cart) and clicks on a “pay” button. If the customer is logged in, the “payment process” task is starting, otherwise he/she is forwarded to a login page. The task returns either a positive or a negative response. The latter case results in handling an error and end the process. If the payment is successfully done, the “order” task is starting, where the items in the shopping cart are saved as order in a CRM system. The process ends with an email notification to the customer.”

Two tasks are identified as services by the Process Designer and marked as “service tasks:”

1. *Payment process service*
2. *Create and save order service*

Both tasks are placeholders for a Web or RESTful service which will later be designed in the Refinement Phase. Figure 5 shows the final process of the first phase.

The following step in the *Refinement Phase* is to extend the process and hence to find suitable processes for the identified service tasks. For the payment process the *Amazon Flexible Payments Service* (FPS) is a candidate, the order process is covered by an imaginary test system (to show how a RESTful service would look like). First, two additional pools are created by the Software Architect, one for each external service, and each pool gets a task, which stands for the specific Web or RESTful service. After that the message flow is designed to reach the specific task within each new pool. For a better understanding of the process, the Software Architect adds grouping artifacts to the process, so that a former single task (like the payment process) is graphically outlined.

The important part now is to describe the tasks with functional descriptions. For the payment process, the WSDL document provided by Amazon is connected to the task. The order process gets a functional description via a WADL document. Both descriptions are indicated by a small black rectangle in the task symbol, which displays the special type of service description.

The final step in this phase is to define the data flow between the different pools in the process. As shown in Figure 6, the data flow is displayed as a data flow artifact connected with the message flow. It shows a (green) checkmark if the data flow is already defined; otherwise it shows a (red) cross. In our example, the data flow from task “Save order” to task “Create and save order” is not defined yet. We assume that the WADL definition of the order process needs parameters `apiKey`, `customerId`, and `shoppingCart` as input (see the `param` tags in lines



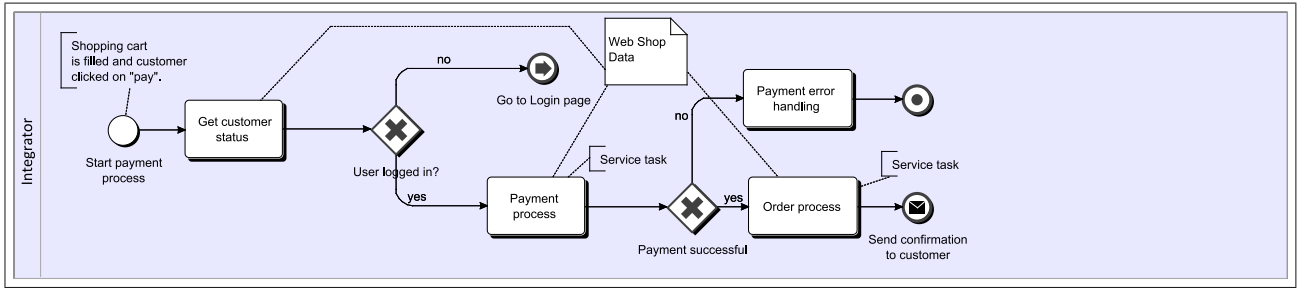


Figure 5: Order process after first phase.

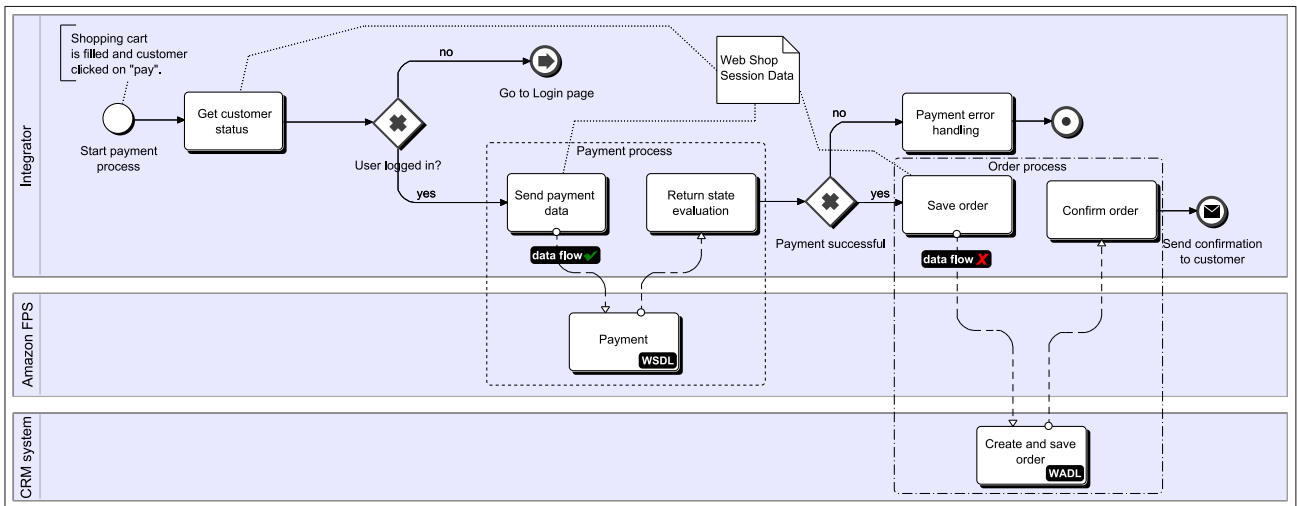


Figure 6: Extended order process during the second phase.

10, 13, and 14 of Listing 2). These parameters are held by the data object “Web Shop Session Data” in the integrator pool.

```

1 <?xml version="1.0" ?>
2 <application targetNamespace="urn:crmsystem"
3   xmlns:crm="http://api.testsystem.org/
   namespaces/">
4   <doc xml:lang="en" title="documentation">
5     Simple service for creating an order.
6   </doc>
7   ...
8   <resources base="http://api.testsystem.org/
   Order/V1/">
9     <resource path="{apiKey}/createOrder">
10      <param name="apiKey" style="query"/>
11      <method name="GET" id="search">
12        <request>
13          <param name="customerId" type="
14            xsd:string" required="true"/>
15          <param name="shoppingCart" type="
16            crm:shoppingcart">
17        </param>
18      </request>
19      <response>
20        <representation mediaType="text/plain"/>
21        <fault status="400" mediaType="text/
22          plain"/>
23      </response>
24    </method>
25  </resource>
26 </resources>
27 </application>

```

Listing 2: WADL description for the “Create and save order” service.

der”, where the parameters of the session data are mapped to the parameters of the RESTful service. Every parameter is copied via the from tag to the destination, the “Create and save order” task. With the data now defined, the service request can be implemented later by the Technology Team without knowing details about the business case.

```

<assign>
  <copy>
    <from
      container="Web Shop Session Data"
      part="apiKey"/>
    <to
      container="Create and save order"
      part="apikey"/>
    </copy>
    <copy>
      <from
        container="Web Shop Session Data"
        part="customerId"/>
      <to
        container="Create and save order"
        part="custId"/>
      </copy>
      <copy>
        <from
          container="Web Shop Session Data"
          part="shoppingCart"/>
        <to
          container="Create and save order"
          part="shoppingCart"/>
        </copy>
      </assign>

```

Listing 3: Assignment of data flow.

Listing 3 declares the data flow between “Save order” and the CRM system task “Create and save or-

The final *Implementation Phase* is not described in detail here. Basically, the integrator platform needs

to hold a data object with the session data of the Web Shop (therefore the Web Shop provides interfaces for reading and writing session data). The Technology Team also has to implement one Web Service request to Amazon FPS and a RESTful service request to the CRM system.

## 5 Conclusions and Future Work

In this paper we have presented a WOA design methodology which abstracts from technology and complex standards and only uses simple Web standards like HTTP, SSL, and XML for communication. In addition, the usage of the SOAP protocol is also needed in cases of Web Service-based systems. For the definition of a service we propose WADL (or WSDL for Web Services) and a simple data flow syntax to describe data mappings for any request within a process. No further standards are needed to describe a fully working WOA.

Our methodology comprises three phases, which are partially also reflected in some of the approaches for the construction of a SOA (as outlined in Section 2). Every phase addresses a distinct role, so that different kinds of knowledge about the processes of an enterprise can be covered. BPMN is used for all phases to define and model the respective business processes, because the benefit of BPMN is the readability and the extensibility. The usage of service descriptions is proposed and with WADL a high degree of technological independence is possible, since WADL is not bound to a specific service or protocol format.

As mentioned in the Introduction, we have seen that our methodology is not just purely conceptual, but rather a hybrid one that meshes conceptual as well as physical aspects of a WOA. We consider this a consequence of the fact that a WOA no longer needs to follow the strict layering of a SOA, but we believe that it is exactly this aspect what will make them more successful than SOAs.

One important aspect of services provided over the Web are SLAs, which define the guaranteed benefits of a provided service. SLAs are an agreement between a service customer and a service provider and often contain information about the availability, stability, and – most importantly – costs of a service. In this paper SLAs have not been considered, but in future work they will be discussed in detail in the context of the Refinement Phase of our methodology. This will lead to the possibility of estimating overall cost of a service, which will be helpful for a *return on investment* (ROI) calculation.

Tool support is another important factor for the use of a methodology. After some testing, we have chosen a plug-in called *SOA Tools Platform Project* (STP) for the Eclipse IDE, which allows for the modelling of BPMN processes. Choosing an Eclipse plug-in also makes it possible to extend the plug-in in the future. So all the added artifacts (and maybe more for SLA usage) to the BPMN metamodel can be constructed and used to create a WOA modelling tool in eclipse.

Another interesting part will be an elaboration of the integrator functionality. Besides existing platforms like *bungeeConnect* or *force.com*, which are so called Platform-as-a-Service providers, there is much space for the development of a BPMN workflow engine which is able to run and administrate BPMN processes directly.

**Acknowledgement.** The authors are grateful to the reviewers of this papers, whose comments have led to several improvements over an earlier version.

## References

- Arsanjani, A. & Allam, A. (2006), Service-oriented modeling and architecture for realization of an soa, in '2006 IEEE International Conference on Services Computing (SCC 2006)', 18-22 September 2006, Chicago, Illinois, USA', IEEE Computer Society, p. 512.
- Colombo, M., Di Nitto, E., Di Penta, M., Distant, D. & Zuccal, M. (2005), Speaking a common language: A conceptual model for describing service-oriented systems, in 'ICSOC 2005', Springer Verlag Berlin Heidelberg 2005, pp. 48–60.
- Fielding, R. T. (2000), Architectural Styles and the Design of Network-based Software Architectures: PhD Thesis, PhD thesis, University of California, Irvine.  
**URL:** <http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm>
- Ganci, J. (2006), *Patterns: SOA foundation service creation scenario*, IBM Redbooks, 1st ed. edn, International Technical Support Organization, Poughkeepsie NY.
- Hadley, M. J. (2006), 'Web application description language (wadl)'.  
**URL:** <https://wadl.dev.java.net/wadl20061109.pdf>
- Hagemann, S., Letz, C. & Vossen, G. (2007), Web service discovery – reality check 2.0, in 'Proc. 3rd International Conference on Next Generation Web Services Practices (NWeSP)', Seoul, Korea', pp. 113–118.
- Letz, C. (2007), Web Service Detection in Service-oriented Software Development: A Semantic Syntactic Approach, PhD thesis, Westfälische Wilhelms-Universität, Münster.
- Leymann, F. (2001), 'Web services flow language'.  
**URL:** <http://xml.coverpages.org/WSFL-Guide-200110.pdf>
- López-Sanza, M., J. Acuña, C., Cuestaa, C. E. & Marcosa, E. (2008), Modelling of service-oriented architectures with uml, in 'Electronic Notes in Theoretical Computer Science Vol. 194 Issue 4, Proceedings of the 6th International Workshop on the Foundations of Coordination Languages and Software Architectures (FOCLASA 2007)', pp. 23–37.
- Mos, A., Boulze, A., Quaireach, S. & Meynier, C. (2008), Multi-layer perspectives and spaces in soa, in 'SDSOA '08, May 11, 2008, Leipzig, Germany', ACM 2008.
- OASIS (2007), 'Web services business process execution language version 2.0'.  
**URL:** <http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.pdf>
- Object Management Group (2008), 'Business process modeling notation, v1.1'.  
**URL:** <http://www.omg.org/docs/formal/08-01-17.pdf>
- Ouyang, C., van der Aalst, W. M., Dumas, M. & ter Hofstede, A. H. (2006), 'Translating bpmn to bpel'.  
**URL:** <http://is.tm.tue.nl/staff/wvdaalst/BPMcenter/reports/2006/BPM-06-02.pdf>
- Overdick, H. (2007), Towards resource-oriented bpel, in '2nd Workshop on Emerging Web Services Technology', Aachen.

- Quartel, D. A., Steen, M. W. A., Pokraev, S. & van Sinderen, M. J. (2007), Cosmo: A conceptual framework for service modelling and refinement.
- Thies, G. & Vossen, G. (2008), Web-oriented architectures: On the impact of web 2.0 on service-oriented architectures (to appear), *in* Proc. 2008 IEEE Asia-Pacific Services Computing Conference, Yilan, Taiwan.
- Vossen, G. (2006), Have service-oriented architectures taken a wrong turn already?, *in* 'IFIP TC 8 International Conference on Research and Practical Issues of Enterprise Information Systems (CONFENIS 2006), Vienna, Austria', pp. xxiii-xxix.
- W3C (2004a), 'Semantic markup for web services'.  
**URL:** <http://www.w3.org/Submission/2004/SUBM-OWL-S-20041122/>
- W3C (2004b), 'Web service architecture'.  
**URL:** <http://www.w3.org/TR/ws-arch/wsa.pdf>
- W3C (2007), 'Web services description language 2.0'.  
**URL:** <http://www.w3.org/TR/wsdl20-primer/>
- Woods, D. & Mattern, T. (2006), *Enterprise SOA: Designing IT for Business Innovation*, O'Reilly Media, Inc.
- WSMO Working Group (2006), 'Web service modeling ontology'.  
**URL:** <http://www.wsmo.org/TR/d2/v1.3/20061021/>





# Multi-Level Domain Modeling with M-Objects and M-Relationships

Bernd Neumayr

Katharina Grün

Michael Schrefl

Department of Business Informatics - Data & Knowledge Engineering,  
Johannes Kepler University Linz, Austria  
E-Mail: {neumayr,gruen,schrefl}@dke.uni-linz.ac.at

## Abstract

Using traditional semantic data modeling, multi-level modeling can be achieved by representing objects in different abstraction hierarchies, namely classification, aggregation and generalization. This, however, leads to accidental complexity, complicating maintenance and extension. Several modeling techniques, like deep instantiation, powertypes and materialization, have been proposed to reduce unnecessary complexity in modeling objects at multiple levels. Multi-level objects (m-objects) and multi-level relationships (m-relationships) build on these results and provide a natural, intuitive representation of the *concretization* of objects and relationships along multiple levels of abstraction. By integrating aspects of the different abstraction hierarchies in a single concretization hierarchy, they improve readability and simplify maintenance and extension as compared to previous approaches. The discussion on conceptual modeling is complemented by a brief presentation of M-SQL, a data manipulation and query language for working with m-objects and m-relationships in an object-relational setting.

**Keywords:** Conceptual Modeling, Multi-Level Modeling, Multiple Abstraction

## 1 Introduction

Modeling domain objects at multiple levels of abstraction has received increased attention over the last years. It is nowadays also referred to as *ontological* multi-level modeling to contrast it from *linguistic* meta-level modeling, which relates to representing modeling language constructs in one or more higher levels, or meta models. Note that multi-level modeling is often associated solely with multiple classification levels, while this paper concerns, more generally, levels in classification, generalization and aggregation hierarchies.

Objects are frequently organized in hierarchies consisting of multiple levels. Examples are product hierarchies, dimension hierarchies in data warehouses, and taxonomies in general. E.g., a *product catalog* may consist of three levels, *category*, *model*, and *physical entity*. Sample instances of these levels could be *car*, *porsche911CarreraS*, and *myPorsche911CarreraS*, respectively.

Conceptual modeling of such hierarchies is straightforward if objects at each level are *uniform*,

i.e. have the same structure. As information systems grow in size or previously independent systems are integrated across related domains, the need to handle levels of similar, yet non-uniform objects arises. Objects at the same level in different subhierarchies may differ from each other. E.g., *product models* have a *listprice*, but *car models* (i.e., objects at level *model* belonging to *car*, an object at level *category*) have an additional attribute *maxSpeed*.

Non-uniformity may go beyond having differently described objects at the same level. Sub-hierarchies beneath two objects at the same level may differ from each other in that they introduce different additional levels. E.g., our product catalog may contain a product category *car*, which is described by an additional level *brand* (with objects like *porsche911*) between levels *model* and *physical entity*.

Such non-uniform hierarchies of domain objects at multiple levels of abstraction can be modeled using general modeling languages like UML through a combination of aggregation, generalization and classification (see left side of Fig. 1 for a simplified UML representation in which additional level *brand* is omitted). While, from a static perspective, this UML representation does not involve unnecessary or *accidental complexity*, the accidental complexity becomes apparent when introducing new domain concepts. Introducing a single domain concept (such as new product category *car*) requires next to introducing an instance *Car:CarCategory* to add multiple classes, (in our case *CarCategory*, *CarModel*, *CarPhysicalEntity*) as well as associated aggregation, generalisation, and instantiation relationships. Moreover, the same kind of classes need to be added for each new product category entered into our product catalogue. The redundancy and fragmentation caused by following such a modeling approach complicates maintenance and extension.

Several modeling techniques have been proposed in recent years to reduce accidental complexity in modeling domain objects at multiple levels of abstraction. The prevailing techniques are *materialization* (Goldstein & Storey 1994, Pirotte et al. 1994, Dahchour et al. 2002), *powertypes* (Odell 1998, Henderson-Sellers & Gonzalez-Perez 2005, Gonzalez-Perez & Henderson-Sellers 2006), and *deep instantiation* (Atkinson & Kühne 2001, Kühne & Schreiber 2007). While these techniques simplify multi-level modeling and support deep characterisation, they do not support (or at least do not directly support) non-uniform sub-hierarchies. *Materialization* provides for modeling concrete objects of category objects differently and comes with a very powerful concept for property definitions and propagations along the materialization hierarchy. The *powertype* approach provides for describing the common properties of subclasses of a class, i.e., one level of the generalization hierarchy, by powertypes. Ontological meta modeling

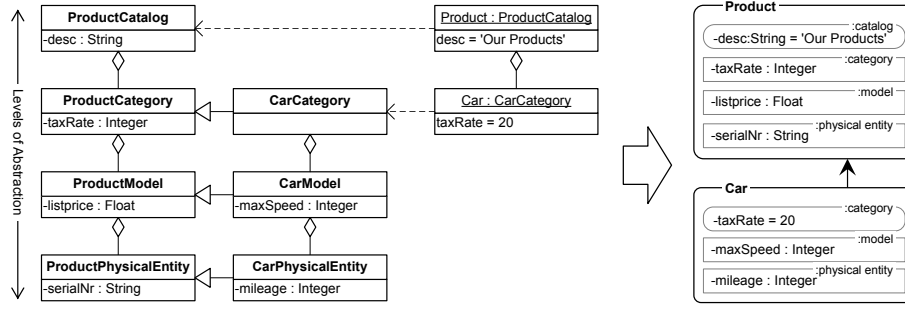


Figure 1: Basic Idea of Multi-Level Objects (simplified), Left: UML, Right: M-Objects

with *deep instantiation* provides for multiple levels of classification whereby an object at one level can describe the common properties for objects at each instantiation-level beneath that level.

*Multi-level objects* (m-objects) and *multi-level relationships* (m-relationships), introduced in this paper, build on these approaches and provide a natural, intuitive presentation of the *concretization* of objects and relationships along multiple levels of abstraction. The basic ideas of our approach are (i) to encapsulate the different levels of abstractions that relate to a single domain concept (e.g., the level descriptions, *catalog*, *category*, *model*, *physicalEntity* that relate to single domain concept, product catalog) into a single m-object (cf. *Product* with these levels at the right hand side in Fig. 1), and (ii) to integrate aspects of the different semantic abstraction hierarchies (aggregation, generalization, and classification) in a single *concretization* hierarchy. Note that we use the term aggregation in a broad sense referring to abstraction levels both with a typical whole-part flavor (e.g. *city-county-state*) or with a materialization flavor (e.g. *physical entity-model-category*). Furthermore our approach provides a naming scheme to address and query meta<sup>n</sup>-classes which are implicitly introduced with the levels of abstractions of a m-object.

A concretization relationship between two m-objects does not reflect that one m-object is at the same time an *instance of*, *component of*, and *subclass of* another m-object as a whole. Rather, a concretization relationship between two m-objects, such as *Car* and *Product* in Fig. 1, is to be interpreted in a multi-faceted way. M-object *Car* is an instance of m-object *Product* with respect to that m-object's second-top level, *category*, and gives values for attributes of that level, e.g., *taxRate*. M-object *Car* is a component of m-object *Product* considered as an object of that m-object's top-level, *catalog*. Each level of m-object *Car* can be seen as a subclass of the corresponding level of m-object *Product*.

Relationships between m-objects are likewise described at multiple levels, associating one or more pairs of levels of the linked m-objects, and are, thus, called m-relationships. M-relationships can again be interpreted in a multi-faceted way, once as relationship occurrence (or sometimes also called relationship instance or link) and once as relationship class (or sometimes also called relationship type, or set), or also as meta-relationship class. They take on these multiple roles depending on what pairs of linked levels one considers and on whether one considers the linked m-objects in an instance or a class-role. M-relationships may be concretized along the concretization hierarchy of linked m-objects. The richness of this concise, powerful modeling concept is discussed in detail in the paper.

The paper is organized as follows: Section 2 defines requirements for multi-level modeling based on a sample problem. Section 3 presents the concept

of m-objects and their consistent concretization in concretization hierarchies. Section 4 introduces m-relationships. Section 5 outlines a data definition and manipulation language M-SQL for m-objects. Section 6 gives an overview of related work and compares it to our approach. Finally, Section 7 concludes the paper.

## 2 Sample Problem and Requirements

In this section, we present our running example and discuss requirements for multi-level modeling approaches.

**Example 1** (Sample Problem). A sample online-store buys and sells *products*, which are described on at least three levels of abstraction: *physical entity*, *model*, and *category*. Each *product category* has associated a *tax rate*, each *product model* has a *list price*. Book editions, i.e. objects at level *model* that belong to *product category book*, additionally have an *author*. Our company keeps track of *physical entities* of *products* (e.g. copies of book *Harry-Potter4*), which are identified by their *serial number*. In addition to *books*, our company starts to sell *cars*, i.e. it introduces *car* as a new *product category*. Cars differ from books in that they are described at an additional level, namely *brand*, and in that they have additional attributes: *maxSpeed* at level *product model* and *mileage* at level *physical entity*. As our sample-online-store specializes on selling cars of *brand Porsche 911*, it wants to be able to register physical entities of this *car brand* at the *Porsche 911 club*. Our company further keeps track of *companies* that produce these products. Companies are likewise described at multiple levels: *industrial sector*, *enterprise*, and *factory*. Producers of cars belong to a specific *industrial sector*, namely *car manufacturer*. To track quality problems, our company also associates with each *physical entity* of *category car* the *factory* at which it was produced. Thereby, this *factory* must belong to the *enterprise*, which produces the *car model*.

To model domains, such as described in Example 1, multi-level modeling approaches may be benchmarked against how they support the following requirements:

- *Multiple levels of abstraction*: It should be possible to describe objects at different levels of abstraction. For example, products are described at levels *category*, *model*, and *physical entity*.
- *Extensibility*: Multi-level modeling should offer extensibility to model non-uniform sub-hierarchies, i.e. to add levels, attributes or relationships. For example, cars are described by an additional level, namely *brand*, and have additional attributes, like *maxSpeed*.

- *Avoid fragmentation and redundancy:* Every information concerning one domain object should be described local to that object to avoid fragmentation and redundancy. E.g., when introducing product category car, its information should be described in one object, which avoids distributing information along various objects and redundant modeling of relationships (cf. Figure 1).
- *Relationships:* The modeling approach should support relationships between objects and their specialization at different levels of abstraction. For example, it should be possible to associate products with companies and to further specialize this relationship by specifying that cars can only be produced by car manufacturers.
- *Queries:* To work with multi-level models, support for queries and navigation between levels is required. Queries can refer to different levels and objects, e.g. to retrieve all car models (i.e. all objects at level *product model* belonging to *product category car*). To determine e.g. the *taxRate* applying to *myPorsche911CarreraS*, navigation between levels of abstraction is necessary.

### 3 Multi-Level Objects

Based on multi-level modeling requirements presented in Section 2, we introduce, exemplify and formally define m-objects and their concretization in this section. We first describe m-objects and how one m-object can concretize another m-object. We then look at the roles which a m-object plays in such a concretization. Building on a formal definition of m-objects we specify rules for consistent concretization of m-objects and for consistent concretization hierarchies. Note that the basic concepts of m-object hierarchies could alternatively be described by providing a mapping to UML and OCL, or to the Higher-order Entity-Relationship Model (HERM) (Thalheim 2000). Our concise definitions are independent of a specific conceptual modeling language and well suited for defining consistency rules and operators for working with m-objects (cf. Section 5).

A m-object encapsulates and arranges abstraction levels in a linear order from the most abstract to the most concrete one. Thereby, it describes itself and the common properties of the objects at each level of the concretization hierarchy beneath itself. A m-object that concretizes another m-object, the parent, inherits all levels except for the top-level of the parent. It may also specialize the inherited levels or even introduce new levels. A m-object specifies concrete values for the properties of the top-level. This top-level has a special role in that it describes the m-object itself. All other levels describe common properties of m-objects beneath itself.

**Definition 1** (M-Object). A m-object  $o = (L_o, A_o, p_o, l_o, d_o, v_o)$  consists of a set of levels  $L_o$ , taken from a universe of levels  $L$  and a set of attributes  $A_o$ , taken from a universe of attributes  $A$ . The levels  $L_o$  are organized in a linear order, as defined by partial function  $\text{parent } p_o : L_o \rightarrow L_o$ , which associates with each level its parent level. Each attribute is associated with one level, defined by function  $l_o : A_o \rightarrow L_o$ , and has a domain, defined by function  $d_o : A_o \rightarrow D$  (where  $D$  is a universe of data types). Optionally, an attribute has a value from its domain, defined by partial function  $v_o : A_o \rightarrow V$ , where  $V$  is a universe of data values, and  $v_o(a) \in d_o(a)$  iff  $v_o(a)$  is defined.

We alternatively represent the order of levels by parent relation  $P_o \subseteq (L_o \times L_o)$ , where  $(l', l) \in P_o$  iff  $p_o(l') = l$ ; and denote its transitive closure by  $P_o^+$  and its transitive and reflexive closure by  $P_o^*$ . We denote the top-level by  $\hat{l}_o$  and the set of attributes associated with the top-level by  $\hat{A}_o$ . Further, we say that a m-object is at level  $l$  if  $l$  is its top-level. We denote by  $O$  the set of m-objects. M-objects, levels, and attributes have names, defined by function  $n : O \cup L \cup A \rightarrow N$ , where  $N$  is the universe of names. Levels of the same m-object must have distinct names. The names of attributes that are associated with the same level of a m-object must be unique as well.

**Example 2.** Consider m-object *Product* of Figure 2, which consists of four levels with one attribute each. Level *category* is the parent level of level *model*. Attribute *taxRate* is associated with level *category*, has domain *Integer* and does not define a value. The m-object is at level *catalog* as this level is its top-level. The top-level defines attribute *desc* of domain *String* and specifies value 'Our Products' for this attribute.

A m-object can *concretize* another m-object, which is referred to as its parent. A concretization-relationship comprises classification-, generalization- and aggregation-relationships between the levels of a m-object and the levels of its parent, as follows:

- *Classification - Instantiation:* Each m-object can be regarded as an instance of its parent m-object. In particular, the top-level of a m-object is an instance of the second top-level of its parent m-object. A m-object must adopt all levels from its parent except for the parents top-level and it can specify values for its attributes.
- *Generalization - Specialization:* The level descriptions of a m-object correspond to subclasses of the corresponding levels of its parent. The m-object can define new levels or add attributes to levels. Thereby, a m-object must not change the relative order of levels it inherits from its parent.
- *Aggregation - Decomposition:* The concretization path between m-objects of different levels expresses an aggregation hierarchy at the instance level. At the schema level the aggregation hierarchy is given by the order of levels of a m-object.

**Example 3** (concretization). M-object *Car* concretizes m-object *Product* in Figure 2. The concretization relationship expresses classification: m-object *Car* is instance of level *category* of m-object *Product* because level *category*, which is the first non-top-level of m-object *Product*, is its top-level. It also specifies a value for its attribute *taxRate*. M-object *Car* specializes m-object *Product* by introducing a new level *brand* and adding attribute *maxSpeed* to level *model* and attribute *mileage* to level *physical entity*. The level *model* of m-object *Car* can be regarded as a subclass of level *model* of m-object *Product*. Cars are further categorized into brands, models and physical entities. These levels define the aggregation hierarchy.

When organizing m-objects in a hierarchy, the name of each m-object, given by function  $n : O \rightarrow N$ , must be unique within the direct descendants of its parent. M-objects organized in a concretization hierarchy inherit properties and functions from their parent m-objects and, thus, may only be partially defined. A child m-object  $o'$  inherits from its parent m-object  $o$  all properties, i.e. levels, attributes (and relationships, cf. Section 4), and function definitions



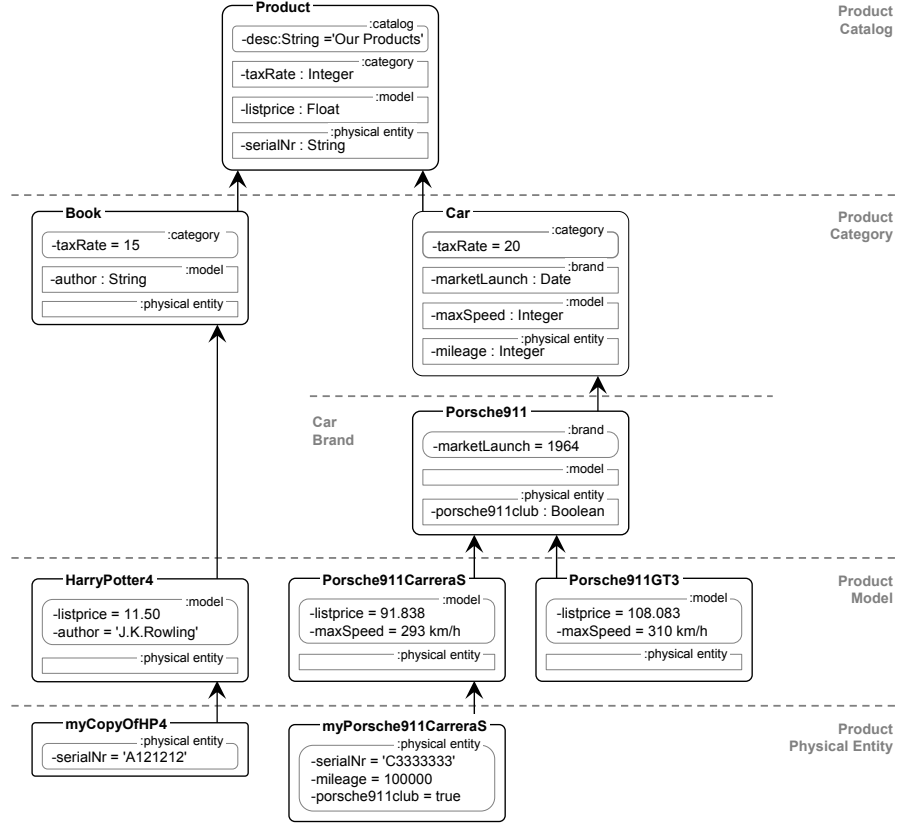


Figure 2: Concretization hierarchy of a product catalog along multiple levels (inherited attributes not shown)

beyond the parent's top-level. The inherited properties are available in all function definitions of the child ( $p_o', l_o', d_o', v_o'$ ). These functions, if only partially specified at the child, are extended for undefined arguments using the corresponding functions of the parent ( $p_o, l_o, d_o, v_o$ ). In the following definition of a consistent concretization, we assume that partially defined m-objects have already been extended as just described.

**Definition 2** (Consistent Concretization). A m-object  $o'$  is a consistent concretization of another m-object  $o$  iff

1. Each level of  $o$ , except for the top-level, is also a level of  $o'$ :  $(L_o \setminus \{\hat{l}_o\}) \subseteq L_{o'}$  (level containment)
2. All attributes of  $o$ , except for the attributes of the top-level, also exist in  $o'$ ,  $(A_o \setminus \hat{A}_o) \subseteq A_{o'}$  (attribute containment)
3. The relative order of common levels of  $o$  and  $o'$  is the same:  $l, l' \in (L_{o'} \cap L_o) : (l, l') \in P_o^+ \rightarrow (l, l') \in P_{o'}^+$  (level order compatibility)
4. Common attributes are associated with the same level, have the same domain, and the same value, if defined: For  $a \in (A_{o'} \cap A_o)$ :
  - (a)  $l_o(a) = l_{o'}(a)$  (stability of attribute levels)
  - (b)  $d_o(a) = d_{o'}(a)$  (stability of attribute domains)
  - (c)  $v_o(a)$  is defined  $\rightarrow v_o(a) = v_{o'}(a)$  (compatibility of attribute values)

**Example 4** (consistent concretization). M-object *Car* is a consistent concretization of m-object *Product* in Figure 2. Except for the top-level *catalog*, each level of *Product* also exists in *Car* (level containment). The same is true for all attributes, i.e. *Car*

only misses attribute *desc*, which is a top-level attribute of *Product* (attribute containment). The levels have the same order in both m-objects, e.g. in each m-object level *category* comes before level *model* (level order compatibility). Both m-objects associate attribute *taxRate* with level *category* (stability of attribute levels) and define domain *Integer* for this attribute (stability of attribute domains). If m-object *Product* defined a value for this attribute, the value would need to be 20 to ensure compatibility of attribute values.

M-objects do not only inherit levels and attributes from their parents, but can also introduce new levels and add attributes to levels. M-objects use names instead of numeric potencies to identify levels, which enables introducing additional levels without affecting existing navigation paths.

**Example 5** (extensibility). In Figure 2, m-object *Car* inherits levels *category*, *model* and *physical entity* from m-object *Product*. It adds level *brand* to define that cars are further categorized by their brand. Note that the additional level *brand* applies only to descendant m-objects of *Car* and not to other sub-hierarchies such as descendants of *Book*. Additionally, m-object *Car* extends level definitions of m-object *Product* by adding attribute *maxSpeed* to level *model* and attribute *mileage* to level *physical entity*.

By concretizing a set of m-objects, the m-objects form concretization hierarchies. There do not only exist consistency criteria for individual concretizations, but also for such hierarchies, which are explained in Definition 3.

**Definition 3** (Consistent Concretization Hierarchy of M-Objects). A concretization hierarchy of a set of m-objects  $O$  is defined by an acyclic relation  $H \subseteq O \times O$ , which forms a forest (set of trees). Let  $o, o' \in O$ , then  $o'$  is said to be a direct concretization of  $o$  or

$o'$  concretises  $o$ , if  $(o', o) \in H$ , and to be an indirect concretization of  $o$  if  $(o', o) \in H^+$ . A concretization hierarchy  $H$  of a set of  $m$ -objects  $O$  is consistent, iff

1. Each  $o \in O$  is a  $m$ -object according to Definition 1.
2. For each pair of  $m$ -objects  $(o', o) \in H$ ,  $o'$  is a consistent concretization of  $o$  according to Definition 2.
3. Each attribute and level is introduced at only one  $m$ -object:

- (a)  $a \in (A_o \cap A_{o'}) : \exists \bar{o} \in O : (o, \bar{o}) \in H \wedge (o', \bar{o}) \in H \wedge a \in A_{\bar{o}}$  (unique induction rule for attributes)
- (b)  $l \in (L_o \cap L_{o'}) : \exists \bar{o} \in O : (o, \bar{o}) \in H \wedge (o', \bar{o}) \in H \wedge l \in L_{\bar{o}}$  (unique induction rule for levels)

**Example 6** (concretization hierarchy). In Figure 2,  $m$ -object *Porsche911* is a direct concretization of  $m$ -object *Car* and an indirect concretization of  $m$ -object *Product*.  $M$ -objects *Book* and *Car* have a common attribute *taxRate*. To be consistent, these  $m$ -objects must have a common ancestor in the concretization hierarchy, which also defines this attribute, namely  $m$ -object *Product* in the example.

Each  $m$ -object plays multiple roles with regard to the classical semantic abstraction hierarchies: (i) A  $m$ -object can be regarded as an aggregate object of its children  $m$ -objects. (ii) It represents for each level of direct or indirect descendants the class of descendant  $m$ -objects of that level. Therefore it plays multiple class roles in multiple generalization hierarchies. (iii) Concerning classification hierarchies, it plays multiple class and meta<sup>*n*</sup> class (meta, meta-meta class, ...) roles: For a given  $n$ , where  $n$  is less than the number of levels of descendants, a  $m$ -object plays multiple meta<sup>*n*</sup> class roles for various combinations of levels of descendants. We discuss this in the remainder of this section.

A  $m$ -object represents for each level of direct or indirect descendants the class of descendant  $m$ -objects of that level. The notion of class refers first to the common structure (often referred to as *type* of a class) and second to the set of descendant  $m$ -objects at a specific level (often referred to as extension or members of a class). To refer to the set of  $m$ -objects at level  $l$  beneath  $m$ -object  $o$ , we write  $o\langle l \rangle$ . For example, *car(model)* refers to the set of  $m$ -objects at level *model* beneath  $m$ -object *Car*.

**Definition 4** (Class Extension). The class of  $m$ -objects of  $m$ -object  $o \in O$  at level  $l \in L_o$ , denoted as  $o\langle l \rangle$ , is defined by

$$o\langle l \rangle \stackrel{\text{def}}{=} \{o' \mid (o', o) \in H^* \wedge \hat{l}_{o'} = l\}.$$

**Example 7** (Class extension).  $M$ -object *Product* possesses the following classes:

```
Product<catalog> = {Product}
Product<category> = {Book, Car}
Product<model> = {HarryPotter4, Porsche911CarreraS,
                  Porsche911GT3}
Product<physical entity> = {myCopyOfHP4,
                           myPorsche911CarreraS}
```

A  $m$ -object does not only define a class for each of its levels, but also, implicitly, multiple meta<sup>*n*</sup>-classes. In this paper we limit the discussion on meta<sup>*n*</sup>-classes on their extensional role, i.e. a meta-class considered as a set of classes, and do not deal with the structural role of meta-classes (meta-types).

**Definition 5** (Meta-Classes and their Extensions).

1. For each  $m$ -object  $o$  and each pair of levels  $(l_0, l_1) \in P_o^+$  meta-class  $o\langle l_1 \langle l_0 \rangle \rangle$  is defined as the set of classes containing for each  $m$ -object  $o'$  that is a descendant of  $o$  at level  $l_1$  class  $o'\langle l_0 \rangle$ , i.e.

$$o\langle l_1 \langle l_0 \rangle \rangle \stackrel{\text{def}}{=} \{o'\langle l_0 \rangle \mid o' \in o\langle l_1 \rangle\}.$$

2. For  $o \in O$  and  $l_0 \in L_o$  meta-class  $o\langle \langle l_0 \rangle \rangle$  is the set of all classes containing  $m$ -objects at level  $l_0$  and that are descendants of  $o$ , i.e.,

$$o\langle \langle l_0 \rangle \rangle \stackrel{\text{def}}{=} \{o'\langle l_0 \rangle \mid (o', o) \in H^*\}$$

**Example 8** (meta-classes and their extension). The following meta-classes are based on  $m$ -object *Product*, as given by Figure 2 (Singleton-meta-classes like *Product<catalog<category>>* are omitted):

```
Product<category<model>> = {Car<model>, Book<model>}
Product<category<physical entity>> = {Car<physical entity>,
                                       Book<physical entity>}
Product<model<physical entity>> = {HarryPotter4<physical
                                     entity>, Porsche911CarreraS<physical
                                     entity>, Porsche911GT3<physical
                                     entity>}
```

```
Product<<model>> = {Product<model>, Car<model>,
                  Book<model>, Porsche911<model>}
```

```
Product<<physical entity>> = {Product<physical entity>,
                             Car<physical entity>, Book<physical
                             entity>, HarryPotter4<physical
                             entity>, Porsche911<physical
                             entity>, Porsche911CarreraS<physical
                             entity>, Porsche911GT3<physical
                             entity>}
```

**Definition 6** (Meta<sup>*n*</sup>-Classes and their Extensions).

1. For each  $m$ -object  $o$  and  $(n+1)$ -tuple of levels  $(l_0, l_1, \dots, l_n)$ , where  $n > 1$  and for  $i=1..n : (l_{i-1}, l_i) \in P_o^+$ , meta<sup>*n*</sup>-class  $o\langle l_n \langle l_{n-1} \dots \langle l_0 \rangle \dots \rangle \rangle$  is defined as the following set of meta<sup>*n-1*</sup>-classes:  
 $\{o'\langle^{n-1} l_{n-1} \dots \langle^0 l_0 \rangle \dots \rangle \mid o' \in o\langle l_n \rangle\}.$
2. For each  $m$ -object  $o$  and each level  $l_0 \in L_o$  meta<sup>*n*</sup>-class ( $n > 1$ ) is the following set of meta<sup>*n-1*</sup>-classes  
 $\{o'\langle^{n-1} l_{n-1} \dots \langle^0 l_0 \rangle \dots \rangle \mid (o', o) \in H^* \wedge (\forall i \in 1..n-1 : (l_{i-1}, l_i) \in P_o^+)\}$

**Example 9** (meta<sup>*n*</sup>-classes and their extensions). The following meta<sup>2</sup>-classes are based on  $m$ -object *Product*, as given by Figure 2 (Singleton-meta<sup>*n*</sup>-classes like *Product<catalog<category<model>>>* are omitted):

```
Product<category<model<physical entity>>> =
{Car<model<physical entity>>, Book<model<physical
entity>>>}
```

```
Product<<<physical entity>>>> = {Product<category<physical
entity>>>, Product<model<physical entity>>>,
Car<brand<physical entity>>>, Car<model<physical entity>>>,
Book<model<physical entity>>>}
```

## 4 Multi-Level Relationships

In this section we introduce, exemplify and formally define multi-level relationships ( $m$ -relationships) as a high-level modeling primitive for modeling relationships at multiple levels of abstraction. We exemplify that, when using traditional modeling and constraint languages (e.g. UML with OCL), dependencies between different abstraction levels of a relationship have to be modeled explicitly and specifically for each such relationship. We therefore advocate the use of  $m$ -relationships as a generic alternative. We show how  $m$ -relationships connect  $m$ -objects at multiple levels, how they can be concretized, and we look

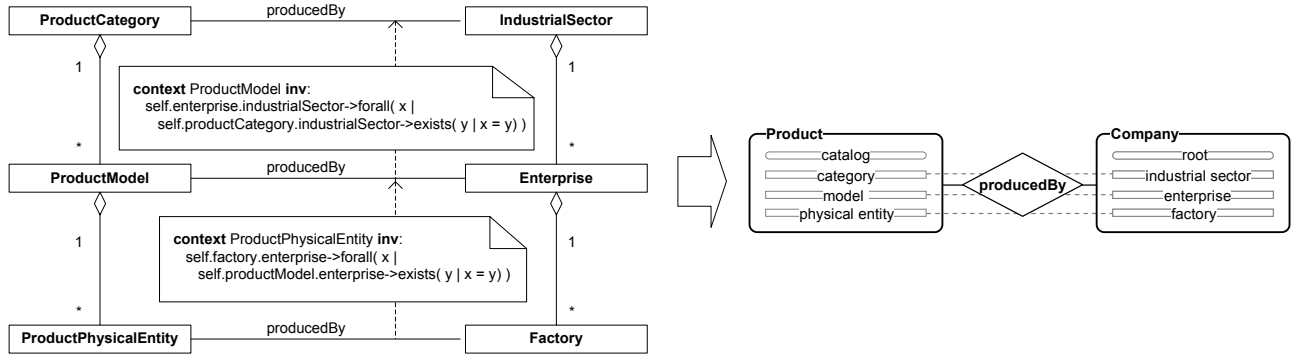


Figure 3: Basic Idea of Multi-Level Relationships (simplified), Left: UML and OCL, Right: M-Relationships

at the roles which they can play in concretizations. Finally, we define rules for consistent concretization of m-relationships. For simplicity, we consider only binary relationships in this paper and do not consider cardinality constraints.

So far we have shown how to reduce accidental complexity by describing domain-objects at multiple levels of abstraction through m-objects. Similarly, relationships occur at multiple levels of abstraction. We exemplify this below.

**Example 10** (Sample Problem continued). Consider our sample product catalog (Example 1). It describes products at three levels of abstraction (*physical entity*, *model* and *category*) and it describes the companies in which products are produced at three levels as well (*factory*, *enterprise*, and *industrial sector*). Relationship *producedBy* between levels *physical entity* and *factory* can be abstracted to *product model* and *enterprise* and further to *product category* and *industrial sector*. The following dependencies exist between these abstraction levels of relationship *producedBy*: (1) A *physical entity* can only be produced at a *factory* that belongs to an *enterprise* that produces the corresponding *product model*. (2) A *product model* can only be produced at an *enterprise* that belongs to an *industrial sector* that produces the corresponding *product category*. Specifically, product category *Car* is produced by industrial sector *Car-Manufacturer*, each model of car brand *Porsche911* is produced by *Porsche Ltd* and physical entity *my-Porsche911CarreraS* is produced by factory *Porsche Zuffenhausen*.

Using standard UML (without OCL) one could specialize an association (e.g. *producedBy* between *PhysicalEntity* and *Factory*) for each specialization of the associated classes (e.g. *Porsche911PhysicalEntity* and *PorscheFactory*). But it is not possible to express implicitly through UML modeling primitives alone the above identified dependencies between the different abstraction levels of a relationship (e.g. that a *physical entity* can only be produced by a *factory* that belongs to an *enterprise* that produces the correspondent *product model*).

Dependencies between abstraction levels of a relationship could be expressed explicitly using a constraint language like OCL. Thereby the relationship has to be split into multiple associations (each representing one abstraction level). Then, the dependencies can be modeled by explicit and relationship-specific constraints that are imposed from a higher to a lower level of abstraction of the *producedBy* relationship (cf. left part of Figure 3). In HERM (Thalheim 2000) dependencies between relationships are defined more concisely by path inclusion constraints, but still have to be defined explicitly and relationship-specific.

The basic idea of m-relationships is to provide a

high-level modeling primitive that (1) encapsulates different abstraction levels of a relationship (cf. right part of Figure 3), (2) implies extensional constraints between different abstraction levels of a relationship (as exemplified above), (3) supports heterogeneous hierarchies (e.g. additional level *brand* at category *Car*), and (4) can be exploited for navigating and querying. This leads to application models that are easier to define, understand, and maintain.

The use of a high-level modeling primitive for m-relationships is favorable over modeling multiple associations with associated explicit constraints; such as modeling a composition in UML is favorable over modeling a simple association and defining its composition-semantics using OCL. This approach is in line with the aim of semantic data models to "provide high-level modeling primitives to capture the semantics of an application environment" (Hammer & McLeod 1981).

M-relationships are analogous to m-objects in that they describe relationships between m-objects at multiple levels of abstraction. M-relationships are bidirectional. To facilitate referencing objects involved in binary relationships, we take, however, a (potentially) arbitrary directional perspective by considering one object in the *source* role and the other object in the *target* role of the relationship. Each m-relationship links a source m-object with a target m-object. Additionally, it connects one or more pairs of levels of the source and the target. These connections between source- and target-levels constitute the abstraction levels of a m-relationship. They define that m-objects at source-level are related with m-objects at target-level. (Remember that a m-object is at a certain level if this level is its top level.) We note that the generic roles *source* and *target*, which we introduced here for simplicity, may be easily replaced by relationship-specific role names.

**Definition 7** (M-Relationship). *Let  $O$  denote a universe of m-objects. A m-relationship  $r = (s_r, t_r, C_r)$  consists of two m-objects,  $s_r \in O$  and  $t_r \in O$ , linked by m-relationship  $r$ . M-relationship  $r$  connects levels of source m-object  $s_r$ ,  $L_{s_r} \subseteq L$ , to levels of target m-object  $t_r$ ,  $L_{t_r} \subseteq L$ , as specified by  $C_r \subseteq (L_{s_r} \times L_{t_r})$ .*

We denote by  $R$  the set of m-relationships. Each m-relationship  $r \in R$  has a name defined by function  $n : R \rightarrow N$  where the name must be unique *only* between the m-objects it links.

**Example 11** (m-relationship). To model that car models have a designer, Figure 4 introduces a relationship *designedBy* between source m-object *Car* and target m-object *Person*. More precisely, it links source-level *model* of m-object *Car* to target-level *individual* of m-object *Person*. While relationship *designedBy* only links one source-level with one target-level, relationship *producedBy* between *Product* and

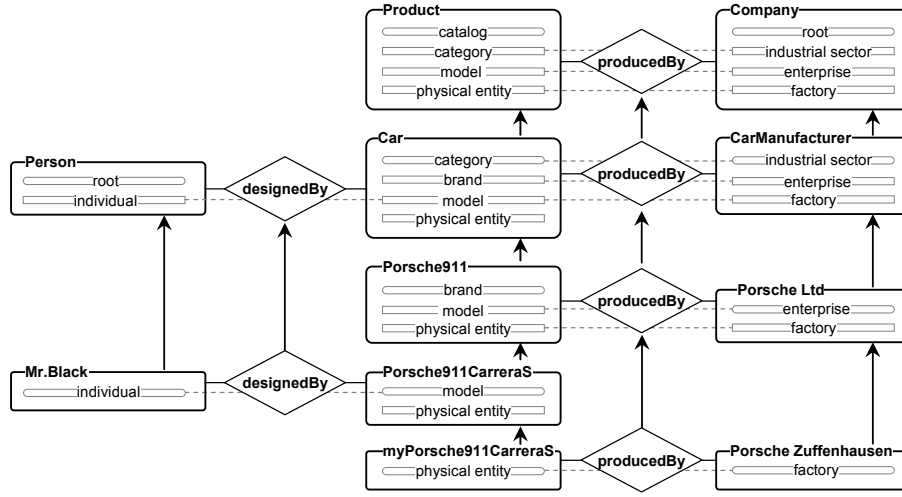


Figure 4: Product catalog modeled with m-objects and m-relationships

*Company* specifies multiple source- and target-levels. The relationship expresses that product categories are produced by industrial sectors, product models by enterprises and physical entities by factories.

Multiple pairs of source-target-levels express that the relationship exists at multiple levels of abstraction. To be consistent, a m-relationship must only specify source levels and target levels that exist in the source object and target object, respectively. Additionally, the relative order between source- and target-levels must be the same. Informally, this means that connections between source- and target-levels must not cross, a property which we call multi-level coherence.

**Definition 8** (Multi-Level Coherence). A m-relationship  $r$  is multi-level coherent iff for  $(l_s, l_t) \in C_r$ ,  $(l'_s, l'_t) \in C_r \rightarrow (((l_s, l'_s) \in P_{s_r}^* \wedge (l_t, l'_t) \in P_{t_r}^*) \vee ((l'_s, l_s) \in P_{s_r}^* \wedge (l'_t, l_t) \in P_{t_r}^*))$ .

**Example 12** (multi-level coherence). Looking at m-relationship *producedBy* between *Car* and *CarManufacturer* of Figure 4, multi-level coherence is fulfilled because the relative order of pairs of source-target-levels (*category* to *industrial sector*, *model* to *enterprise*, *physical entity* to *factory*) complies with the relative order of target-levels (*industrial sector*, *enterprise*, *factory*) at target m-object *Company*.

Like m-objects, m-relationships can be concretized. Concretizing a m-relationship means to substitute its source or its target for a direct or indirect concretization. The concretizes-relationship between two m-relationships expresses instantiation and/or specialization.

**Example 13** (concretization). Consider m-relationship *designedBy* between m-objects *Car* and *Person* in Figure 4. M-relationship *designedBy* between m-objects *Porsche911CarreraS* and *Mr.Black* concretizes this m-relationship. In this case, the concretization solely represents an instantiation, i.e. no further concretization is possible. Now look at m-relationship *producedBy* between m-objects *Product* and *Company*. M-relationship *producedBy* between *Car* and *CarManufacturer* concretizes this m-relationship, expressing that cars can only be produced by car manufacturers. This concretization further defines that each car model must be produced by enterprises that belong to the car manufacturer sector. Similarly, each physical entity of cars must be produced by a factory of the car manufacturer sector.

Dependent on the levels which a m-relationship connects, it can play various roles:

- *Relationship class*: A relationship that links a non-top-level of the source m-object with a non-top-level of the target m-object can be regarded as a relationship class. It defines that m-objects at the source-level may or must — depending on whether the relationship is optional or mandatory, a distinction which we do not discuss in this paper — have a relationship with m-objects at the target-level, which are concretizations of the target m-object. A m-relationship that links  $n$  pairs of non-top-levels, with  $n \geq 2$ , can further be seen as a meta <sup>$n-1$</sup>  relationship class because these links constrain how this relationship has to be further concretized.
- *Relationship occurrence*: A m-relationship that links the top-level of the source m-object with the top-level of the target m-object can be seen as relationship occurrence.
- *Shared relationship*: A m-relationship that links a non-top-level with a top-level is shared with lower levels, i.e. it is applicable to m-objects at this non-top-level. It cannot be further concretized by m-objects that still possess that non-top-level, but only by more concrete m-objects beneath.

**Example 14** (roles of m-relationships). In Figure 4, the relationship *designedBy* between source *car model* and target *person individual* can be regarded as a relationship class, defining that each car model may or must (see above) have a relationship with an individual person. In our example, there is such a relationship between the car model *Porsche911CarreraS* and the individual person *Mr.Black*. This relationship represents a relationship occurrence. Similarly, relationship *producedBy* between *Product* and *Company* connects source-level *model* of m-object *Product* with target-level *enterprise* of m-object *Company*, which is a relationship class. There is no explicit relationship occurrence between m-object *Porsche911CarreraS* and m-object *Porsche Ltd*. But there is a shared relationship between m-objects *Porsche911* and *Porsche Ltd* at the respective levels. This relationship defines that each *Porsche911 model*, i.e. including *Porsche911CarreraS*, is produced by *Porsche Ltd*. It also takes the role of a relationship class, constraining the producers of physical entities of *Porsche911* to factories of *Porsche Ltd*.



The relationship cannot be further concretized by m-object *Porsche911CarreraS*, which still possesses the (former) non-top-level *model*, but only at m-object *myPorsche911CarreraS*.

M-relationships can be concretized like m-objects. This is achieved by concretizing source and target objects. For a consistent concretization of a m-relationship to a more concrete m-relationship, both relationships must connect the same levels with regard to their common levels.

**Definition 9** (Consistent Concretization of M-Relationships). *A m-relationship  $r'$  is a consistent concretization of m-relationship  $r$  iff*

1.  $r'$  concretizes the source object or the target object of  $r$ , or concretizes both objects related by  $r$ :  $((s_{r'}, s_r) \in H^+ \wedge (t_{r'}, t_r) \in H^*) \vee ((s_{r'}, s_r) \in H^* \wedge (t_{r'}, t_r) \in H^+)$  (source and/or target concretization)
2.  $r$  and  $r'$  connect the same source-target levels for the levels shared between its source and target objects:  $l_s \in (L_{s_r} \cap L_{s_{r'}}), l_t \in (L_{t_r} \cap L_{t_{r'}}) \rightarrow ((l_s, l_t) \in C_r \leftrightarrow (l_s, l_t) \in C_{r'})$  (stability of source-target levels)

**Example 15** (consistent concretization). M-relationship *producedBy* between *Car* and *CarManufacturer* is a consistent concretization of m-relationship *producedBy* between *Product* and *Company* in Figure 4. Its source *Car* concretizes *Product* and its target *CarManufacturer* concretizes *Company* (target and/or source concretization). Both relationships have the same pairs of source-target-levels (*category* to *industrial sector*, *model* to *enterprise*, *physical entity* to *factory*) and thus fulfill stability of source-target levels.

Like m-objects, m-relationships are organized in a concretization hierarchy, defined by an acyclic relation  $H_R \subseteq R \times R$ , which forms a forest (set of trees). A concretization hierarchy  $H_R$  of a set of m-relationships  $R$  is *consistent*, iff for each pair of m-relationships  $(r', r) \in H_R$ ,  $r'$  is a consistent concretization of  $r$ .

## 5 Working with M-Objects

In this section, we show how to create and query m-objects and m-relationships, which can be achieved by extending existing data manipulation and query languages. As sample language, we briefly sketch multi-level SQL (M-SQL), which is a homogenous, light-weight extension to SQL-3, by examples. A full presentation of the language is beyond the scope of the paper. We first demonstrate how to create m-objects and m-relationships and then define *upward navigability*, and *navigation along m-relationships* as basis for querying m-objects and m-relationships.

To create multi-level models, M-SQL introduces an operator to create resp. concretize m-objects and m-relationships. Figure 5 exemplifies how to create m-object *Product* and its concretization *Car* using M-SQL. It also shows how to create the m-relationship *producedBy* between *Product* and *Company*, and its concretization to relationship *producedBy* between *Car* and *CarManufacturer*.

Upward navigation allows direct and stable access from a m-object  $o$  to its ancestor m-object at a specific level  $l$ , denoted by  $o[l]$ . For example, to navigate from m-object *myPorsche911CarreraS* to its ancestor at level *category*, i.e. *Car*, we write *myPorsche911CarreraS[category]*.

```
CREATE M-OBJECT Product
  (catalog (desc STRING),
   category (taxRate INTEGER),
   model (listPrice FLOAT),
   physicalEntity (serialNr STRING)),
SET desc = "Our Products";

CREATE M-RELATIONSHIP producedBy
  BETWEEN Product AND Company
  CONNECTING (category TO industrialSector,
             model TO enterprise,
             physicalEntity TO factory);

CREATE M-OBJECT Car UNDER Product
  ADD LEVEL brand UNDER LEVEL category
             (marketLaunch DATE),
  ALTER LEVEL model ADD (maxSpeed INTEGER),
  ALTER LEVEL physicalEntity ADD
             (mileage INTEGER),
  SET category.taxRate = 20;

CREATE M-RELATIONSHIP producedBy BETWEEN
  Car AND CarManufacturer UNDER
  producedBy BETWEEN Product AND Company;
```

Figure 5: Creating m-objects *Product* and *Car* and m-relationships *producedBy* as shown in Figures 2 and 4

**Definition 10** (Upward Navigation). *The ancestor m-object of m-object  $o \in O$  at level  $l \in L_o$ , denoted as  $o[l]$ , is defined by*

$$o[l] \stackrel{\text{def}}{=} o' : (o, o') \in H^* \wedge \hat{l}_{o'} = l.$$

Navigation along multi-level relationships is similar to dereferencing references in SQL-3. But, navigation in the context of concretization hierarchies of m-relationships is multi-faceted and expresses that all concretizations of a given m-relationship are to be traversed that lead to a m-object at some specified level.

**Definition 11** (Navigation along m-relationships). *The set of target m-objects (source m-objects) at level  $l$  reached by traversing a concretization of relationship  $r$  from source m-object (from target m-object, resp.)  $o$ , denoted as  $o \rightarrow r(l)$  ( $o \leftarrow r(l)$ , resp.), is defined by*

$$o \rightarrow r(l) \stackrel{\text{def}}{=} \{o' \in O \mid \exists r' \in R : (r', r) \in H_R^* \wedge s_{r'} = o \wedge t_{r'} = o' \wedge \hat{l}_{o'} = l\}$$

$$(o \leftarrow r(l) \stackrel{\text{def}}{=} \{o' \in O \mid \exists r' \in R : (r', r) \in H_R^* \wedge t_{r'} = o \wedge s_{r'} = o' \wedge \hat{l}_{o'} = l\}).$$

Class extension, upward navigation and navigation along m-relationships serve as basis for extending SQL-3's select-statement to support querying multi-level models. Identifiers of classes of m-objects of the form  $o(l)$  can be used where SQL-3 allows table names, primarily in the FROM-part of select statements. Upward navigation of the form  $o[l]$  is allowed where SQL-3 allows dereferencing references, primarily in the select- and where-part of select statements. Navigation along m-relationships is allowed where dereferencing of references is expected.

```
SELECT c.maxSpeed, c.listPrice *
  (1 + c[category].taxRate) AS grossPrice
FROM Car<model> c
WHERE c->Car-producedBy-CarManufacturer<enterprise>
      = PorscheLtd;
```

Figure 6: M-SQL query retrieving information about car models



**Example 16** (M-SQL query). Figure 6 demonstrates how to query all car models (class *Car(model)*) that are produced by car manufacturer *PorscheLtd* (navigation along concretisations of m-relationship *producedBy* between *Car* and *CarManufacturer*, identified by *Car-producedBy-CarManufacturer*, to m-objects at level *enterprise*), retrieving their *maxSpeed* and their *grossPrice* (using upward navigation).

Variables can be used in place of object names in class or meta<sup>n</sup>-class identifiers to refer to classes or meta<sup>n</sup>-classes in queries. Such identifiers can be used wherever table names are expected in SQL-3.

```
1)SELECT m.name, c.taxRate, m.listPrice
   FROM Product<category> c, c<model> m
   WHERE c.taxRate > 15 AND m<physicalEntity>.COUNT > 10

2)SELECT mp.name, mp.COUNT
   FROM Product<model<physical entity>> mp
```

Figure 7: Querying members of members and meta-classes

**Example 17** (Querying members of members). Query 1 in Figure 7 retrieves the name, the tax rate and the list price of each product model that belongs to a product category with a *taxRate* higher than 15 and which have more than 10 members at level physical entity. Query 2 retrieves for m-object *Product* the size of each class of physical entities at level model, i.e. in our running example: (*HarryPotter4(physical entity)*,1), (*Porsche911CarreraS(physical entity)*,1), (*Porsche911GT3(physical entity)*,0).

## 6 Related Work

Several approaches have been presented in recent years to model domain objects at multiple levels of abstraction. The prevailing techniques are *materialization* (Goldstein & Storey 1994, Pirotte et al. 1994, Dahchour et al. 2002), *powertypes* (Odell 1998, Henderson-Sellers & Gonzalez-Perez 2005, Gonzalez-Perez & Henderson-Sellers 2006), and *potency-based deep instantiation* (Atkinson & Kühne 2001, Kühne & Schreiber 2007). Note that each approach has been developed with a different focus in mind and that therefore the approaches complement one another. In this section, we compare these approaches with regard to the requirements presented in Section 2. For a detailed comparison of these different approaches see (Neumayr & Schrefl 2008). We conclude with an overview of further related work. For a comprehensive introduction to meta-modeling and multi-level modeling we refer to (Olivé 2007).

To the best of our knowledge, modeling of relationships at multiple levels of abstractions, as presented in this paper, has not been studied so far. To some extent, type parameters provided by some object-oriented languages can be used as a work-around (for an overview see (Bruce 2002)). In the absence of a modeling primitive for m-relationships in traditional conceptual modeling, multiple associations and explicit constraints between these associations have to be used to represent relationships at multiple levels of abstraction. In UML, these dependencies can be defined using the object constraint language (OCL) (cf. Figure 3). Similarly, in the Higher-order Entity-Relationship Model (HERM) (Thalheim 2000), these dependencies can be defined using path inclusion constraints.

*Materialization* (Goldstein & Storey 1994, Pirotte et al. 1994, Dahchour et al. 2002) is a generic relationship type, which can be regarded as a special kind of

aggregation (aggregation in its broader sense as used in this paper). Each materialization relationship connects two object classes, a more abstract with a more concrete one. In this respect, this approach supports multiple levels of abstraction. To add attributes to non-uniform objects (e.g. book vs. car), it provides a very flexible and powerful attribute propagation mechanism. Propagation of attribute definitions over more than one materialization level could be accomplished by composing attribute propagation types T1 and T3 (as sketched in (Neumayr & Schrefl 2008)), however this might harm readability of models. Materialization is not suited for adding additional levels to certain sub-hierarchies and it does not offer special support for relationships and querying.

*Powertypes* were introduced by Odell (Odell 1998) and are further investigated by Henderson-Sellers and Gonzalez-Perez (Henderson-Sellers & Gonzalez-Perez 2005, Gonzalez-Perez & Henderson-Sellers 2006), especially as a basis for software engineering methodology. The powertype approach also offers support for ontological multi-level modeling as discussed in this paper. A powertype describes common properties of direct subclasses of the class it is associated with. A cascaded setup of multiple powertypes can be used to describe more than one level beneath some class, i.e. common properties of indirect subclasses on a specific level of the generalization hierarchy. By specializing powertypes, this approach supports extensibility but does not completely avoid fragmentation and redundancy.

*Ontological metamodeling with potency-based deep instantiation* was introduced by Atkinson and Kühne (Atkinson & Kühne 2001). They also introduced the distinction between ontological and linguistic meta-modeling (Atkinson & Kühne 2003). Later, Kühne extended this approach to programming with multi-level models (Kühne & Schreiber 2007). Deep instantiation supports unbound classification levels. Considering the examples given by Kühne in (Kühne & Schreiber 2007), they would model our sample problem by combining deep instantiation and subclassing. Deep instantiation enables to define attributes of objects not only in their class, but also in a meta<sup>n</sup>-class. With regard to extensibility, it supports adding attributes, levels and relationships. However, new classification levels apply to all sub-hierarchies, which complicates modeling non-uniform sub-hierarchies. Concerning fragmentation, deep instantiation cannot avoid multiple unconnected model elements on different classification levels. While this approach supports relationships and their specialization, it does not consider multi-level relationships. As a work-around, similar results could be achieved by using type parameters as shown in (Kühne & Schreiber 2007).

In the field of database design and conceptual modeling, starting from the classical work on semantic data models (for an overview see (Schrefl et al. 1984, Hull & King 1987, Peckham & Maryanski 1988)), a lot of research has been centered around classification, generalization and aggregation. Concerning multiple classification levels, Klas and Schrefl provided an approach to deep instantiation for multi-level object-oriented databases (Klas & Schrefl 1995). Telos (Mylopoulos et al. 1990) and its successor ConceptBase (Jarke et al. 1995) allowed unbound classification levels. Troyer and Janssen embed schemas into Schema Object Types and discuss its implication on subtyping (Troyer & Janssen 1993).

## 7 Conclusion

Applications of m-objects and m-relationships are manifold. As information systems grow in size or previously independent systems are integrated across related domains, the need to handle levels of similar, yet non-uniform objects arises. This need arises not only for conceptual models as such, but also and especially for data warehouses. Data warehouses that aggregate data from multiple organisations or enterprises with heterogenous information systems have to deal with heterogenous dimension hierarchies. M-object hierarchies can be used to cope with these heterogeneities. Take for example a sales cube, modeled using the Dimensional Fact Model (Golfarelli et al. 1998), with three dimensions, *product*, *location* and *date*, where the *product* dimension consists of two levels, namely *model* and *category*. Using the basic Dimensional Fact Model, it is not possible to define that some product categories have additional levels, like in our sample problem (see Example 1), products of category *car* have an additional level, namely *brand*. In order to support modeling of such heterogenous dimension hierarchies in an extended Dimensional Fact Model one can simply replace the dimension *product* by m-object *product* and reuse its heterogenous level structure.

Apart from domain and data warehouse modeling, m-objects and m-relationships can serve as a basis for corresponding extensions of object-oriented programming languages, object-oriented and object-relational databases as well as for customizing information systems.

M-objects and m-relationships constitute a novel approach to multi-level domain modeling, which supports modeling objects and relationships at multiple levels of abstraction. The concretization hierarchy of m-objects combines aspects of the different abstraction hierarchies *classification*, *generalization* and *aggregation*, which avoids fragmentation and redundancy in modeling. Extending m-objects along the concretization hierarchy enables to model non-uniform objects as well as non-uniform sub-hierarchies. M-relationships link m-objects at multiple levels of abstraction and can be specialized along the concretization hierarchy. Working with m-objects and m-relationships is supported by M-SQL, which extends SQL-3 to create and query multi-level models.

## References

- Atkinson, C. & Kühne, T. (2001), The essence of multilevel metamodeling, in M. Gogolla & C. Kobryn, eds, 'Proceedings of the 4<sup>th</sup> International Conference on the UML 2000, Toronto, Canada', LNCS 2185, Springer Verlag, pp. 19–33.
- Atkinson, C. & Kühne, T. (2003), 'Model-driven development: A metamodeling foundation.', *IEEE Software* **20**(5), 36–41.
- Bruce, K. B. (2002), *Foundations of Object-Oriented Languages: Types and Semantics*, The MIT Press, Cambridge, Massachusetts.
- Dahchour, M., Pirotte, A. & Zimányi, E. (2002), 'Materialization and its metaclass implementation.', *IEEE Trans. Knowl. Data Eng.* **14**(5), 1078–1094. 0605.
- Goldstein, R. C. & Storey, V. C. (1994), 'Materialization', *IEEE Trans. Knowl. Data Eng.* **6**(5), 835–842.
- Golfarelli, M., Maio, D. & Rizzi, S. (1998), 'The dimensional fact model: A conceptual model for data warehouses', *Int. J. Cooperative Inf. Syst.* **7**(2-3), 215–247.
- Gonzalez-Perez, C. & Henderson-Sellers, B. (2006), 'A powertype-based metamodelling framework', *Software and System Modeling* **5**(1), 72–90.
- Hammer, M. & McLeod, D. (1981), 'Database description with sdm: a semantic database model', *ACM Trans. Database Syst.* **6**(3), 351–386. 0605.
- Henderson-Sellers & Gonzalez-Perez (2005), 'Connecting powertypes and stereotypes', *Journal of Object Technology* **4**, 83–96.
- Hull, R. & King, R. (1987), 'Semantic database modeling: survey, applications, and research issues', *ACM Comput. Surv.* **19**(3), 201–260.
- Jarke, M., Gellersdörfer, R., Jeusfeld, M. A. & Staudt, M. (1995), 'Conceptbase - a deductive object base for meta data management.', *J. Intell. Inf. Syst.* **4**(2), 167–192.
- Klas, W. & Schrefl, M. (1995), *Metaclasses and Their Application - Data Model Tailoring and Database Integration*, Springer.
- Kühne, T. & Schreiber, D. (2007), Can programming be liberated from the two-level style: multi-level programming with deepjava, in R. P. Gabriel, D. F. Bacon, C. V. Lopes & G. L. S. Jr., eds, 'OOPSLA', ACM, pp. 229–244.
- Mylopoulos, J., Borgida, A., Jarke, M. & Koubarakis, M. (1990), 'Telos: Representing knowledge about information systems', *ACM Trans. Inf. Syst.* **8**(4), 325–362.
- Neumayr, B. & Schrefl, M. (2008), Comparison criteria for ontological multi-level modeling, Technical Report 08.03, Department of Business Informatics - Data & Knowledge Engineering, Johannes Kepler University Linz, Austria.  
**URL:** <http://www.dke.jku.at/papers/TR0803.pdf>
- Odell, J. J. (1998), *Advanced Object-Oriented Analysis & Design Using UML (also published as James Odell: Power Types. JOOP 7(2): 8-12 (1994))*, Cambridge University Press, chapter Power Types, pp. 23–32.
- Olivé, A. (2007), *Conceptual Modeling of Information Systems*, Springer.
- Peckham, J. & Maryanski, F. (1988), 'Semantic data models', *ACM Comput. Surv.* **20**(3), 153–189.
- Pirotte, A., Zimányi, E., Massart, D. & Yakusheva, T. (1994), Materialization: A powerful and ubiquitous abstraction pattern., in J. B. Bocca, M. Jarke & C. Zaniolo, eds, 'VLDB', Morgan Kaufmann, pp. 630–641. 0605.
- Schrefl, M., Tjoa, A. M. & Wagner, R. (1984), Comparison-criteria for semantic data models, in 'ICDE', IEEE Computer Society, pp. 120–125.
- Thalheim, B. (2000), *Entity-Relationship Modeling: Foundations of Database Technology*, Springer.
- Troyer, O. D. & Janssen, R. (1993), On modularity for conceptual data models and the consequences for subtyping, inheritance & overriding, in 'ICDE', IEEE Computer Society, pp. 678–685.

# Reverse Engineering of XML Schemas to Conceptual Diagrams

Martin Nečaský

Department of Software Engineering  
Charles University, Prague, Czech Republic  
Email: necasky@ksi.mff.cuni.cz

## Abstract

It is frequent in practice that different logical XML schemas representing the same reality from different viewpoints exist. There is also usually a conceptual diagram modeling the reality independently of the viewpoints. It is important to keep the XML schemas and conceptual diagram consistent as they are both utilized for different purposes. In practice, this is however rarely the case. In this paper, we propose a reverse engineering method as a solution to this problem. We provide a semi-automatic algorithm that produces mappings of components of the XML schemas to components of the conceptual diagram. The method only provides suggestions for the mapping and manual participation of a domain expert is therefore required.

**Keywords:** xml schema, conceptual model, reverse engineering.

## 1 Introduction

Without any doubt, XML is currently a de-facto standard for data representation. Its popularity is given by the fact that it is well-defined, easy-to-use and, at the same time, enough powerful. With a growing popularity of XML, there is also a growing need for effective methods and tools for designing XML data. In recent research, there has appeared several approaches that concentrate on so called *forward engineering* methods. These approaches usually apply the ER model (such as (Dobbie et al. 2000), (Mani 2004)) or UML class model (such as (Routledge et al. 2002) or (Bernauer et al. 2003)). They suppose designing a conceptual diagram of the problem domain first. After that, a representation in an XML schema language is derived automatically from the conceptual diagram. Usually, the applied XML schema language is XML Schema (Thompson et al. 2004). There exist recent surveys of this area, e.g. (Nečaský 2008, Domínguez et al. 2007, Bernauer et al. 2004).

However, these approaches have not considered a crucial fact that information systems usually do not apply only one XML format but several (e.g. for sending purchase orders, browsing product catalogs, viewing sales reports, etc). These XML formats represent different views

```
<order-request
  issue-date="20/06/2008">
  <ship-addr>
    <street>X</street>
    <postcode>X</postcode>
    <city>X</city>
  </ship-addr>
  <bill-addr>
    <street>Y</street>
    <postcode>Y</postcode>
    <city>Y</city>
  </bill-addr>
  <messenger mno="M45"/>
  <ol product-code="P475">
    <price>458</price>
    <quantity>3</quantity>
  </ol>
</order-request>

<distribution
  product-code="P475">
  <rgn name="CZ">
    <purchase no="3820192"
      amount="5"/>
    <purchase no="3820199"
      amount="2"/>
  </rgn>
  <rgn name="SK">
    <purchase no="3820298"
      amount="4"/>
  </rgn>
</distribution>
```

Figure 1: Purchase Request and Product Distribution XML Documents

on the data in a system. It is natural since there are different groups of users who view the data (e.g. about customers, products or purchases) from different perspectives. Therefore, one concept can be represented in various XML formats in different ways.

Example 1 demonstrates the situation. There are two XML documents. The XML document on the left demonstrates an XML format for purchase requests. The other represents an XML format for product sales reports. Both formats represent products, customers and purchases but in different XML structures, i.e. with different XML elements and attributes.

Current approaches are not sufficient for designing such XML formats since they automatically translate a conceptual diagram into an XML schema. Therefore, this leads to augmenting a conceptual diagram for the needs of the corresponding XML format. It means enriching the diagram with syntactical constructs that model hierarchical structure (since XML is hierarchical in its nature), deciding whether a given part of the data should be represented as an XML element or attribute, etc. In the result, there is a separate conceptual diagram for each XML format. However, a conceptual diagram should be abstracted from the details of a concrete logical model (e.g. XML) and from a particular user view (e.g. XML format).

In our previous work (Nečaský 2007, 2008), we have developed a conceptual model for XML that overcomes the disadvantages of the existing approaches. We present the model briefly later in this paper. We can anticipate the main idea standing behind the model. It is a division of the conceptual modeling process to two steps. In the first step, a conceptual diagram describing a problem domain independently of its representation in various XML formats is designed. In the second step, required XML formats are designed on the base of the conceptual diagram.

In this paper, we further extend our conceptual model with so called *reverse engineering* capabilities. We are motivated by a common situation in current information systems. As we have already discussed, there are usually

This paper was supported by the Grant Agency of Czech Republic (project 201/09/0990) and by the Ministry of Education of the Czech Republic (grant MSM0021620838).

Copyright ©2009, Australian Computer Society, Inc. This paper appeared at the Sixth Asia-Pacific Conference on Conceptual Modelling (APCCM 2009), Wellington, New Zealand, January 2009. Conferences in Research and Practice in Information Technology (CRPIT), Vol. 96, Markus Kirchberg and Sebastian Link, Ed. Reproduction for academic, not-for profit purposes permitted provided this text is included.

several XML formats each described by an XML schema. Usually, there also exists a UML class diagram or ER diagram, that describes the data at the conceptual level. This conceptual diagram is usually developed at the beginning of the development process but never used later. Consequently, the XML schemas are designed separately from the conceptual diagram and are therefore not explicitly mapped to the conceptual diagram. A common consequence is that the XML schemas are inconsistent with the conceptual diagram as well as with each other. This makes not only their design but also their maintenance harder (e.g. their evolution, change impact analysis, etc.). Suppose for example that we need to make a change in an XML schema, e.g. to remove an XML element declaration. This change can cause additional changes in other XML schemas as well to keep them consistent with each other. Today, it is necessary to make these additional changes manually which is time-consuming and error-prone. If we had a conceptual diagram and each XML schema was mapped to the conceptual diagram, we could propagate the change to the conceptual diagram first and from here to the other XML schemas automatically. This would automate the evolution process significantly.

Reverse engineering of XML schemas, as we understand it in this paper, means to map existing XML schemas to an existing conceptual diagram. Because manual reverse engineering would be time-consuming and error-prone activity, we try to find a semi-automatic method, i.e. a method that is still performed by a domain expert but supported by a computer.

**Related Work.** There exist several approaches to reverse engineering of XML schemas to UML class diagrams such as (Jensen et al. 2003)(Yang et al. 2006). There is also a recent survey in (Yu & Steele 2005). Their common characteristics is that they automatically translate an XML schema to a corresponding UML class diagram. However, the following facts, that we consider crucial for a reverse-engineering method to be successfully applicable in practice, have not been addressed yet:

1. UML class diagram modeling data at the conceptual level usually exists. Often, it is created during initial phases of the development process and rarely used later during system maintenance.
2. Several XML schemas describing different XML formats applied in the system exist. These formats reflect different perspectives of particular users. However, the XML schemas are mostly designed separately from the UML class diagram.

If we apply existing approaches on a set of XML schemas, we get a set of separate UML class diagrams each being the result of an automated reverse engineering of the respective XML schema. These UML class diagrams are not interrelated neither with each other nor with the existing UML class diagram. Therefore, we can not utilize the reverse engineered UML class diagrams for, e.g. XML schema maintenance mentioned earlier.

**Contribution** In this paper we try to overcome the described disadvantages of existing approaches to reverse engineering of XML schemas. For this purpose, we apply the Model-Driven Architecture (MDA) (Miller & Mukerji 2003) which considers two types of models. Platform-Independent Model (PIM) enables one to model data independently of any representation in any concrete data model. Platform-Specific Model (PSM) allows one to model representation of data modeled by the PIM diagram using constructs of a selected data model such as XML.

In our approach, a PIM diagram is a UML class diagram that models data independently of its representation in XML, i.e. it is a conceptual diagram of the data. A PSM diagram is also a UML class diagram but models how the data is represented in a particular XML format. It models an XML schema of this XML format at the conceptual level. At this point, it is important to stress explicitly that

an XML schema and its PSM diagram represent a particular view on the system while the system is described independently of this view by the PIM diagram. The XML schema represents the view at the logical level, without any connection to the PIM diagram, while the PSM diagram represents the view at the conceptual level, with an explicit mapping to the PIM diagram.

In this paper, we consider an existing PIM diagram and a set of XML schemas. We suppose that the XML schemas were designed manually without any explicit relationship to the PIM diagram. The XML schemas could also be imported to the system, e.g. because of needs of communication with other systems. This is a common situation in practice. Instead of automatic translation of each XML schema to a separate UML class diagram, we propose a semi-automatic method that maps components of the XML schemas to components of the PIM diagram. For each XML schema, the method constructs a PSM diagram that models the XML schema at the conceptual level and describes the semantics of its components in terms of the PIM diagram. The result is that the XML schemas are mapped to the PIM diagram. In other words, the PIM diagram integrates the XML schemas at the conceptual level. This facilitates maintenance of the XML schemas as well as other related tasks (e.g. their integration, data storage, etc.). For example, if a new user requirement appears, corresponding changes are made in the PIM diagram and are automatically propagated through the reverse engineered PSM diagrams to the XML schemas. A change can also be done in an XML schema or its PSM diagram and automatically propagated through the PIM diagram to the other XML schemas.

Reverse engineering of XML schemas with an exploitation of an existing PIM diagram has not been studied yet to our best knowledge. This brings a new challenge of exploitation of semi-automatic schema mapping techniques ((Shvaiko & Euzenat 2005) (Chiticariu et al. 2007)) in reverse engineering techniques.

## 2 XML Schema

In this section we briefly describe the XML Schema language (Thompson et al. 2004) as it is an essential technology for this paper. It describes syntactical structure of XML documents, i.e. what XML elements and attributes can be used. XML Schema is an XML dialect, i.e. schemas are XML documents. An example XML schema is depicted in Figure 2. Since XML Schema provides a lot of constructs, we consider only basic ones to keep the complexity of the paper acceptable.

The basic construct is *element declaration*. It is specified by an element `element` and declares elements with a given name. An element declaration has a simple or complex type. A simple type specifies that the declared elements contain text values. A complex type specifies that the elements have attributes and contain child elements. E.g., there is an element declaration with a name `order-request` at line 02 in Figure 2. It has assigned a complex type `OrderRequest` and declares elements `order-request` with attributes and child elements defined by the complex type. An element declaration with a name `street` has assigned a simple type `string`. It declares elements `street` containing a string value.

*Attribute declaration* is specified by an element `attribute` and is used to declare attributes. It has a name and a simple type specifying values of the declared attributes. E.g., there is an attribute declaration with a name `issue-date` at line 13.

Each simple or complex type is described by an XML Schema construct called *type definition*. It is specified by an element `simpleType` or `complexType`, respectively. A type definition has a name that identifies the type

```

01 <schema xmlns="http://www.w3.org/2001/XMLSchema">
02 <element name="order-request" type="OrderRequest"/>
03 <complexType name="OrderRequest">
04 <sequence>
05 <element name="ship-addr" minOccurs="0"
06   type="Address"/>
07 <element name="bill-addr" minOccurs="0"
08   type="Address"/>
09 <choice>
10 <element name="messenger" type="Messenger"/>
11 <element name="van" type="Van"/>
12 </choice>
13 <element name="ol" type="OL"
14   maxOccurs="unbounded"/>
15 </sequence>
16 <attribute name="issue-date" type="date"/>
17 </complexType>
18 <complexType name="Address">
19 <sequence>
20 <element name="street" type="string"/>
21 <element name="postcode" type="string"/>
22 <element name="city" type="string"/>
23 </sequence></complexType>
24 <complexType name="Messenger">
25 <attribute name="mno" type="string"/></complexType>
26 <complexType name="Van">
27 <attribute name="vno" type="string"/></complexType>
28 <complexType name="OL">
29 <sequence>
30 <element name="price" type="decimal"/>
31 <element name="quantity" type="integer"/>
32 </sequence>
33 <attribute name="product-code" type="string"/>
34 </complexType></schema>

```

Figure 2: XML Schema

in the XML schema<sup>1</sup>. In this paper we are interested only in complex types. E.g., there is a complex type definition `OrderRequest` at line 03. A complex type definition contains so called *content model* which defines child elements. It further contains a set of attribute declarations that define attributes. Even though XML Schema provides several constructs for defining content models, we consider only a construct *sequence*. It contains a list of element declarations and models an ordered sequence of child elements. It can also contain *choice* constructs. A *choice* contains one or more element declarations and models that only one of them can appear among child elements in a parent element.

### 3 Conceptual Model

In this section, we briefly introduce our MDA-based conceptual model for XML. For its full description see (Nečaský 2008).

#### 3.1 Platform-Independent Model

As a platform-independent model (PIM), we use UML class diagrams. Even though UML provides more constructs, we consider only classes with attributes and binary associations. As we mentioned in the introduction, a PIM diagram describes the problem domain independently of a representation of the domain in a concrete data model such as relational or XML.

**Example 1** Figure 3 shows a PIM diagram of a company. A class *Purchase* models purchases. It has attributes *purchase-no* and *date* modeling relevant purchase characteristics. An association connecting *Purchase* and *Item* models that purchases contain items. Associations can have labels that explicitly specify the semantics for the reader. For example, *Purchase* and *Address* are associated by two associations with labels *ship* and *bill*, respectively.

<sup>1</sup>There can also be anonymous definitions but we omit them in this paper

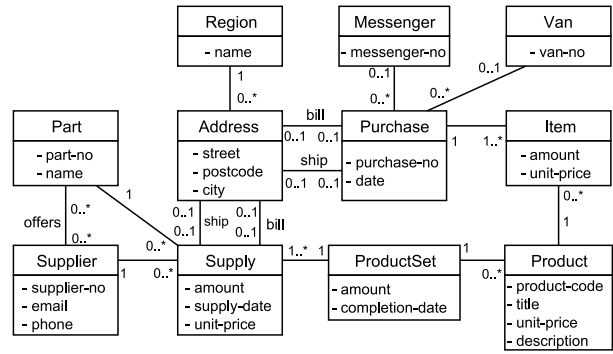


Figure 3: PIM diagram

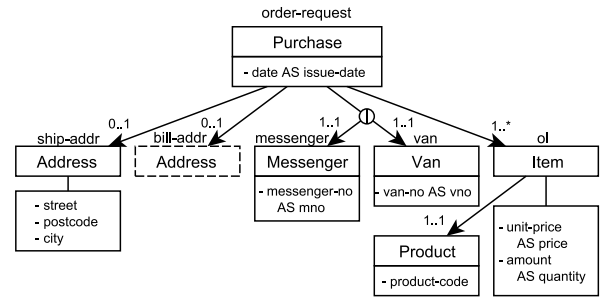


Figure 4: Purchase PSM diagram

#### 3.2 Platform-Specific Model

As a platform-specific model (PSM), we use UML class diagrams extended with some constructs for modeling XML specific details. A PSM diagram models a given XML format. There can be more PSM diagrams derived from a PIM diagram each modeling a separate XML format. The PSM diagram describes not only the structure of the format but also its semantics in terms of the PIM diagram since it uses its classes and associations.

**Example 2** Figure 4 depicts a PSM diagram derived from the PIM diagram depicted in Figure 3. It models the XML format for purchase requests demonstrated by the XML document depicted in Figure 1 on the left.

A PSM diagram is a tree. It can be translated to a representation in an XML schema language (see (Nečaský 2008)). Basic PSM building blocks are UML classes and directed binary UML associations.

A PSM class  $C_{psm}$  represents a PIM class  $C$  and specifies how instances of  $C$  are represented in the modeled XML format.  $C_{psm}$  has the same name as  $C$  and zero or more attributes of  $C$ . For an attribute  $Attr$ , an expression *Attr AS a* specifies that  $Attr$  is assigned with an alias  $a$ . We use an alias if we want an attribute to be represented in the XML format with a name different from its original name.  $C_{psm}$  further contains an ordered list of zero or more PSM associations going from  $C_{psm}$ . This list is called *content* of  $C_{psm}$ .

A PSM association  $A_{psm}$  goes from a parent class to a child class. It represents a construction called *nesting join* that describes the semantics of  $A_{psm}$  in terms of components of the PIM diagram. We introduce nesting joins later in this section. Here, we anticipate that a nesting join specifies nesting of instances of PIM classes represented by the PSM classes connected by  $A_{psm}$ .

A PSM class  $C_{psm}$ , that represents a PIM class  $C$ , models that  $C$  is represented in XML documents as a set of XML attributes and sequence of XML elements. The XML attributes are modeled by the attributes of  $C_{psm}$ . An attribute  $Attr$  models an XML attribute with a name given by an alias of  $Attr$  or name of  $Attr$  (if  $Attr$

does not have an alias). The XML elements are modeled by the content of  $C_{psm}$ . Let  $A_{psm}$  be a PSM association in the content going to a PSM class  $C'_{psm}$  that represents a PIM class  $C'$ .  $A_{psm}$  models that the XML code representing an instance  $c'$  of  $C'$  is contained in the XML code representing an instance  $c$  of  $C$  if  $c'$  is nested in  $c$  by  $A_{psm}$ .

$C_{psm}$  can have assigned a label called *element label*. It is displayed above  $C_{psm}$ . If  $C_{psm}$  has an element label  $l$ , the XML elements and attributes modeled by  $C_{psm}$  are enclosed in an XML element named  $l$ . Otherwise, they are propagated to the closest ancestor with an element label. An existence of such an ancestor is ensured since each root PSM class must have an element label.

PSM further contains constructs for modeling XML syntactic details. An *attribute container* can be contained in the content of a PSM class  $C_{psm}$  and contains one or more attributes of  $C_{psm}$ . It models that the attributes are represented as XML elements not attributes. A *content choice* can also be contained in the content of a PSM class  $C_{psm}$  and models variants in the content of  $C_{psm}$ . It contains two or more PSM associations going from  $C_{psm}$  and specifies that only one of them can be instantiated for each instance of  $C_{psm}$ . A *structural representative*  $R_{psm}$  is a PSM class that inherits attributes and content of another PSM class  $C_{psm}$ . Both  $R_{psm}$  and  $C_{psm}$  must represent the same PIM class.  $R_{psm}$  can have its own element label.

**Example 3** Assume again the PSM diagram depicted in Figure 4. Its root  $Purchase_{psm}$  represents *Purchase*. It has an element label *order-request*. Further, it has an attribute *date* with an alias *issue-date*. The other attribute *purchase-no* of *Purchase* is not represented. The content of  $Purchase_{psm}$  contains a PSM association going to a PSM class  $Address_{psm}$  and PSM association going to a structural representative of  $Address_{psm}$ . A structural representative is displayed as a class but with a dashed line. The associations are followed by a content choice. It is displayed by a circle with an inner '|' and contains two PSM associations going to  $Messenger_{psm}$  and  $Van_{psm}$ . It specifies that each purchase has only a messenger or van but not both. Finally, there is a PSM association going to  $Item_{psm}$ . It nests items in corresponding purchases. The diagram also contains attribute containers. E.g.,  $Address_{psm}$  has its attributes *street*, *postcode* and *city* separated to an attribute container.

An XML document depicted in Figure 1 on the left is an XML representation of a purchase as modeled by the PSM diagram in Figure 4. Because the root  $Purchase_{psm}$  has the element label *order-request*, the XML representation of the purchase is enclosed in an XML element *order-request*. Its attribute *date* with the alias *issue-date* specifies that a purchase date is represented as an XML attribute *issue-date* of *order-request*.

The PSM association going to  $Address_{psm}$  with the element label *ship-addr* specifies that a ship address is nested in the purchase. The XML representation of the ship address is modeled by  $Address_{psm}$ . It is enclosed in an XML element *ship-addr* because of the element label. Similarly, the XML representation of a bill address is enclosed in an XML element *bill-addr*. Because the attributes of  $Address_{psm}$  are separated to the attribute container, the XML elements *ship-addr* and *bill-addr* have child elements *street*, *postcode* and *city*.

The PSM association going to  $Item_{psm}$  with element label *ol* specifies that items are nested in the purchase. An XML representation of each item is enclosed in an XML element *ol*. The PSM association going from  $Item_{psm}$  to  $Product_{psm}$  specifies that each item has nested a purchased product. Because  $Product_{psm}$  does not have an element label, the XML representation of the product, which is XML attribute *product-code*, is not enclosed in a separate XML element but propagated to the upper XML element *ol*.

### 3.3 Nesting Joins

Each PSM class represents a PIM class. It means that semantics of the PSM class is specified by the PIM class. In this section, we propose a formalism for specifying semantics of PSM associations. Informally, semantics of a PSM association specifies what child instances are nested in a given parent instance.

Basically, semantics of a PSM association  $A_{psm}$  can be specified by a PIM association  $A_{pim}$ . Assume that  $A_{psm}$  goes from a PSM class  $C_{psm}$  to a PSM class  $C'_{psm}$  where the PSM classes represent PIM classes  $C$  and  $C'$ , respectively. The semantics of  $A_{psm}$  can be specified by  $A_{pim}$  if  $A_{pim}$  connects  $C$  and  $C'$ . In that case  $A_{psm}$  nests an instance of  $C'$  in an instance of  $C$  if the instances are connected by  $A_{pim}$ .

Since PSM diagrams represent views on PIM diagrams, we need a more advanced mechanism to specify semantics of PSM associations. The first generalization discussed in this paper is specification of semantics by a path in a PIM diagram instead of PIM association. The principle is similar to the previous case since a PIM association can be comprehended as a path of length 1. Informally, a path goes from a PIM class  $C$  to a PIM class  $C'$ . If the semantics of a PSM association  $A_{psm}$  is described by this path,  $A_{psm}$  nests an instance of  $C'$  in an instance of  $C$  if the instances are connected by the path. We define paths in PIM diagrams formally in the following definition.

**Definition 1** A PIM path  $P$  is an expression  $C_1 - \dots - C_n$  where  $C_1, \dots, C_n$  are PIM classes and for each  $1 \leq i < n$ , there is a PIM association connecting  $C_i$  with  $C_{i+1}$ . If there are two or more associations connecting  $C_i$  and  $C_{i+1}$ , we need to distinguish the required association by its name  $l$  and write  $(l, C_{i+1})$  instead of  $C_{i+1}$ . We say that  $P$  goes from  $C_1$  to  $C_n$ .  $C_n$  is called terminal class of  $P$ .

Consistency between a PIM diagram and derived PSM diagrams is ensured by the following definition.

**Definition 2** If a PIM path  $C_1 - \dots - C_n$  specifies the semantics of a PSM association  $A_{psm}$ , we say that  $A_{psm}$  represents the PIM path.  $A_{psm}$  can represent the PIM path only if  $C_{psm}$  represents  $C_1$  and  $C'_{psm}$  represents  $C_n$ .

Formally, the semantics of a PSM association representing a PIM path is defined by the following definition.

**Definition 3** Let  $A_{psm}$  be a PSM association representing a PIM path  $C_1 - \dots - C_n$ . Let  $c_1$  and  $c_n$  be instances of  $C_1$  and  $C_n$ , respectively.  $A_{psm}$  nests  $c_n$  in  $c_1$  if  $c_n \in c_1[C_1 - \dots - C_n]$ .  $c_1[C_1 - \dots - C_n]$  denotes a set that is defined recursively as follows:

$$c_i[C_i - \dots - C_n] = \bigcup_{c_{i+1} \in c_i(C_{i+1})} c_{i+1}[C_{i+1} - \dots - C_n], \\ c_n[C_n] = \{c_n\}$$

where  $c_i(C_{i+1})$  is a set of all instances of  $C_{i+1}$  connected with  $c_i$  by the respective PIM association. If  $c_n \in c_1[C_1 - \dots - C_n]$ , we say that  $c_n$  is accessible by  $P$  from  $c_1$ .

**Example 4** The semantics of all PSM associations depicted in Figure 4 can be specified by PIM associations depicted in Figure 3. For example, the PIM association named *ship* connecting PIM classes *Purchase* and *Address* specifies the semantics of the PSM association going from  $Purchase_{psm}$  to  $Address_{psm}$ .

On the other hand, there can be PSM associations whose semantics can not be described simply by a PIM association. Suppose for example a PSM diagram depicted in Figure 5 on the right. There is a PSM association going from  $Product_{psm}$  to  $Region_{psm}$ . However, there is no PIM association in the PIM diagram in Figure 3 connecting *Product* and *Region*. We need to specify that the PSM association nests in each product a list of regions



from where the product was purchased. This semantics is specified in terms of the PIM diagram by a PIM path *Product–Item–Purchase–(bill,Address)–Region*.

We further propose a generalization of PIM paths for describing semantics of PSM associations. This generalization is called *nesting join*. Suppose again a PSM association  $A_{psm}$  with semantics specified by a PIM path going from a PIM class  $C$  to  $C'$ . This semantics can also be interpreted as a grouping of instances of  $C'$  by  $A_{psm}$ . More precisely, instances of  $C'$  form a group if they are nested by  $A_{psm}$  in the same instance of  $C$ . Therefore, each instance of  $C$  has a nested group of instances of  $C'$ . This group is defined by  $A_{psm}$ . We can extend this mechanism to grouping instances of  $C'$  not only by its parent but also one or more ancestors. The best way to explain this is to show an example.

**Example 5** Suppose a PSM diagram depicted in Figure 5 on the left. There is a PSM class  $Supply_{psm}$ . It has ancestors  $Supplier_{psm}$ ,  $Part_{psm}$  and  $ProductSet_{psm}$ .  $Supply_{psm}$  represents a PIM class *Supply* which models supplies of parts. Parts are supplied by suppliers. A product set is produced from supplied parts. For each supply, we therefore have its supplier, supplied part, and product set. In the PSM diagram, we want to model an XML structure where supplies are grouped by suppliers, parts and products sets. More precisely, supplies form a group if they have the same supplier, part and product set. To represent this grouping in the required hierarchical structure, the PSM association going from  $Supplier_{psm}$  to  $Part_{psm}$  must nest a part in a supplier if there is a supply of the part by the supplier. Further, the PSM association going from  $Part_{psm}$  to  $ProductSet_{psm}$  must nest a product set in a part, that is nested in a given supplier, if there is a supply of the part to the product set by the supplier. Finally, the PSM association going from  $ProductSet_{psm}$  to  $Supply_{psm}$  must nest a supply in a product set, that is nested in a given part and supplier, if the supply is supplied by the supplier and supplies the part to the product set. We can also say that a supply is nested in a product set in the context of a part and supplier.

We use nesting joins to describe such semantics. A nesting join must specify a grouped PIM class (e.g. *Supply*), joined PIM classes (e.g. *Supplier* with *Part*, *Part* with *ProductSet*, or *ProductSet* with *Supply*, respectively), and PIM classes that form the context for the grouping (e.g. empty context for the former PSM association, *Supplier* for the second, and *Supplier* and *Part* for the other, respectively).

In the rest of this section, we introduce nesting joins formally. Before this, we define some auxiliary terms.

**Definition 4** We say that a PIM path is direct if it does not contain the same PIM class twice or more times. The only exception is the beginning and end of the path.

**Definition 5** Let  $P$  be a PIM path.  $rev(P)$  denotes  $P$  in the reversed direction. It goes from  $C_n$  to  $C_1$  through the same PIM associations as  $P$ .

Now, we are ready to define nesting joins formally.

**Definition 6** A nesting join is described by an expression

$$C^{P_1, \dots, P_k} [P \rightarrow Q]$$

$C$  is a PIM class whose grouping is described by the nesting join.  $P_1, \dots, P_k$  are direct PIM paths that go from  $C$  to PIM classes that form a context for the grouping.  $P$  and  $Q$  are direct PIM paths that go from  $C$ .  $P$  and  $Q$  are called parent and child of the nesting join. The arrow between  $P$  and  $Q$  specifies an orientation of the nesting join. To simplify the expression, we can leave the starting  $C$  from  $P_1, \dots, P_k$ ,  $P$  and  $Q$ , since they must start with  $C$  anyway.

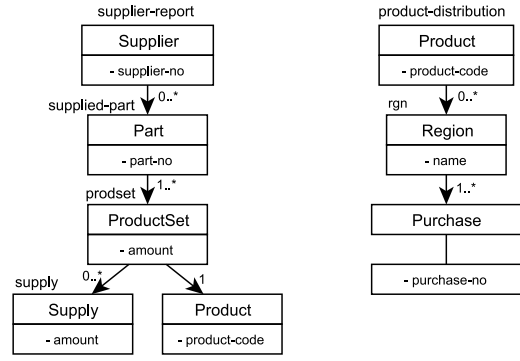


Figure 5: Supplier Report and Product Distribution PSM Diagrams

Consistency between a PIM diagram and derived PSM diagrams is ensured by the following definition.

**Definition 7** If a nesting join  $C^{P_1, \dots, P_k} [P \rightarrow Q]$  specifies the semantics of a PSM association  $A_{psm}$ , we say that  $A_{psm}$  represents the nesting join. Let  $A_{psm}$  goes from a PSM class  $C_{psm}$  to  $C'_{psm}$  that represent PIM classes  $C$  and  $C'$ , respectively.  $A_{psm}$  can represent the nesting join only if the following conditions are satisfied:

- (J1)  $C$  and  $C'$  are terminal classes of  $P$  and  $Q$ , respectively
- (J2) if  $k > 0$ , there is a PSM association that goes to  $C_{psm}$  and represents a nesting join  $C^{P_1, \dots, P_{k-1}} [P_k \rightarrow P]$

This ensures that PIM classes that form the context of  $C^{P_1, \dots, P_k} [P \rightarrow Q]$  are also represented in the PSM diagram as ancestors of the parent of  $A_{psm}$ . Formally, the semantics of a PSM association representing a nesting join is given by the following definition.

**Definition 8** Let  $A_{psm}$  be a PSM association representing a nesting join  $C^{P_1, \dots, P_k} [P \rightarrow Q]$ . For each  $k$ -tuple  $p_1, \dots, p_k$ , where  $p_i$  is an instance of the terminal class of  $P_i$ ,  $A_{psm}$  nests an instance  $q$  of the terminal class of  $Q$  in an instance  $p$  of the terminal class of  $P$  if there is an instance  $c$  of  $C$  such that  $p \in c[P]$ ,  $q \in c[Q]$ , and  $\forall 1 \leq i \leq k : p_i \in c[P_i]$ . We say that  $q$  is nested in  $p$  in the context of  $p_1, \dots, p_k$ .

**Example 6** Assume again the PSM diagram depicted in Figure 5 on the left. As we explained before, its hierarchical structure represents grouping of instances of *Supply*. Therefore, we need nesting joins to specify the semantics of PSM associations forming this structure. The PSM association going from  $Supplier_{psm}$  to  $Part_{psm}$  nests in each supplier a list of supplied parts. Formally, it nests an instance *part* of *Part* in an instance *supplier* of *Supplier* if there exists an instance *supply* of *Supply* such that *supplier*  $\in$  *supply*  $\llbracket$  *Supply–Supplier*  $\rrbracket$  and *part*  $\in$  *supply*  $\llbracket$  *Supply–Part*  $\rrbracket$ . This semantics is specified by a nesting join

$$Supply[Supply - Supplier \rightarrow Supply - Part]$$

We can also leave the grouped class *Supply*, i.e. we can write

$$Supply[Supplier \rightarrow Part]$$

The PSM association going from  $Part_{psm}$  to  $ProductSet_{psm}$  nests in each part a list of product sets to which the part was supplied. Moreover, the superior supplier has to be considered, i.e. the part contains only the product sets to which it was supplied by the supplier. Such semantics is specified by

<pre> &lt;supplier-report   supplier-no="S1"&gt; &lt;supplied-part   part-no="P121"&gt;   &lt;prodset amount="1200"     product-code="PR47"&gt;     &lt;supply amount="800"/&gt;     &lt;supply amount="1600"/&gt;   &lt;/prodset&gt; &lt;/supplied-part&gt; &lt;/supplier-report&gt; </pre>	<pre> &lt;supplier-report   supplier-no="S2"&gt; &lt;supplied-part   part-no="P121"&gt;   &lt;prodset amount="2000"     product-code="PR32"&gt;     &lt;supply amount="1500"/&gt;   &lt;/prodset&gt; &lt;/supplied-part&gt; &lt;/supplier-report&gt; </pre>
--	---

Figure 6: Supplier Report XML Documents

$$Supply^{Supplier}[Part \rightarrow ProductSet]$$

Formally, for each superior instance *supplier* of *Supplier*, the PSM association nests an instance *productset* of *ProductSet* in an instance *part* of *Part* if there exists an instance *supply* of *Supply* such that  $supplier \in supply[Supply - Supplier]$ ,  $part \in supply[Supply - Part]$ , and  $productset \in supply[Supply - ProductSet]$ . In other words, it joins *ProductSet* instances with *Supplier* and *Part* instances on the described conditions and groups the result by *Supplier* and *Part*.

The PSM association going from  $ProductSet_{psm}$  to  $Supply_{psm}$  nests in each product set a list of supplies supplied by the superior supplier and supplying the superior part. This semantics is specified by

$$Supply^{Supplier, Part}[ProductSet \rightarrow]$$

Two example XML documents modeled by this PSM diagram are depicted in Figure 6. The left-hand side XML document is for a supplier with number 'S1' and the right-hand side is for a supplier with number 'S2'. We can see that both supplied the same part with number 'P121'. However, the part has nested in each XML document different product set depending on the superior supplier. This is modeled by the context of the PSM association going from  $Part_{psm}$  to  $ProductSet_{psm}$ .

**Example 7** We can also use longer PIM paths in nesting joins. Assume the PSM diagram depicted on the right hand side of Figure 5. The PSM associations in the diagram represent respectively the following nesting joins:

$$Purchase[Item-Product \rightarrow (bill, Address)-Region]$$

$$Purchase^{Item-Product}[(bill, Address)-Region \rightarrow]$$

The former specifies that the PSM association going from  $Product_{psm}$  to  $Region_{psm}$  nests an instance *region* of *Region* in an instance *product* of *Product* if there exists an instance *purchase* of *Purchase* such that  $product \in purchase[Purchase - Item - Product]$  and  $region \in purchase[Purchase - (bill, Address) - Region]$ . Informally, it connects to each product a list of regions from where the product has been purchased. The latter specifies that the PSM association going from  $Region_{psm}$  to  $Purchase_{psm}$  connects to each region the list of purchases from the region that purchase the superior product.

We unify the proposed mechanisms for specifying semantics of PSM associations (i.e. PIM associations, PIM paths and nesting joins). We comprehend a PIM association as a PIM path of length 1. Further, we comprehend a PIM path *P* going from a PIM class *C* to *C'* as a nesting join

$$C'[rev(P) \rightarrow C']$$

Both are equivalent since *P* nests instances of *C'* in instances of *C*. In other words, it groups instances of *C'* and nests the groups to corresponding instances of *C*. This grouping is described by the nesting join.

**Example 8** Assume the PSM diagram in Figure 4. The PSM association going from  $Purchase_{psm}$  to  $Address_{psm}$  with an element label *ship-addr* represents a nesting join

$$Address[(ship, Purchase) \rightarrow]$$

Formally, it nests an instance *a* of *Address* in an instance *p* of *Purchase* if there exists an instance *a'* of *Address* such that  $p \in a'[Address - (ship, Purchase)]$  and  $a \in a'[Address] = \{a'\}$ , i.e.  $a = a'$ . Informally, it nests in each purchase its ship address. The other PSM associations represent the following nesting joins respectively:  $Address[(bill, Purchase) \rightarrow]$ ,  $Messenger[Purchase \rightarrow]$ ,  $Van[Purchase \rightarrow]$ ,  $Item[Purchase \rightarrow]$  and  $Product[Item \rightarrow]$ .

#### 4 XML Schema Reverse Engineering

The conceptual model proposed in the previous section can be used for modeling XML schemas as follows. We first design a PIM diagram and model each XML schema as a PSM diagram derived from the PIM diagram. The PSM diagram can then be mechanically translated to an XML Schema representation. In this paper, we are interested in the reversed process that starts with one or more XML schemas. We suppose that a conceptual PIM diagram already exists and we need to construct PSM diagrams that model the XML schemas in terms of the PIM diagram. Since doing this manually would be time-consuming and error-prone task, we show how to semi-automate this process. We suppose XML Schema as a language for syntactical description of XML schemas.

Formally, the problem is given as follows. We have an XML schema  $S_{xml}$  and a PIM diagram  $S_{pim}$ . We need to construct a PSM diagram  $S_{psm}$  that models the same XML format as  $S_{xml}$  and is derived from  $S_{pim}$ . In other words, PSM classes from  $S_{psm}$  must represent PIM classes from  $S_{pim}$  and PSM associations from  $S_{psm}$  must represent nesting joins specified over components of  $S_{pim}$ . We separate the process to two steps. In a first step a first approximation of the target  $S_{psm}$  is mechanically derived from the XML schema. We call the result of the first step *initial PSM diagram*. In a second step the first approximation is refined by mapping components of  $S_{psm}$  to components of  $S_{pim}$ . We describe both steps in detail in the following subsections.

There can be situations that go beyond the scope of the paper. First, we suppose that a given PIM diagram and XML schemas model the same data. If not, it can be impossible to fully map an XML schema to the PIM diagram since a required attribute, class or association can be missing. This requires a refinement of the PIM diagram which is not considered in this paper. Second, we suppose only basic constructions for mapping, i.e. mapping a PSM attribute/class to an equivalent PIM attribute/class and mapping a PSM association to an equivalent nesting join. However, there can be more complex situations that require, e.g. to map a concatenation of more PSM attributes to one PIM attribute. This situations are not therefore covered by this paper. On the other hand, it is only a technical problem to extend the proposed solution with such mapping constructs.

Our solution can not automatically provide the right solution of the mapping problem. We only look for a good approximation. It means that we estimate a mapping of a given component of an XML schema to components of the PIM diagram. However, the final decision about the mapping is left to a domain expert.

##### 4.1 Initial PSM Diagram Construction

The translation of  $S_{xml}$  to an initial PSM diagram starts with global element declarations in  $S_{xml}$ . Only those having assigned a complex type are considered. The transla-



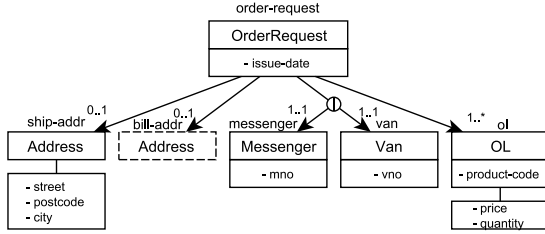


Figure 7: Initial PIM Diagram

tion continues recursively to declarations of their child elements. To simplify the algorithm for the purposes of this paper, we suppose that all complex types are defined globally in the XML schema (locally defined complex types can be transformed to global declarations by assigning auxiliary names). Moreover, we do not work with various simple types that can be defined with XML Schema constructs. We consider all of them as they were the basic XML Schema simple type `string`.

Let  $E$  be an element declaration with a name  $l$  and complex type  $T$ . We need to translate  $E$  and  $T$ . Since, there can be more element declarations sharing  $T$ , it is possible that  $T$  has already been translated during the translation of another element declaration. Therefore, the translation of  $E$  depends on whether  $T$  has been translated or not. Formally,  $E$  is translated as follows:

- (E1) If  $T$  has not been translated yet,  $E$  is translated to a PSM class  $C_{psm}$  with an element label  $l$ . The name of  $C_{psm}$  is given by the name of  $T$  (because it is defined globally, it must have a name). Moreover,  $C_{psm}$  is set as so called *base class* of  $T$ .  $T$  is translated as we describe in a while ((T1–3) below).
- (E2) If  $T$  has already been translated during the translation of another element declaration  $E'$ , it has a base class  $C'_{psm}$ .  $C'_{psm}$  is the result of the translation of  $E'$  according to (E2). In that case  $E$  is translated to a structural representative of  $C'_{psm}$ . The structural representative has an element label  $l$ .

If  $T$  has not been translated yet, we need to translate its attribute declarations and content model. A declaration of an attribute  $A$  with a name  $l$  is translated to a PSM attribute of  $C_{psm}$  with a name  $l$ . The content model of  $T$  can be defined by various XML Schema constructs. As we mentioned in Section 2, we consider only sequence. A sequence can contain element declarations and choice constructs. A choice construct can contain element declarations. The components of the content model of  $T$  are translated as follows:

- (T1) Element declaration  $E'$  with a name  $l$  and simple type  $T'$  is translated to a PSM attribute with a name  $l$ . The cardinality of the new attribute is set according to `minOccurs` and `maxOccurs` of  $E'$ . The attribute is placed into an attribute container assigned to  $C_{psm}$ . If there are more sibling element declarations with a simple type, the resulting attributes are coupled into one attribute container.
- (T2) Element declaration  $E'$  with a complex type  $T'$  is translated to a PSM class or structural representative  $C'_{psm}$  according to (E1–2). A PSM association  $A_{psm}$  going from  $C_{psm}$  to  $C'_{psm}$  is created. The values of `minOccurs` and `maxOccurs` of  $E'$  are used as the minimal and maximal cardinality of  $C'_{psm}$  in  $A_{psm}$ .
- (T3) choice is translated to a content choice assigned to  $C_{psm}$ . The element declarations in the choice are translated recursively according to (T1–2) but assigned to the content choice instead of  $C_{psm}$ .

**Example 9** Assume the XML schema depicted in Figure 2. It is translated to an initial PSM diagram depicted in Figure 7. There is one global element declaration `order-request` with a complex type `OrderRequest`. Because `OrderRequest` has not been translated yet, `order-request` is translated according to (E1) to a PSM class `OrderRequestpsm` with an element label `order-request`.

Further, `OrderRequest` is translated. The attribute declaration `issue-date` is translated to a PSM attribute `issue-date` of `OrderRequestpsm`. The content model of `OrderRequest` is translated as follows.

The element declaration `ship-addr` has a complex type `Address` and (T2) is applied. `ship-addr` is translated according to (E1) because `Address` has not been translated yet. The result is a PSM class `Addresspsm` with an element label `ship-addr`. A PSM association going from `OrderRequestpsm` to `Addresspsm` is created. The cardinality constraint of `Addresspsm` in the PSM association is 0..1. Within the scope of the translation of `ship-addr`, `Address` is translated. It has no attributes and its content model contains element declarations `street`, `postcode` and `city` with simple types. They are translated according to (T1) to an attribute container assigned to `Addresspsm` with PSM attributes `street`, `postcode` and `city`, respectively.

The element declaration `bill-addr` has a complex type `Address` and (T2) is applied. Because `Address` has already been translated, `bill-addr` is translated according to (E2) to a structural representative of the base class of `Address` which is `Addresspsm`. The structural representative has an element label `bill-addr`. A PSM association going from `OrderRequestpsm` to the structural representative is created.

The choice is translated according to (T3) to a content choice in `OrderRequestpsm`. The element declarations `messenger` and `van` are translated according to (T2) to PSM classes `Messengerpsm` and `Vanpsm` with element labels `messenger` and `van`, respectively.

The element declaration `ol` has a complex type `OL` and (T2) is applied. (E1) is further applied because the complex type has not been translated yet. A PSM class `OLpsm` is created with an element label `ol`. The declaration of the attribute `product-code` is translated to a PSM attribute `product-code`. The declarations of the elements `price` and `quantity` are translated according to (T1) to PSM attributes `price` and `quantity` in an attribute container assigned to `OLpsm`.

## 4.2 PSM Diagram Semantics Refinement

An initial PSM diagram captures structure of  $\mathcal{S}_{xml}$ . However, we also need to describe semantics of  $\mathcal{S}_{xml}$  in terms of the PIM diagram  $\mathcal{S}_{pim}$ . It means to map components of  $\mathcal{S}_{xml}$  to components of  $\mathcal{S}_{pim}$ . A naïve solution is to let a domain expert to map the components manually. However, this is an error-prone and time-consuming task.

In this section, we propose an algorithm for semi-automatic mapping of  $\mathcal{S}_{xml}$  to  $\mathcal{S}_{pim}$ . It is semi-automatic since it just provides with mapping suggestions but still requires a participation of a domain expert. In the first two subsections we describe complementary algorithms for measuring similarity of strings and PIM paths weighting. In the third subsection, we describe the mapping algorithm in detail.

### 4.2.1 String Similarity

We will need to compute the similarity between two strings  $s_1$  and  $s_2$ . We could utilize various widely known algorithms for measuring syntactical and semantical similarity (see (Shvaiko & Euzenat 2005) for their survey). For simplicity, we utilize only the longest common substring of  $s_1$  and  $s_2$  since advanced algorithms for mea-

```

01 weightPaths(PIMClass  $C$ , PIMClass  $C'$ , String[]  $S$ )
02 int[] result;
03 for each PIM path  $P$  going from  $C$  to  $C'$ 
04   result[ $P$ ] :=  $w(P, S)$ 
05 return result

```

Figure 8: PIM Path Weighting Algorithm

suring string similarity are not in our main interest in this paper. The similarity between  $s_1$  and  $s_2$  is computed as

$$\text{sim}(s_1, s_2) = \frac{l(s_1, s_2)}{\max\{l(s_1), l(s_2)\}}$$

where  $l(s_1, s_2)$  denotes the length of the longest common substring of  $s_1$  and  $s_2$  and  $l(s)$  denotes the length of  $s$ .

We will also need to measure the similarity between two sets of strings  $S_1$  and  $S_2$ . It is computed as a sum of pairwise similarities of strings from  $S_1$  with strings from  $S_2$  normalized by the number of pairs:

$$\text{ssim}(S_1, S_2) = \frac{\sum_{s_1 \in S_1, s_2 \in S_2} \text{sim}(s_1, s_2)}{|S_1||S_2|}$$

#### 4.2.2 PIM Paths Weighting

We will also utilize an auxiliary algorithm **weightPaths** that weights direct PIM paths going from a PIM class  $C$  to a PIM class  $C'$ . The algorithm is depicted in Figure 8. It has  $C$  and  $C'$  as parameters. The third parameter  $S$  is a set of strings that influences the weight of the PIM paths. A weight of a given PIM path  $P$  is a number from the interval  $(0, 1)$  (including 0 and 1). It decreases with the growing length of  $P$  and increases with the similarity of the labels of the PIM associations and names of the PIM classes along  $P$  with the strings from  $S$ .

Formally, the weight of a given direct PIM path  $P = C_1 - \dots - C_n$  is computed as follows:

$$w(P, S) = \left( \sum_{i=1}^{n-1} \frac{1 + \text{ssim}(\{a_i, c_{i+1}\}, S)}{i} \right) * \frac{1}{2n}$$

where for each  $i \in [1, n]$ ,  $c_i$  is the name of the PIM class  $C_i$  and for each  $i \in [1, n - 1]$ ,  $a_i$  is the label of the PIM association connecting  $C_i$  and  $C_{i+1}$  in  $P$ .

#### 4.2.3 Semi-automatic Mapping Algorithm

In this section we propose a semi-automatic algorithm **classMap** that maps components of an initial PSM diagram  $\mathcal{S}_{xml}$  to components of a PIM diagram  $\mathcal{S}_{pim}$ . The algorithm is depicted in Figure 11. We start by applying **classMap** on the root PSM class  $\mathcal{S}_{xml}$ . It maps the root to a corresponding PIM class and follows recursively to the descendants. According to the classification proposed in (Shvaiko & Euzénat 2005), the proposed algorithm belongs to the class of structural schema-based mapping techniques that measure similarity of the schema components on the base of children in a combination with string based techniques.

For an actual PSM class  $C_{psm}$  from  $\mathcal{S}_{xml}$ , **classMap** proceeds in the following steps:

- *Class Mapping Estimation* computes a similarity of  $C_{psm}$  with each PIM class  $C$ . The similarity is a combination of a similarity of names and attributes of both classes as well as a similarity of children of  $C_{psm}$  with neighbors of  $C$ . Therefore, it does not use only basic syntactical similarity but also structural similarity of the neighborhood of  $C_{psm}$  with the neighborhood of  $C$ .

```

01 attrSim(PIMClass  $C$ , PSMAttribute  $Attr_{psm}$ )
02 int[][] result;
03 for each PIM attribute  $Attr$ 
04   PIMClass  $C' := Attr.class$ ;
05   int  $sim := \text{sim}(Attr_{psm}.name, Attr.name)$ ;
06   if  $C = C'$ 
07     result[ $Attr$ ][.] :=  $sim$ ;
08   else
09     string  $labels[] := \{Attr_{psm}.name\}$ ;
10     result[ $Attr$ ] := weightPaths( $C, C', labels$ );
11   for each direct PIM path  $P$  going from  $C$  to  $C'$ 
12     result[ $Attr$ ][ $P$ ] := result[ $Attr$ ][ $P$ ] *  $sim$ ;
13 return result;

```

Figure 9: Attribute Similarity Algorithm

- *Class Mapping Specification* is performed by the domain expert who selects a PIM class for mapping of  $C_{psm}$  from the list of PIM classes ordered by their similarity with  $C_{psm}$  computed in the previous step.
- *Association Mapping* performs mapping of a PSM association going to  $C_{psm}$ , if there is any. Since a parent  $C'_{psm}$  of the PSM association as well as its child  $C_{psm}$  are mapped to PIM classes  $C'$  and  $C$ , respectively, the algorithm offers the list of PIM paths connecting  $C'$  and  $C$  ordered by their weights (see **weightPaths** algorithm depicted in Figure 4.2.2). The domain expert selects the right PIM path from a list of the PIM paths ordered by their weights. From the selected PIM path, an equivalent nesting join is constructed for mapping of the PSM association.
- *Subtree Mapping* performs mapping of attributes of  $C_{psm}$  and recursive mapping of the subtree of  $C_{psm}$ .

**Example 10** Figure 4 shows the resulting PSM diagram after applying **classMap** on  $OrderRequest_{psm}$  from the initial PSM diagram depicted in Figure 7.

If  $C_{psm}$  is a structural representative of  $C'_{psm}$ , it must represent the same PIM class as  $C'_{psm}$  (lines 02–05,  $C_{psm}.pim$  denotes the PIM class represented by  $C_{psm}$ ). Otherwise, the four steps are performed. In the rest of this sections, we describe each step in detail.

(1) **Class Mapping Estimation**. The first part of the **classMap** algorithm (lines 06–15) estimates mapping of  $C_{psm}$ . It measures a similarity of  $C_{psm}$  with each PIM class  $C$ . First, it computes a string similarity of a name of  $C_{psm}$  with a name of  $C$  and similarity of an element label of  $C_{psm}$  with the name of  $C$ . The maximum of the two values is stored to *initSim* (line 08).

Next, **classMap** estimates mapping of the PSM attributes in  $C_{psm}$  and in attribute containers assigned to  $C_{psm}$  (lines 09–11). It assumes that  $C_{psm}$  is mapped to  $C$ . The estimation itself is computed for each PSM attribute  $Attr_{psm}$  of  $C_{psm}$  at line 11 by calling **attrSim** depicted in Figure 9. **attrSim** takes  $C$  and  $Attr_{psm}$  as parameters and computes a 2-dimensional matrix called *attribute similarity matrix*. The matrix is computed as follows.  $Attr_{psm}$  can be mapped to any PIM attribute  $Attr$ .  $Attr$  can be an attribute of  $C$  or an attribute of another PIM class  $C'$ . In the former case, the similarity of  $Attr_{psm}$  with  $Attr$  is computed as a similarity of their names (line 07). In the latter case, the similarity is moreover influenced by direct PIM paths connecting  $C$  and  $C'$ . This corresponds to a natural intuition.  $Attr_{psm}$  is a PSM attribute of  $C_{psm}$ . We consider that  $C_{psm}$  is mapped to  $C$ . If  $Attr_{psm}$  is mapped to  $Attr$  of  $C'$ ,  $C_{psm}$  represents a join of  $C$  and  $C'$ . Therefore, there must be a direct PIM path connecting  $C$  and  $C'$  otherwise the join can not be performed. Because there can be more direct PIM paths connecting  $C$  and  $C'$ , we assign a weight to each of them by calling **weightPaths** (line 10) with parameters  $C, C'$  and  $\{A_{psm}.name\}$ . The weight of a given

```

01 childSim(PIMClass  $C$ , PSMClass  $C'_{psm}$ )
02   int[][] result;
03   for each PIM class  $C'$ 
04     int  $sim := \max(sim(C_{psm}.name, C'.name),$ 
05                    $sim(C'_{psm}.label, C'.label));$ 
06     string  $labels[] := \{C'_{psm}.name, C'_{psm}.label\};$ 
07      $result[C'] := \mathbf{weightPaths}(C, C', labels);$ 
08     for each direct PIM path  $P$  going from  $C$  to  $C'$ 
09        $result[C'][P] := result[C'][P] * sim;$ 
10   return result;

```

Figure 10: Child Similarity Algorithm

direct PIM path  $P$  increases with a decreasing length of  $P$  and with a growing similarity of names and labels along  $P$  with the name of  $Attr_{psm}$ . The resulting similarity of  $Attr_{psm}$  with  $Attr$  for a given PIM path  $P$  is the weight of  $P$  multiplied by the string similarity of the names of  $Attr_{psm}$  and  $Attr$  (line 12). **attrSim** returns the similarity of  $Attr_{psm}$  with each PIM attribute  $Attr$  for each direct PIM path connecting  $C$  and  $C'$  where  $C'$  is the PIM class of  $Attr$ . Note that  $.$  at line 07 denotes an empty PIM path and is added to suit the structure of the result.

The algorithm **classMap** does not consider the whole attribute similarity matrix for  $C$  and  $A_{psm}$  to estimate the mapping of  $C_{psm}$ . It uses only the maximal value in the matrix (line 11) which is added to the variable  $attrSim$ . The whole matrix is used later to suggest mapping of attributes to the domain expert.

Finally, **classMap** estimates mapping of children of  $C_{psm}$  (lines 12–14). It considers that  $C_{psm}$  is mapped to  $C$ . The estimation of a mapping of a given child  $C'_{psm}$  is computed by **childSim** depicted in Figure 10. It has  $C$  and  $C'_{psm}$  as parameters and returns a 2-dimensional matrix called *child similarity matrix*.  $C'_{psm}$  can be mapped to any PIM class. For an actual PIM class  $C'$ , the similarity of  $C'_{psm}$  and  $C'$  is measured as follows. First, the similarity of the name and element label of  $C'_{psm}$  with the name of  $C'$  is computed (line 04) and stored to  $sim$ . Second, PIM paths connecting  $C$  and  $C'$  are weighted by **weightPaths** with parameters  $C$ ,  $C'$  and  $\{C'_{psm}.name, C'_{psm}.label\}$  (line 06). This corresponds to a natural intuition.  $C'_{psm}$  is a child of  $C_{psm}$ .  $C_{psm}$  is mapped to  $C$  (consideration). Therefore,  $C'_{psm}$  can be mapped to  $C'$  if there is a PIM path connecting  $C$  and  $C'$ . There can be more such PIM paths. The weight of a given PIM path  $P$  increases with a decreasing length of  $P$  and with a growing similarity of names and labels along  $P$  with the name and element label of  $C'_{psm}$ . Finally, we multiply the weight of a each PIM path connecting  $C$  and  $C'$  by  $sim$  (lines 07–08). **childSim** returns the similarity of  $C'_{psm}$  with each PIM class  $C'$  for each direct PIM path connecting  $C$  and  $C'$ .

The result of **childSim** is utilized by **classMap** at line 14. Only its maximum is considered and is added to the variable  $childSim$ . The whole matrix is used later for mapping PSM associations.

The estimated similarity of  $C_{psm}$  with  $C$  is computed as an avg of  $initSim$ ,  $attrSim$  and  $childSim$  (line 15).

**Example 11** Assume **classMap** applied on *OrderRequest<sub>psm</sub>*. The first part of the algorithm estimates mapping of *OrderRequest<sub>psm</sub>* by computing its similarity with each PIM class. We show how the similarity of *OrderRequest<sub>psm</sub>* with *Purchase* is computed.

Similarity of the name, resp. element label, of *OrderRequest<sub>psm</sub>* with the name of *Purchase* is computed first and the maximum of both is taken. The result is 0.08 since their common substring has length 1.

Next, the algorithm estimates mapping of the attributes of *OrderRequest<sub>psm</sub>*. For each of the attributes, the at-

```

01 classMap(PSMClass  $C_{psm}$ )
02   if  $C_{psm}$  is structural representative of  $C'_{psm}$ 
03      $C_{psm}.pim := C'_{psm}.pim;$ 
04      $C_{psm}.name := C'_{psm}.name;$ 
05     return;
06   int[] estimatedSim;
07   for each PIM class  $C$ 
08     int  $initSim := \max(sim(C_{psm}.name, C.name),$ 
09                        $sim(C_{psm}.label, C.name));$ 
09     int  $attrSim := 0;$ 
10     for each  $Attr_{psm} \in C_{psm}.attrs$ 
11        $attrSim := attrSim + \max(\mathbf{attrSim}(C, Attr_{psm}));$ 
12     int  $childSim := 0;$ 
13     for each  $C'_{psm} \in C_{psm}.childClasses$ 
14        $childSim := childSim + \max(\mathbf{childSim}(C, C'_{psm}));$ 
15     int  $estimatedSim[C] :=$ 
16        $\frac{initSim + attrSim + childSim}{1 + size(C_{psm}.attrs) + size(C_{psm}.children)};$ 
17   show PIM classes ordered by estimatedSim in
18     descending order;
19   user selects a candidate  $C$  for mapping of  $C_{psm}$ ;
20    $C_{psm}.pim := C;$ 
21   if  $C_{psm}$  is not a root
22      $A_{psm} :=$  PSM association going to  $C_{psm}$ ;
23      $C_{psm}^{par} := C_{psm}.parentClass;$ 
24      $C_{psm}^{par} := C_{psm}^{par}.pim;$ 
25     string  $labels[] := \{C_{psm}.name, C_{psm}.label\};$ 
26      $weights := \mathbf{weightPaths}(C_{psm}^{par}, C, labels);$ 
27     show PIM paths going from  $C_{psm}^{par}$  to  $C$  ordered by
28        $weights$  in descending order
29     user selects a PIM path  $P$  from the list
30       for mapping of  $A_{psm}$ ;
31      $A_{psm}.pim := C[rev(P) \rightarrow C];$ 
32   for each  $C'_{psm} \in C_{psm}.childClasses$ 
33     classMap( $C'_{psm}$ );
34   for each  $Attr_{psm} \in C_{psm}.attrs$ 
35     attrMap( $Attr_{psm}$ );

```

Figure 11: Class Mapping Algorithm

tribute similarity matrix is computed by **attrSim** with a consideration that *OrderRequest<sub>psm</sub>* is mapped to *Purchase*. The matrix contains a field for each PIM attribute *Attr* and each direct PIM path going from *Purchase* to the PIM class of *Attr*.

Assume the matrix for the PSM attribute *issue-date<sub>psm</sub>*. We show the computation of the similarity of *issue-date<sub>psm</sub>* with the following three PIM attributes:

- *date of Purchase*:  $sim(issue-date, date) = 0.40$  is computed and line 07 is applied.
- *completion-date of ProductSet*:  $sim(issue-date, completion-date) = 0.33$  is computed and lines 09–12 are applied. **weightPaths** with parameters *Purchase*, *ProductSet* and string *issue-date* is called. It finds each direct PIM path going from *Purchase* to *ProductSet* and computes its weight. There are several PIM paths. For example, the weight of *Purchase* – (*ship*, *Address*) – (*ship*, *Supply*) – *ProductSet* is 0.34. The resulting similarity of *issue-date<sub>psm</sub>* with *completion-date* for this PIM path is therefore  $0.33 * 0.34 = 0.11$ . All other PIM paths have a lower weight and are not therefore considered for the estimation.
- *supply-date of Supply*: Analogously, we get 0.21.

The similarity of *issue-date<sub>psm</sub>* with other PIM attributes is insignificant. We return back to **classMap**. The algorithm takes only the maximal value 0.40 from the matrix, i.e. the mapping of *issue-date<sub>psm</sub>* to the PIM attribute *date of Purchase* is considered. The variable  $attrSim$  summarizing the maximal similarities of the attributes of *OrderRequest<sub>psm</sub>* with PIM attributes is therefore increased by 0.40.

	issue-date	ship-addr	bill-addr	messenger	van	ol	init	est
Purchase	0.40 da	0.74 (sh,Ad)	0.74 (bi,Ad)	0.75 Me	0.75 Va	0.07 (bi,Ad)-(bi,Su)	0.08	0.50
Supply	0.46 su-da	0.74 (sh,Ad)	0.74 (bi,Ad)	0.38 (sh,Ad)-(sh,Pu)-Me	0.35 (sh,Ad)-(sh,Pu)-Va	0.07 (sh,Ad)-Re	0.08	0.40
ProductSet	0.33 co-da	0.45 Su-(sh,Ad)	0.45 Su-(bi,Ad)	0.30 Su-(sh,Ad)-(sh,Pu)-Me	0.29 Su-(sh,Ad)-(sh,Pu)-Va	0.08 Pr	0.08	0.28

Table 1: Evaluation of  $OrderRequest_{psm}$  Mapping Estimation

After the estimation of mapping of the attributes of  $OrderRequest_{psm}$ , the algorithm **classMap** estimates mapping of the children of  $OrderRequest_{psm}$ . It computes for each child  $C'_{psm}$  of  $OrderRequest_{psm}$  the child similarity matrix by calling **childSim** (line 14) with a consideration that  $OrderRequest_{psm}$  is mapped to Purchase. The matrix contains a field for each PIM class  $C'$  and direct PIM path going from Purchase to  $C'$ .

Assume the computation of the child similarity matrix for the child  $Address_{psm}$  with the element label ship-addr. **childSim** computes the similarity of  $Address_{psm}$  with each PIM class  $C'$  for each PIM path going from Purchase to  $C'$ . An interesting PIM class is Address. **childSim** computes the string similarity of the names of  $Address_{psm}$  and Address which is 1 (line 04). The similarity of the element label of  $Address_{psm}$  with the name of Address is lower. Further, PIM paths going from Purchase to Address are weighted by **weightPaths** with parameters Purchase, Address and strings 'address' and 'ship-addr'. There are two such PIM paths: Purchase – (ship, Address) and Purchase – (bill, Address) with weights 0.74 and 0.68, respectively. The string 'ship-addr' influences the weight of the former because there is a label ship along the path which has non-zero similarity with 'ship-addr'. The weights are then multiplied by  $sim = 1$ .

The similarity of  $Address_{psm}$  with other PIM classes is insignificant. After the matrix for  $Address_{psm}$  is computed, we return back to **classMap** where we take only the maximal value from the matrix, i.e. 0.74.

Finally, the estimated similarity of  $OrderRequest_{psm}$  with Purchase is computed. The result is depicted in Table 1 in the last column (see Example 12 for details).

**Example 12** Table 1 shows the estimated similarity of  $OrderRequest_{psm}$  with PIM classes Purchase, Supply and ProductSet in the last column est. These PIM classes have the highest similarity with  $OrderRequest_{psm}$ . For each PIM class, we show the maximal value from the attribute similarity matrix for the attribute issue-date<sub>psm</sub>. We also show the corresponding PIM attribute for which the similarity was computed<sup>2</sup>. For example, the column (Supply, issue-date) shows the maximum from the attribute similarity matrix for issue-date<sub>psm</sub> and Supply. It was computed for the PIM attribute supply-date of Supply. We further show the maximal value from the child similarity matrix for each child of  $OrderRequest_{psm}$ . We show the corresponding PIM path for each value. For example, the cell (Supply, messenger) shows the maximal value from the child similarity matrix for the child Messenger<sub>psm</sub> and PIM class Supply. It also shows the PIM path for which the value was computed, i.e. Supply – (ship, Address) – (ship, Purchase) – Messenger.

Table 2 shows the estimated similarity of the PSM class  $OL_{psm}$  with PIM classes Item and Address. The similarity with other PIM classes is insignificant. It shows that we can estimate the similarity even though the name and element label of the PSM class have nothing in common with the names of the PIM classes.

**(2) Class Mapping Specification.** The second part of **classMap** (lines 16–18) performs the mapping of  $C_{psm}$

	product-code	price	quantity	init	est
Item	0.64 Pr.pr-cd	0.5 un-pr	0.25 am	0.00	0.35
Address	0.33 postcode	0.30 Su.un-pr	0.14	0.00	0.19

Table 2: Evaluation of  $OL_{psm}$  Mapping Estimation

with a participation of the expert. It shows the list of PIM classes ordered by their estimated similarity with  $C_{psm}$  (line 16) in descending order. The expert selects a PIM class  $C'$  (line 17) and  $C_{psm}$  is mapped to  $C'$  (line 18). This completes the mapping of  $C_{psm}$ .

**Example 13** After the estimation of mapping of  $OrderRequest_{psm}$  in Example 11, we show the list of all PIM classes ordered by their estimated similarity with  $OrderRequest_{psm}$ . The first three PIM classes are shown in Table 1. The expert selects Purchase.  $OrderRequest_{psm}$  is therefore mapped to Purchase.

**(3) Association Mapping.** The third part of the **classMap** algorithm (lines 19–27) performs the mapping of the PSM association  $A_{psm}$  going to  $C_{psm}$  if  $C_{psm}$  is not a root. Let  $A_{psm}$  go from  $C'_{psm}$ . We need to find a nesting join that describes the semantics of  $A_{psm}$ . We already have that  $C'_{psm}$  is mapped to a PIM class  $C'$  and  $C_{psm}$  to  $C$ . We need a direct PIM path going from  $C'$  to  $C$  as the base of the nesting join. The right PIM path must be selected by the domain expert. The algorithm only suggests suitable possibilities by weighting the PIM paths (line 24). Afterwards, the expert selects a PIM path  $P$  from the list of PIM paths ordered by their weights (lines 25–26) and  $A_{psm}$  is mapped to a nesting join  $C[rev(P) \rightarrow C]$  (line 27). Furthermore, it can be necessary to add a context to the nesting join. We describe this possibility later in Section 4.2.4.

**Example 14** We have  $OrderRequest_{psm}$  mapped to Purchase. Assume that we have its child  $Address_{psm}$  with the element label ship-addr mapped to Address. We need to map the PSM association going from  $OrderRequest_{psm}$  to  $Address_{psm}$ . The algorithm weights direct PIM paths going from Purchase to Address by calling **weightPaths** with parameters Purchase, Address and strings 'address' and 'ship-addr'. The PIM paths were already weighted during the estimation of mapping of  $OrderRequest_{psm}$  in Example 11. We can therefore utilize the results. The algorithm shows the PIM paths ordered by their weight in descending order. The expert selects the path Purchase – (ship, Address) and the PSM association is consequently mapped to  $Address[(ship, Purchase) \rightarrow ]$ . If the associations connecting Purchase and Address were distinguished by labels with the same similarity with 'address' and 'ship-addr', the PIM paths would have the same weight and we could not suggest the right mapping.

**(4) Subtree Mapping.** Finally, **classMap** maps the PSM attributes of  $C_{psm}$  to PIM attributes and child PSM classes of  $C_{psm}$  to PIM classes (lines 28–31). Each child is mapped recursively by **classMap** (line 29). Each attribute is mapped by an algorithm **attrMap** (line 31) depicted in Figure 12. It maps a PSM attribute  $Attr_{psm}$  in  $C_{psm}$  or in an attribute container assigned to  $C_{psm}$ .  $C_{psm}$

<sup>2</sup>PIM paths and attributes in the table are abbreviated – for each step and attribute only the first two characters are shown

```

01 attrMap (PSMAttribute  $Attr_{psm}$ )
02  $C := Attr_{psm}.class.pim$ ;
03  $sim := attrSim(C, Attr_{psm})$ ;
04 show an ordered list of PIM attributes,
   the order of  $A$  is given by the maximal
   value in  $sim[A]$  (descending order);
05 user selects a PIM attribute  $Attr$  to be
   mapped to  $Attr_{psm}$ ;
06  $Attr_{psm}.alias := Attr_{psm}.name$ ;
07  $Attr_{psm}.pim := Attr$ ;
08  $C' := Attr.class$ ;
09 if  $C \neq C'$ 
10   PSM class  $C'_{psm}$  is created;
11    $C'_{psm}.psm := C'$ ;  $C'_{psm}.name := C'.name$ ;
12   move  $Attr_{psm}$  to  $C'_{psm}$ ;
13   PSM association  $A_{psm}$  is created;
14    $A_{psm}.parent := A_{psm}.class$ ;
15    $A_{psm}.child := C'_{psm}$ ;
16   show PIM paths going from  $C$  to  $C'$ 
   ordered by  $sim[Attr]$  in descending order;
17   user selects a PIM path  $P$  for mapping to  $A_{psm}$ ;
18    $A_{psm}.pim := C'[rev(P) \rightarrow C]$ ;

```

Figure 12: Attribute Map Algorithm

was mapped with  $C$  in the previous part. **attrMap** starts with computing the attribute similarity matrix for  $Attr_{psm}$  and  $C$  (line 03). The matrix was computed during the estimation of mapping  $C_{psm}$  and can be reused. The algorithm shows an ordered list of PIM attributes (line 04) in the descending order. The order of a PIM attribute  $Attr$  is given by the maximal value in  $sim[Attr]$ . The expert selects a PIM attribute  $Attr$  from the list and  $Attr$  is mapped with  $Attr_{psm}$ .

If  $Attr$  is from  $C$ , we are done. If  $Attr$  is not from  $C$  but another PIM class  $C'$ ,  $Attr_{psm}$  can not stay with its PSM class  $C_{psm}$  because a PSM class representing  $C$  can have only attributes of  $C$ . We therefore create a new PSM class  $C'_{psm}$  representing  $C'$  (lines 10–11) and  $Attr_{psm}$  is moved to  $C'_{psm}$  (line 12).  $C'_{psm}$  must be added as a child of  $C_{psm}$ . Therefore, a PSM association  $A_{psm}$  is created with  $C_{psm}$  as parent and  $C'_{psm}$  as child.  $A_{psm}$  must represent an appropriate nesting join. Therefore, we need to determine an appropriate PIM path  $P$  going from  $C$  to  $C'$ .  $P$  is selected by the domain expert from the list of PIM paths ordered by their weight in  $sim[Attr]$  (lines 16–17). Finally,  $A_{psm}$  is mapped to the nesting join  $C'[rev(P) \rightarrow]$ .

**Example 15** Assume that the PSM class  $OL_{psm}$  from the initial PSM diagram was mapped to the PIM class *Item* and we are mapping its PSM attribute *product-code*<sub>psm</sub>. **attrMap** gets the attribute matrix similarity for *product-code*<sub>psm</sub> and *Item*. It was computed during the estimation of mapping of  $OL_{psm}$ . The algorithm shows the list of PIM attributes ordered by their maximal estimated similarity with *product-code*<sub>psm</sub>. The PIM attribute *product-code* of the PIM class *Product* is the most similar PIM attribute to *product-code*<sub>psm</sub> as shown in Table 2 in the cell (*Item*,*product-code*). The expert selects *product-code* to be mapped to *product-code*<sub>psm</sub>. Because *product-code* is from *Product* and not *Item*, a child PSM class *Product*<sub>psm</sub> mapped to *Product* is created and *product-code*<sub>psm</sub> is moved here. Moreover, a PSM association going from  $OL_{psm}$  to *Product*<sub>psm</sub> is created. **attrMap** displays the list of PIM paths going from *Item* to *Product* ordered by their weights. The user selects *Item* – *Product* and the new PSM association is mapped to *Product*[*Item* →].

#### 4.2.4 Setting Class Context

Assume a PSM association  $A_{psm}$  whose parent  $C'_{psm}$  represents a PIM class  $C'$  and child  $C_{psm}$  represents  $C$ . The

algorithm **classMap** automatically maps  $A_{psm}$  to a nesting join  $C[P \rightarrow C]$  where  $P$  is a PIM path going from  $C$  to  $C'$ . The nesting join specifies that an instance  $c$  of  $C$  is nested in an instance  $c'$  of  $C'$  if  $c' \in c[P]$ , i.e.  $C$  instances are joined with  $C'$  instances and grouped by  $C'$ . However, this semantics can be wrong because it can be necessary to add a context to the nesting join.

Since it is very hard to determine the context automatically, we need a domain expert to decide. It should be as easy as possible for the domain expert. We propose the following procedure. After the semiautomatic mapping, the expert selects a non-root PSM class  $C_{psm}$  representing a PIM class  $C$ . There is a PSM association  $A_{psm}$  going to  $C_{psm}$ .  $A_{psm}$  represents a nesting join  $C[P \rightarrow C]$ . The expert can add PIM classes represented by one or more ancestors of the parent of  $C_{psm}$  to the context of the nesting join. To satisfy the conditions (J1) and (J2) introduced in Section 3.3, if an ancestor  $C''_{psm}$  is considered, each PSM class on the path from  $C''_{psm}$  to the parent of  $C_{psm}$  must be considered too. Therefore, it is enough when the domain expert specifies a number of the ancestors that should be considered. Let  $k$  denote this number. For each of the ancestors, we need to add to the context the right PIM path going from  $C$  to the PIM class represented by the ancestor. Moreover, we also need to change the nesting joins represented by the PSM associations connecting the ancestors because the conditions (J1) and (J2) introduced in Section 3.3 must be satisfied.

Formally, let  $C_{1,psm}, \dots, C_{k+2,psm}$  be PSM classes such that  $C_{k+2,psm} = C_{psm}$  and  $C_{i,psm}$  is the parent of  $C_{i+1,psm}$  for each  $i \in [1, k+1]$ . Let  $C_{i,psm}$  represent a PIM class  $C_i$  for each  $i \in [1, k+2]$  ( $C_{k+2} = C$ ). Let  $A_{1,psm}, \dots, A_{k+1,psm}$  denote PSM associations where  $A_{i,psm}$  goes from  $C_{i,psm}$  to  $C_{i+1,psm}$  for each  $i \in [1, k+1]$  ( $A_{k+1,psm} = A_{psm}$ ). Let  $A_{i,psm}$  represent a nesting join  $C_{i+1}[P_i \rightarrow C_{i+1}]$  where  $P_i$  is a PIM path going to  $C_i$  for each  $i \in [1, k+1]$ . Finally, we suppose that for each  $i \in [1, k]$  the following conditions are satisfied:

- (C1)  $P_i$  contains a step which is the PIM class  $C$ , i.e.  $P_i = P_{i,1} - C - P_{i,2}$  ( $P_{i,1}$  and  $P_{i,2}$  can be empty),
- (C2)  $P_{i+1,2} = rev(P_{i,1}) - C_{i+1}$  where  $rev(P)$  denotes  $P$  in the reversed direction (we put  $P_{k+1,2} = P_{k+1}$ ).

If (C1–2) are not satisfied, the context can not be set. Each PIM path  $C - P_{i,2}$  determines a PIM path that we need to put to the context. We therefore need to add the PIM paths  $P_{1,2}, \dots, P_{k,2}$  to the context of the nesting join represented by  $A_{psm}$  ( $= A_{k+1,psm}$ ). The nesting joins represented by  $A_{1,psm}, \dots, A_{k,psm}$  must be updated as well to satisfy the conditions (J1) and (J2).

For each  $i \in [1, k]$  we need  $A_{i,psm}$  to represent a nesting join  $C[P_{i,2} \rightarrow P_{i+1,2}]$  instead of  $C_{i+1}[P_{i,1} - C - P_{i,2} \rightarrow C_{i+1}]$ . We show that the semantics described by both nesting joins is the same. The former nests an instance  $c_{i+1}$  of  $C_{i+1}$  in an instance  $c_i$  of  $C_i$  if the following condition (S1) is satisfied:

$$(\exists c \in \llbracket C \rrbracket)(c_{i+1} \in c[C - P_{i+1,2}] \wedge c_i \in c[C - P_{i,2}])$$

i.e. if there is an instance  $c$  of  $C$  such that  $c_{i+1}$  is accessible from  $c$  by the PIM path  $C - P_{i+1,2}$  and  $c_i$  is accessible from  $c$  by  $C - P_{i,2}$ . The latter nests  $c_{i+1}$  in  $c_i$  if the following condition (S2) is satisfied:

$$c_i \in c_{i+1}[[C_{i+1} - P_{i,1} - C - P_{i,2}]]$$

i.e. if  $c_i$  is accessible from  $c_{i+1}$  by  $C_{i+1} - P_{i,1} - C - P_{i,2}$ .

We show that (S1) and (S2) are equivalent. Assume that (S2) is satisfied for  $c_{i+1}$  and  $c_i$ . It is equivalent to

$$(\exists c \in \llbracket C \rrbracket)(c \in c_{i+1}[[C_{i+1} - P_{i,1} - C]] \wedge c_i \in c[[C - P_{i,2}]])$$

i.e. (S2) is satisfied if and only if there is an instance  $c$  of  $C$  such that  $c$  is accessible from  $c_{i+1}$  by  $C_{i+1} - P_{i,1} - C$  and  $c_i$  from  $c$  by  $C - P_{i,2}$ . This is further equivalent to

$$(\exists c \in \llbracket C \rrbracket) (c_{i+1} \in c[C - rev(P_{i,1}) - C_{i+1}] \wedge c_i \in c[C - P_{i,2}])$$

because if  $c$  is accessible from  $c_{i+1}$  by  $C_{i+1} - P_{i,1} - C$  then  $c_{i+1}$  must be accessible from  $c$  by the reversed PIM path  $C - rev(P_{i,1}) - C_{i+1}$ . Because (C2) is satisfied, we can change  $rev(P_{i,1}) - C_{i+1}$  with  $P_{i+1,2}$  and we get (S1). Therefore, we have that (S1) is equivalent with (S2).

Now, we can set the required context and ensure that the conditions (J1) and (J2) from Section 3.3 are satisfied. We set  $A_{psm}$  to represent a nesting join

$$C^{P_{1,2}, \dots, P_{k,2}} [P_{k+1,2} \rightarrow C]$$

and for each  $i \in [1, k]$  we set  $A_{i,psm}$  to represent

$$C^{P_{1,2}, \dots, P_{i-1,2}} [P_{i,2} \rightarrow P_{i+1,2}]$$

**Example 16** We demonstrate the proposed technique on a simple example. Assume the PSM diagram depicted on the left side of Figure 5. It was reverse engineered from an XML schema that we do not display. Its components were mapped to components from the PIM diagram in Figure 3 by **classMap**. The ancestors of  $Supply_{psm}$  are joined by PSM associations that represent nesting joins  $Part[Supply - Supplier \rightarrow ]$ ,  $ProductSet[Supply - Part \rightarrow ]$  and  $Supply[ProductSet \rightarrow ]$ , respectively. However, these joins do not describe the true semantics of the PSM associations. The true semantics is that suppliers are considered in the context of suppliers, parts and product sets as we showed in Example 6. Therefore, the domain expert selects the PSM class  $Supply_{psm}$  and puts  $k = 2$  to specify that  $Supply$  instances are not grouped only by  $ProductSet$  but also by the two ancestors  $Part$  and  $Supplier$ . We can easily verify that the conditions (C1–2) are satisfied. Both paths  $Supply - Supplier$  and  $Supply - Part$  contain a step  $Supply$  and (C1) is therefore satisfied. We have the following  $P_{1,1} = \cdot$ ,  $P_{1,2} = Supplier$ ;  $P_{2,1} = \cdot$ ,  $P_{2,2} = Part$ ;  $P_{3,2} = ProductSet$  where  $rev(\cdot) - Part = P_{2,2}$  and  $rev(\cdot) - ProductSet = P_{3,2}$  and (C2) is therefore satisfied as well. We can therefore update the nesting joins represented by the PSM associations according to the proposed technique as follows (respectively):

$$\begin{aligned} & Supply[Supplier \rightarrow Part] \\ & Supply_{Supplier}^{Supplier} [Part \rightarrow ProductSet] \\ & Supply_{Supplier, Part}^{Supplier, Part} [ProductSet \rightarrow] \end{aligned}$$

## 5 Conclusions

We studied reverse engineering of XML schemas. We supposed a set of XML schemas and an existing conceptual diagram. We proposed a semi-automatic method that finds semantic interrelations between the XML schemas and the conceptual diagram.

Currently, we are developing a tool for testing the proposed method in practice. We plan to extend the proposed algorithms with parameters for tuning them to fit real scenarios. It will be also necessary to further expand some technical details of the algorithms. We presented only a part of the conceptual model. The full model, that was proposed in (Nečaský 2008), contains some technical extensions that allow to model more advanced XML features and therefore need to be considered for reverse engineering as well. Further, some more advanced algorithms for measuring not only syntactical but also semantical similarity of strings could be utilized. Last but not least is an optimization of the proposed algorithms.

## References

- Bernauer, M., Kappel, G. & Kramler, G. (2003), Representing XML Schema in UML - An UML Profile for XML Schema, Technical report.
- Bernauer, M., Kappel, G. & Kramler, G. (2004), Representing XML Schema in UML - A Comparison of Approaches, in N. Koch, P. Fraternali & M. Wirsing, eds, 'ICWE', Vol. 3140 of *Lecture Notes in Computer Science*, Springer, pp. 440–444.
- Chiticariu, L., Hernández, M. A., Kolaitis, P. G. & Popa, L. (2007), Semi-Automatic Schema Integration in Clio, in C. Koch, J. Gehrke, M. N. Garofalakis, D. Srivastava, K. Aberer, A. Deshpande, D. Florescu, C. Y. Chan, V. Ganti, C.-C. Kanne, W. Klas & E. J. Neuhold, eds, 'VLDB', ACM, pp. 1326–1329.
- Dobbie, G., Xiaoying, W., Ling, T. & Lee, M. (2000), ORA-SS: An Object-Relationship-Attribute Model for Semi-Structured Data, Technical Report TR21/00, Dpt. of Computer Science, National University of Singapore.
- Domínguez, E., Lloret, J., Pérez, B., Rodríguez, Á., Rubio, A. L. & Zapata, M. A. (2007), A Survey of UML Models to XML Schemas Transformations, in B. Benatallah, F. Casati, D. Georgakopoulos, C. Bartolini, W. Sadiq & C. Godart, eds, 'WISE', Vol. 4831 of *Lecture Notes in Computer Science*, Springer, pp. 184–195.
- Jensen, M. R., Møller, T. H. & Pedersen, T. B. (2003), 'Converting XML DTDs to UML diagrams for conceptual data integration', *Data Knowl. Eng.* **44**(3), 323–346.
- Mani, M. (2004), ERX: A Conceptual Model for XML, in Z. Bellahsene, T. Milo, M. Rys, D. Suciu & R. Unland, eds, 'XSym', Vol. 3186 of *Lecture Notes in Computer Science*, Springer, pp. 128–142.
- Miller, J. & Mukerji, J. (2003), *MDA Guide Version 1.0.1*, Object Management Group. <http://www.omg.org/docs/omg/03-06-01.pdf>.
- Nečaský, M. (2007), XSEM - A Conceptual Model for XML, in J. F. Roddick & A. Hinze, eds, 'Fourth Asia-Pacific Conference on Conceptual Modelling (APCCM2007)', Vol. 67 of *CRPIT*, ACS, Ballarat, Australia, pp. 37–48.
- Nečaský, M. (2008), Conceptual Modeling for XML, PhD thesis, Charles University. <http://kocour.ms.mff.cuni.cz/~necasky/dw/thesis.pdf>.
- Routledge, N., Bird, L. & Goodchild, A. (2002), UML and XML Schema, in X. Zhou, ed., 'Australasian Database Conference', Vol. 5 of *CRPIT*, Australian Computer Society.
- Shvaiko, P. & Euzénat, J. (2005), 'A Survey of Schema-Based Matching Approaches', *J. Data Semantics* **6**, 146–171.
- Thompson, H. S., Beech, D., Maloney, M. & Mendelsohn, N. (2004), *XML Schema Part 1: Structures (Second Edition)*, W3C. <http://www.w3.org/TR/xmlschema-1/>.
- Yang, W., Gu, N. & Shi, B. (2006), Reverse Engineering XML, in J. Ni & J. Dongarra, eds, 'IMSCCS (2)', IEEE Computer Society, pp. 447–454.
- Yu, A. & Steele, R. (2005), An Overview of Research on Reverse Engineering XML Schemas into UML Diagrams, in 'ICITA (2)', IEEE Computer Society, pp. 772–777.



# Synthesis of Orchestrators from Service Choreographies

Stephen McIlvenna<sup>1</sup>, Marlon Dumas<sup>2,1</sup>, Moe Thandar Wynn<sup>1</sup>

<sup>1</sup>Faculty of Science and Technology, Queensland University of Technology  
GPO Box 2434, Brisbane, Queensland 4001, Australia

{s.mcilvenna, m.wynn}@qut.edu.au

<sup>2</sup>Institute of Computer Science, University of Tartu, Estonia

marlon.dumas@ut.ee

## Abstract

Interaction topologies in service-oriented systems are usually classified into two styles: choreographies and orchestrations. In a choreography, services interact in a peer-to-peer manner and no service plays a privileged role. In contrast, interactions in an orchestration occur between one particular service, the orchestrator, and a number of subordinated services. Each of these topologies has its trade-offs. This paper considers the problem of migrating a service-oriented system from a choreography style to an orchestration style. Specifically, the paper presents a tool chain for synthesising orchestrators from choreographies. Choreographies are initially represented as communicating state machines. Based on this representation, an algorithm is presented that synthesises the behaviour of an orchestrator, which is also represented as a state machine. Concurrent regions are then identified in the synthesised state machine to obtain a more compact representation in the form of a Petri net. Finally, it is shown how the resulting Petri nets can be transformed into notations supported by commercial tools, such as the Business Process Modelling Notation (BPMN).

**Keywords:** service composition, choreography, orchestration, Petri nets, BPMN.

## 1 Introduction

A Service-Oriented Architecture (SOA) is a software architecture where the basic elements are services, meaning entities offer some functionality to other entities, which themselves can be services. At the implementation level, an SOA manifests itself in the form of a collection of software services that exist at certain endpoints and exchange messages according to certain contracts. A software service is called a Web Service (WS) if it applies Web standards such as eXtensible Markup Language (XML), Web Service Description Language (WSDL), and/or SOAP.

A typical approach to design an SOA is to identify basic services and then to compose them into larger services, or conversely, to identify larger services and to then decompose them into smaller services. In either case, the cornerstone for SOA design is the definition of

compositions of services. This work is concerned with how these compositions of services are modelled, and specifically, how different perspectives for modelling such compositions of services can be reconciled.

Depending on their interaction topology, service compositions are usually classified into two styles: choreographies and orchestrations (Peltz 2003). In a choreography, no service plays a privileged role (peer-to-peer topology), whereas in an orchestration, interactions occur between one particular service, the orchestrator, and a number of other subordinated services (hub-and-spoke topology). For example, Figure 1 illustrates a business-to-business choreography involving a buyer, a supplier and a shipper, while Figure 2 shows the corresponding orchestration.

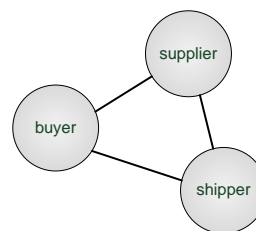


Figure 1: Choreographed composition

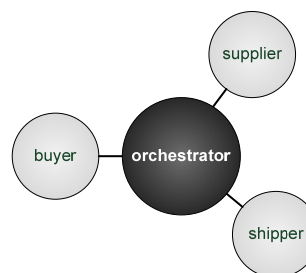


Figure 2: Orchestrated composition

The choice between a choreographed and an orchestrated service composition approach may be driven by a number of factors, often of an organisational nature. With reference to the above example, it may be that initially, the supplier deals with a single shipper for all orders. The buyer interacts with the supplier in order to agree on the purchase order and to pay for the goods, but when it comes to delivery issues, the buyer needs to interact directly with the shipper. Subsequently, the supplier may decide that in order to provide a more uniform customer experience and to closely monitor the performance of its shippers, it is desirable to have a single point of interaction with the customer (the ‘orchestrator’)

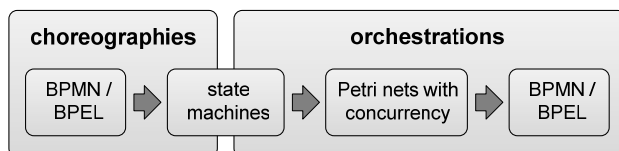
in Figure 2). This orchestrator would handle interactions related to the purchase order, payment and also delivery. Having done that, it becomes possible to introduce additional value-added services in the orchestrator, such as providing the buyer with the possibility to choose between making a single payment for the goods and for the delivery, or making separate payments, or choosing between different delivery modes.

From the technical perspective, this change in topology requires that an orchestrator is developed and deployed and that all the interactions between the services in the composition be channelled through the orchestrator. This paper provides a technique to automate the development of such an orchestrator from a given choreography.

Choreographies are initially represented using Finite State Machines (FSMs). Based on this representation, an algorithm is presented that synthesises the behaviour of an orchestrator, which is also represented as a state machine. The synthesised state machine may be rather large and unreadable, because the interactions that an orchestrator needs to manage tend to occur in any order or in parallel, and this parallelism leads to state explosion.

Accordingly, concurrent regions are identified in the synthesised state machine in order to obtain a more compact representation in the form of a Petri net. This is achieved using existing results from the field of theory of regions (Cortadella, Kishinevsky et al. 1998). The paper then shows how the resulting Petri nets can be transformed into notations supported by commercial tools, such as the Business Process Modelling Notation (BPMN) (Object Management Group 2008). The result is a skeleton of an orchestrator. This skeleton can be extended and refined using existing business process modelling tools and used as a basis to generate code in executable languages such as the Business Process Execution Language (BPEL).

The entire tool chain for orchestrator synthesis is depicted in Figure 3. The specific contributions of the paper are: (i) a technique for synthesising orchestrators from choreography specifications using state machines as a specification language, and (ii) a technique for transforming Petri nets into BPMN diagrams. The tool chain, starting from a choreography specified as FSMs, has been implemented and tested on a number of scenarios with varying degrees of complexity.



**Figure 3: Composition viewpoints bridged by orchestrator synthesis with state machines**

The rest of the paper is structured as follows. Section 2 provides some background on service behaviour modelling. Section 3 presents the proposed orchestrator synthesis algorithm. Section 4 shows how Petri nets representing the resulting orchestrator can be transformed into BPMN diagrams. Section 5 describes the validation of the approach and Section 6 discusses related work. Conclusions are drawn in Section 7.

## 2 Background: Modelling service behaviour

Choreographies can be described as a set of *interface FSMs*, where an interface FSM defines both the message exchanges in which a given participant can engage, and their message control-flow dependencies.

In an environment where messages can be buffered and transmission is not instantaneous, unbounded queues can be problematic for compatibility verification (Bultan, Su et al. 2006). Reasoning with protocols can be simplified by either bounding the queue length (Berardi, Calvanese et al. 2005), or removing queues entirely (Benatallah, Casati et al. 2006).

An assumption sometimes taken (Yellin and Strom 1997; Benatallah, Casati et al. 2006) is an environment where message transmission is instantaneous, meaning the FSMs of the sender and receiver for any given interaction advance in synchrony. While this assumption is not in line with some communication protocols which support asynchronous message transfer, it appears solutions developed under the assumption of synchronous messaging may be transposable to asynchronous environments as referred to in Section 6. Thus, we make this assumption of synchronicity to simplify orchestrator synthesis.

Having made this assumption about the communication medium, we also need to adopt a language for capturing service behaviour. Languages such as BPMN and BPEL could be used to for this purpose. However, these languages are complex in terms of the number of constructs they support, hindering their suitability as a basis for reasoning about service behaviour. Also, these languages are meant for capturing service-oriented business processes rather than capturing the behaviour that one service exposes to other services. For example, both languages allow one to capture internal actions and decisions that a service-oriented process makes during its execution. However, when capturing service behaviour for orchestrator synthesis purposes, we are only interested in capturing the externally visible behaviour that each service exposes.

In light of this, we adopt FSMs as the language for capturing service behaviour. This choice is in line with previous work on component and service behaviour specification (Yellin and Strom 1997; Benatallah, Casati et al. 2006; Berardi, Calvanese et al. 2005). Accordingly, a choreography is captured as a collection of communicating state machines. This design choice is further justified in Section 6.

Specifically, we rely on the notion of *interface FSM*, which is essentially an FSM where the transitions are labelled with communication actions – either sending or receiving a message. To ensure protocols describe only external behaviour, the FSMs we deal with are deterministic, meaning that every state is labelled, and for any given state there are no two outgoing transitions with exactly the same label. In order to deterministically model choices based on message content, we use Boolean guards expressed in terms of message content. For example to capture the requirement that depending on the content of a message of type *OrderResponse*, the FSM should follow one transition or another, we append expressions like  $[processed=true]$  and  $[processed=false]$  to the message type. Hence, one transition could be



labelled with 'OrderResponse[processed=true]' and another with 'OrderResponse[processed=false]'.

The orchestrators synthesised by the algorithm presented later in the paper, typically perform message forwarding, and we found that representing a message forwarding action as separate receive and send transitions leads to cumbersome models. To simplify the specification of orchestrators, message exchanges are represented as a quadruplet  $\text{Exchange} = (p_{\text{from}} : \text{Party}, p_{\text{to}} : \text{Party}, \text{msg} : \text{MessageType}, \text{forwarding} : \text{Boolean})$ . This tuple specifies the initial sender of the message, the final recipient of the message, the type of the message, and whether the message is directly exchanged between the two parties or it is received from one party (by the orchestrator) and forwarded to the other.

Formally, an interface FSM is a tuple  $(S, s_0, E, \delta)$  where:

- $S$  is a set of states. A state is labelled with an identifier in the case of an elementary state, or a tuple, possibly with other nested tuples, in the case of a composite state derived during the merging of two or more other interface FSMs.
- $s_0 \in S$  is the initial state.
- $E$  is a set of message exchanges specified as quadruplets, as previously discussed.
- $\delta : S \times E \rightarrow S$  is a transition function to connect states via message exchanges.

A *behavioural interface* is defined as a combination of an interface FSM and the set of parties the FSM represents, the pair  $(P : \{\text{Party}\}, \text{sm} : \text{FSM})$ . Normally, the set of parties  $P$  of a behavioural interface will contain only one element, because a behavioural interface represents the behaviour that one party exposes to one or several parties. But in the case of an orchestrator service, the behavioural interface represents the aggregated behaviour of multiple subordinated services and  $P$  will contain multiple parties. For convenience, we will use the term *orchestrator interface*, as shorthand to refer to the behavioural interface of an orchestrator service.

Behavioural interfaces of the choreography participants in our working example are shown in Figures 4 to 6. These interfaces are based on the Voluntary Inter-industry Commerce Standard (VICS) for order management (GS1 US 2007).

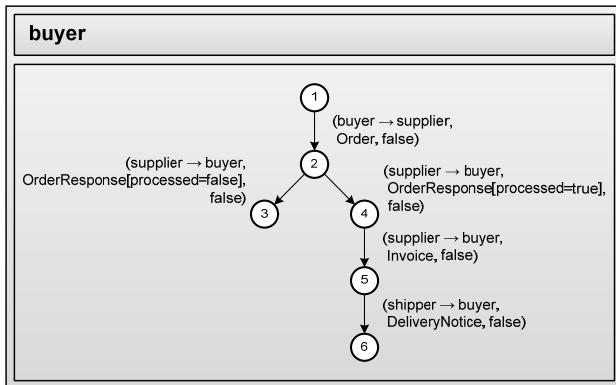


Figure 4: Behavioural interface of the buyer

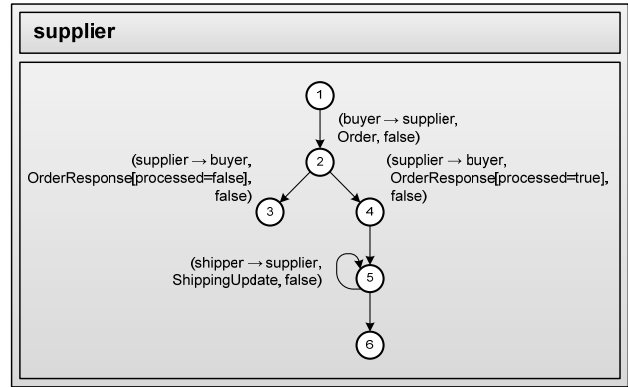


Figure 5: Behavioural interface of the supplier

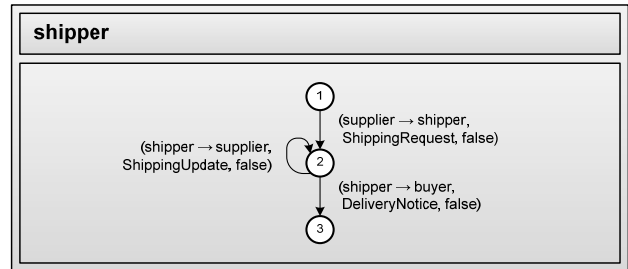


Figure 6: Behavioural interface of the shipper

### 3 Orchestrator synthesis

The goal of orchestrator synthesis is to generate a behaviourally compatible message forwarding service capable of intercepting messages within a given choreography. In this paper, we propose a synthesis algorithm which merges interface FSMs from any number of parties in a choreography to produce an orchestrator. The algorithm comprises three main functions and makes use of the following auxiliary functions:

- For any ordered list  $L$ ,  $\text{enqueue}(L, n)$  adds  $n$  to the end of  $L$ , and  $\text{dequeue}(L)$  removes the first element from the front of  $L$ . If  $n$  is a list, each element is added in order to the end of  $L$ .
- For a message exchange  $e$ ,  $\text{fromParty}(e)$ ,  $\text{toParty}(e)$ ,  $\text{msg}(e)$ ,  $\text{forwarding}(e)$ , retrieve the corresponding components.
- For a state machine  $\text{sm}$ ,  $\text{states}(\text{sm})$  returns all states,  $\text{initialState}(\text{sm})$  the initial state, and  $\text{finalStates}(\text{sm})$  the set of final states, which is derivable by finding all states having no outgoing transitions.
- For a composite state  $c$ ,  $\text{s1}(c)$  and  $\text{s2}(c)$  return the two contained states.
- For a transition  $t$ ,  $\text{exchange}(t)$  returns the message exchange, and  $\text{source}(t)$  and  $\text{target}(t)$  the source and target states respectively.

For better logic clarity,  $\delta$  is also characterised as a set of Transition objects, each composed of a message exchange, one source and one target state. Also, unordered sets are denoted by  $\{\}$  and ordered lists by  $[\ ]$ .

### 3.1 Synthesis of multiple interfaces

Our algorithm is capable of synthesising any number of behavioural interfaces forming a choreography into a single orchestrator interface. The approach used is to fully merge two of the input interfaces, then merge the result with a third interface, and so on in pairs, until all input interfaces have been synthesised into the orchestrator. This high level processing of taking all input interfaces and synthesising the orchestrator is performed by Function 1, `synthesise()`, and is visualised in Figure 7. If a deadlock condition is detected while synthesising any interface pair, `synthesise()` immediately indicates synthesis is not possible.

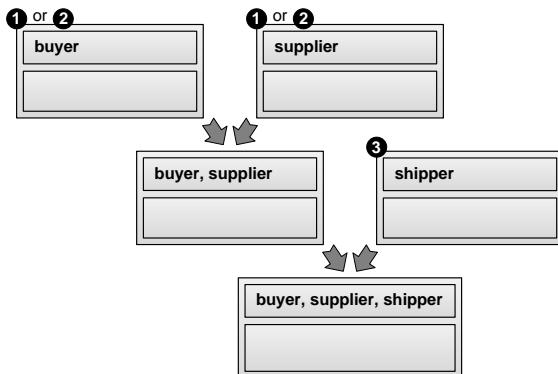
```

Function: synthesise
Input:  $I_{all} : [Interface]$ 
Output: Interface  $\cup$  Deadlock
Preconditions:  $|I_{all}| \geq 2$ 
Variables: synthesised : Interface  $\cup$  Deadlock
begin
    synthesised := synthesiseInterfacePair(
        dequeue( $I_{all}$ ), dequeue( $I_{all}$ ))
    if synthesised is Deadlock
        return synthesised
    end if
    while  $I_{all} \neq []$ 
        synthesised := synthesiseInterfacePair(
            synthesised, dequeue( $I_{all}$ ))
        if synthesised is Deadlock
            return synthesised
        end if
    end while
    return synthesised
end

```

**Function 1: synthesise()**

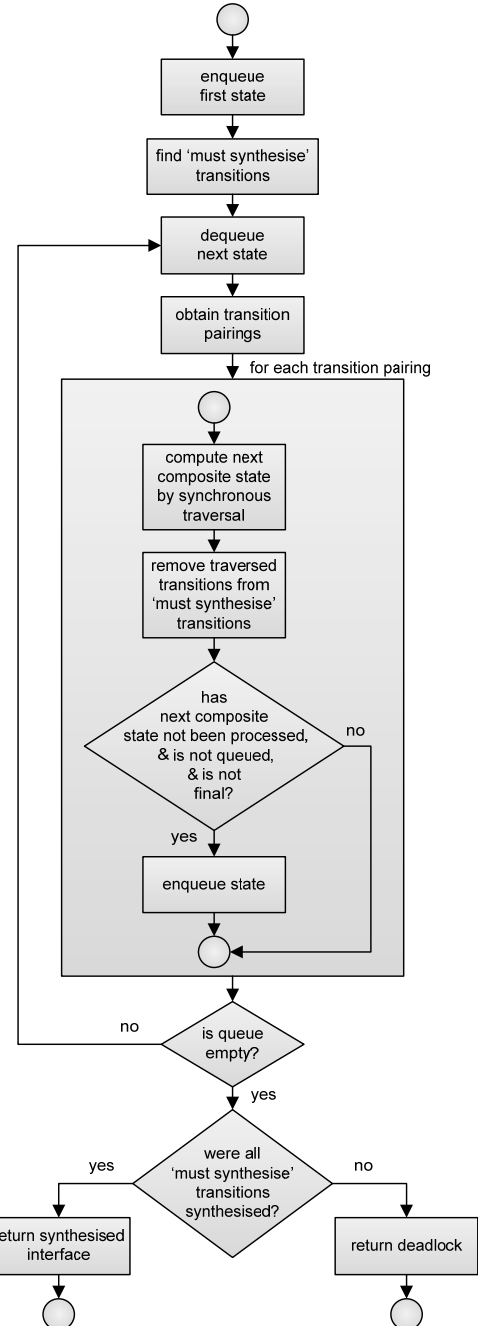
Input interfaces must be ordered according to the role of each interface in the choreography. It is assumed only one interface FSM can send a message from the initial state – the others start their execution by receiving a message. In the case of the choreography involving a buyer, a supplier and a shipper, the buyer would start the choreography. The two interfaces exchanging the first message must be the first two in the input list to ensure the synthesised orchestrator interface captures the choreography from start to end. Subsequent interfaces in the list must appear in the order they come into the choreography. The effect of this ordering on the working example is shown in Figure 7.



**Figure 7: Synthesis of interface pairs showing the ordering of the input list**

### 3.2 Synthesis of an interface pair

A pair of interfaces can be completely merged by Function 2, `synthesiseInterfacePair()`, depicted in Figure 8. This function synchronously traverses the two input interfaces to build the orchestrator interface by performing a breadth-first search of the two input FSMs, and combining the lists of parties represented by both interfaces. The input FSMs are searched in synchrony by looking at state pairs where a state pair is composed of a state from each FSM. Merging a pair of interfaces is realised with a visitor pattern (Palsberg and Jay 1998) to perform the breadth-first search.



**Figure 8: Overview of synthesiseInterfacePair()**

The root node for performing a breadth first search is the composite state derived from the two initial states of input FSMs. This state is the initial state of the synthesised FSM, and is placed in a queue of states to be processed. Only composite states are placed in this queue,

and each time one is dealt with, it is placed into a pool of states which have already been processed. Each state is visited only once. Synthesis of the input pair is complete when all states have been visited and the queue of states to visit is empty.

After the initial state has been removed from the queue, another function attempts to find pairs of message exchanges that can occur between the two input FSMs. A match is apparent when both FSMs can synchronously exchange a message and advance to their next respective states.

As seen in Function 3, transitionPairings() explores which messages can be sent or received from the states currently being processed from each input FSM, and attempts to find matches. If a match is found, the pair of

transitions is added as a tuple to a set of pairings. If a match for a transition is not found, it is added to a tuple with the other element empty, representing that the interaction cannot yet be orchestrated, which is a normal situation if the party with which this transition should be paired has not yet been dealt with. A tuple indicating that a transition could not be matched is only added to the set of pairings if the exchange is orchestrated, or is related to an interface not yet considered for orchestrator synthesis. Otherwise, the inability to match a given transition with at least one other transition means that one party can send a message while the other is not in a state to receive it, or vice-versa. In some cases, if there is no other way to progress to a new pair of states from the current pair of states, this situation indicates a deadlock.

**Function:** synthesiseInterfacePair

**Input:**  $i_a = (P_a, sm_a = \{S_a, s_{0\_a}, E_a, \delta_a\}) : \text{Interface}$ ,  $i_b = (P_b, sm_b = \{S_b, s_{0\_b}, E_b, \delta_b\}) : \text{Interface}$

**Output:**  $\text{Interface} \cup \text{Deadlock}$

**Preconditions:**  $\forall s \in S_b, s$  is ElementaryState

**Variables:**  $S_{0\_m}, S_{\text{current}}, S_{\text{new}}, S_{\text{toAdd}} : \text{CompositeState}$

$S_{\text{toVisit}} : [\text{CompositeState}]$ ,  $S_{\text{visited}} : \{\text{CompositeState}\}$

$sm_m = \{S_m, s_{0\_m}, E_m, \delta_m\} : \text{FSM}$

$t_{a\_mustSynthesise} : \{\text{Transition}\}$ ,  $t_{b\_mustSynthesise} : \{\text{Transition}\}$

$e_{\text{new}} : \text{Exchange}$

$TP_{\text{all}} : \{(\text{Transition}, \text{Transition})\}$

**begin**

$S_{0\_m} := (s_{0\_a}, s_{0\_b})$

$\text{states}(sm_m) := \{s_{0\_m}\}$

$\text{enqueue}(S_{\text{toVisit}}, S_{0\_m})$

$P_{\text{known}} := P_a \cup P_b$

$t_{a\_mustSynthesise} := \{t \in \delta_a ; e \in \text{exchange}(t) \mid \text{fromParty}(e) \in P_{\text{known}} ; \text{toParty}(e) \in P_{\text{known}} ; \neg \text{forwarding}(e) \bullet t\}$

$t_{b\_mustSynthesise} := \{t \in \delta_b ; e \in \text{exchange}(t) \mid \text{fromParty}(e) \in P_{\text{known}} ; \text{toParty}(e) \in P_{\text{known}} ; \neg \text{forwarding}(e) \bullet t\}$

**while**  $S_{\text{toVisit}} \neq \{\}$

$S_{\text{current}} := \text{dequeue}(S_{\text{toVisit}})$

$S_{\text{visited}} := S_{\text{visited}} \cup \{S_{\text{current}}\}$

$TP_{\text{all}} = \text{transitionPairings}(i_a, s1(S_{\text{current}}), i_b, s2(S_{\text{current}}))$

**for each**  $(t_a, t_b)$  **in**  $TP_{\text{all}}$

**if**  $t_a \neq \text{NULL} \wedge t_b \neq \text{NULL}$

$s_{\text{new}} := (\text{target}(t_a), \text{target}(t_b))$

$e_{\text{new}} := (\text{fromParty}(\text{exchange}(t_a)), \text{toParty}(\text{exchange}(t_a)), \text{msg}(\text{exchange}(t_a)), \text{true})$

**else if**  $t_a = \text{NULL}$

$s_{\text{new}} := (s1(S_{\text{current}}), \text{target}(t_b))$

$e_{\text{new}} := \text{exchange}(t_b)$

**else**

$s_{\text{new}} := (\text{target}(t_a), s2(S_{\text{current}}))$

$e_{\text{new}} := \text{exchange}(t_a)$

**end if**

$\text{states}(sm_m) := \text{states}(sm_m) \cup \{s_{\text{new}}\}$

$\text{exchanges}(sm_m) := \text{exchanges}(sm_m) \cup \{e_{\text{new}}\}$

$t_m := t_m \cup \{(s_{\text{current}}, e_{\text{new}}) \rightarrow s_{\text{new}}\}$

$t_{a\_mustSynthesise} := t_{a\_mustSynthesise} \setminus \{t_a\}$

$t_{b\_mustSynthesise} := t_{b\_mustSynthesise} \setminus \{t_b\}$

**if**  $s_{\text{new}} \notin S_{\text{visited}} \wedge s_{\text{new}} \notin S_{\text{toVisit}} \wedge \neg(s1(s_{\text{new}}) \in \text{finalStates}(sm_a) \wedge s2(s_{\text{new}}) \in \text{finalStates}(sm_b))$

$\text{enqueue}(S_{\text{toVisit}}, s_{\text{new}})$

**end if**

**end for**

**end while**

**if**  $t_{a\_mustSynthesise} \neq \{\} \vee t_{b\_mustSynthesise} \neq \{\}$

**return** Deadlock

**end if**

**return**  $\text{Interface}(P_a \cup P_b, sm_m)$

**end**

**Function 2: synthesiseInterfacePair()**

```

Function: transitionPairings
Input:  $i_a = (P_a, sm_a = (S_a, s_{0\_a}, E_a, \delta_a)) : \text{Interface}$ ,
 $s_a : \text{State}$ ,  $i_b = (P_b, sm_b = (S_b, s_{0\_b}, E_b, \delta_b)) : \text{Interface}$ ,
 $s_b : \text{ElementaryState}$ 
Output:  $\{(Transition, Transition)\}$ 
Preconditions:  $s_a \in S_a$ ,  $s_b \in S_b$ 
Variables:  $T_{pairs} : \{(Transition, Transition)\}$ 
 $T_{out\_a}, T_{out\_b}, T_{done\_a}, T_{done\_b} : \{Transition\}$ 
 $P_{known} : \{Party\}$ 
begin
   $T_{out\_a} := \{t \in T_a \mid \text{source}(t) = s_a\}$ 
   $T_{out\_b} := \{t \in T_b \mid \text{source}(t) = s_b\}$ 
  for each  $t_a$  in  $T_{out\_a}$ 
    for each  $t_b$  in  $T_{out\_b}$ 
      if  $\text{exchange}(t_a) = \text{exchange}(t_b)$ 
         $T_{pairs} := T_{pairs} \cup \{(t_a, t_b)\}$ 
         $T_{done\_a} := T_{done\_a} \cup \{t_a\}$ 
         $T_{done\_b} := T_{done\_b} \cup \{t_b\}$ 
      end if
    end for
  end for
   $P_{known} := P_a \cup P_b$ 
  for each  $t_a$  in  $T_{out\_a}$ 
    if  $t_a \notin T_{done\_a} \wedge (\text{fromParty}(\text{exchange}(t_a)) \notin P_{known} \vee$ 
       $\text{toParty}(\text{exchange}(t_a)) \notin P_{known}) \vee$ 
       $\text{forwarding}(\text{exchange}(t_a))$ 
       $T_{pairs} := T_{pairs} \cup \{(t_a, \text{NULL})\}$ 
    end if
  end for
  for each  $t_b$  in  $T_{out\_b}$ 
    if  $t_b \notin T_{done\_b} \wedge (\text{fromParty}(\text{exchange}(t_b)) \notin P_{known} \vee$ 
       $\text{toParty}(\text{exchange}(t_b)) \notin P_{known}) \vee$ 
       $\text{forwarding}(\text{exchange}(t_b))$ 
       $T_{pairs} := T_{pairs} \cup \{(\text{NULL}, t_b)\}$ 
    end if
  end for
return  $T_{pairs}$ 
end

```

### Function 3: transitionPairings()

With respect to the working example, when merging the pair of behavioural interfaces (buyer, supplier), and when the pair of states being processed is (buyer 4, supplier 4) the message exchange (supplier  $\rightarrow$  buyer, Invoice, false) is not added to the pairings as it is known this exchange cannot yet occur with the supplier, since the supplier first needs to send a ShippingRequest to the shipper. Therefore, for the state (buyer 4, supplier 4) the pairing function only returns the tuple (NULL, (supplier  $\rightarrow$  shipper, ShippingRequest, false)) indicating one non-orchestrated interaction be added to the orchestrator interface.

If it occurs that the set of message exchange pairings is empty, then there are no messages that can be synchronously exchanged in the respective states of the composite state being processed. Therefore, deadlock is declared and orchestrator synthesis is terminated. The partially synthesised orchestrator could also be preserved if desired. If message exchange pairings are found, they are added to the orchestrator interface, such that matching exchange pairs are added as orchestrated exchanges, where forwarding=true, and others as non-orchestrated exchanges, where forwarding=false. Based on the result of pairing message exchanges, a new message transition

is added to the synthesised FSM along with a composite state representative of the synchronous advancement performed. This new composite state is then queued for processing, but only if the state has not already been processed or queued for processing, and is not final. Synthesis of the two input FSMs is complete once both have been completely traversed.

With respect to the working example, the composite state (buyer 4, supplier 4) causes the pairing function to return (NULL, (supplier  $\rightarrow$  shipper, ShippingRequest, false)). The non-orchestrated interaction is added to the orchestrator and the next state leading on from (buyer 4, supplier 4) is computed. This next state will involve the same buyer state (buyer 4) since the first element of the pairing tuple is NULL, and will involve the next supplier state (supplier 5). The state (buyer 4, supplier 5) is then added to the queue of states to visit.

The product of synthesising the buyer and supplier interfaces is shown in Figure 9, where each state (buyer x, supplier y) is shortened to  $(b_x, s_y)$ . All interactions involving the buyer and supplier are orchestrated since these parties are fully represented in the orchestrator. Before the shipper is processed, it is not possible to know whether its interface FSM is compatible with that of the other services in the choreography, so interactions involving the shipper remain non-orchestrated.

Finally, synthesise() processes the shipper interface and the interface where  $P = \{\text{buyer, supplier}\}$  to produce the complete orchestrator  $P = \{\text{buyer, supplier, shipper}\}$ .

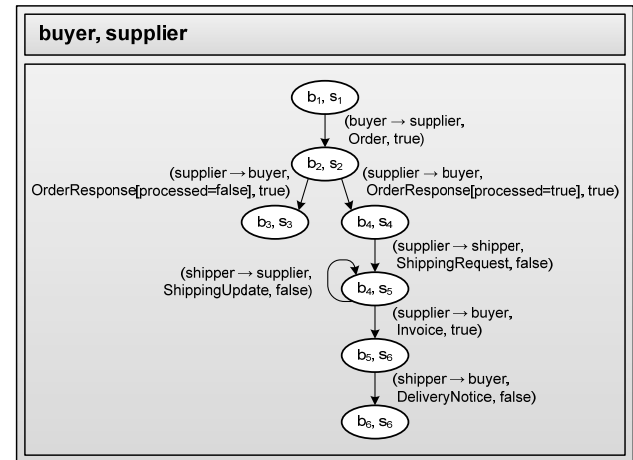
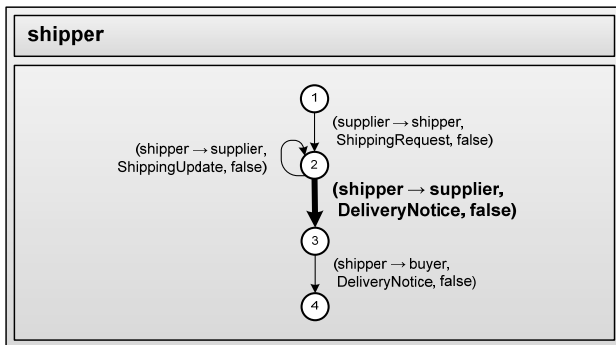


Figure 9: Result of merging the buyer and supplier interfaces

The synthesiseInterfacePair function performs a depth-first search over a graph whose nodes represent pairs of state – one state from each interface being synthesised. Each of these ‘state pairs’ is visited at most once. Similarly, every pair of transitions (one transition from each of the original state machines) is visited at most once. Thus the worst-complexity of the algorithm is  $O(N_1 * N_2 + E_1 * E_2)$  where  $N_1$  and  $N_2$  are the number of states in each of the two interfaces, and  $E_1$  and  $E_2$  are the number of transitions.

### 3.3 Deadlock detection

The synthesis algorithm detects deadlock within `synthesiseInterfacePair()` by identifying non-synthesised transitions in both input interfaces which block synchronous advancement and cause any transition to not be added to the synthesis product. With respect to our working example, let's try to synthesise the interface pair from Figure 9 and Figure 10. The set of known parties for this synthesis is {buyer, supplier, shipper} meaning the synthesised product captures the combined behaviour of these three parties, and therefore all interactions between these three parties must be captured as orchestrated interactions in the synthesised interface.



**Figure 10: Shipper interface with an interaction causing deadlock**

The synthesis algorithm proceeds normally until the shipper attempts to send a `DeliveryNotice` to the supplier (the interaction highlighted in Figure 10). At this stage, the supplier's behaviour is already fully captured in the interface with which the shipper's interface is being merged (the interface shown in Figure 9), and it is known that the supplier can never be in a state where it can receive a message of type `DeliveryNotice`. Prior to attempting synthesis of the interfaces in Figure 9 and Figure 10, this troublesome transition in the shipper's interface is earmarked as being a 'must synthesise' transition because it is non-orchestrated and involves only synthesised parties. Since the transition cannot be traversed during the synthesis, it remains marked as 'must synthesise', along with any subsequent, unreachable transitions and other untraversed transitions from the interface in Figure 9. Deadlock is therefore detected if any 'must synthesise' transition cannot be synthesised.

## 4 Orchestrator modelling

Synthesised orchestrators provide the grounding for migrating choreographies to orchestrations, which can then be augmented with additional functionality, such as lower-level multi-party message adaptation, or higher-level value-adding.

While state machines provide a practical bridge between the two service composition viewpoints, they are not very readable or maintainable due to state explosion, a drawback encountered where multiple messages may be exchanged in any order. In such a scenario, each possible sequence of message exchanges must be explicitly represented, leading to verbose FSMs, which although executable, are not easily maintainable.

Petri nets provide a graphical language capable of expressing concurrent interactions, and consist of place nodes, transition nodes, and directed arcs. They have been used for high level workflow management (van der Aalst 1998) and on a more detailed level for modelling process behaviour (Hinz, Schmidt et al. 2005).

### 4.1 State machines to Petri nets

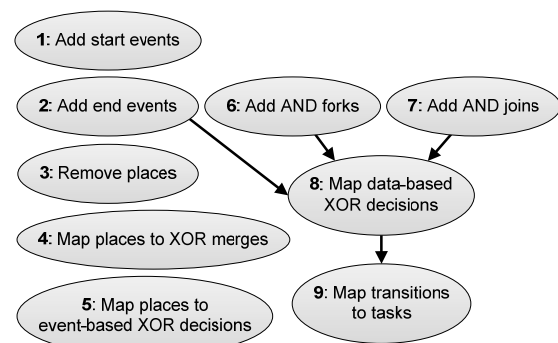
We leverage the theory of regions (Cortadella, Kishinevsky et al. 1998) to identify regions of concurrency in FSMs, and replace these regions with their concurrent equivalents. The techniques derived from this theory are implemented in the Petrify tool (Cortadella, Kishinevsky et al. 1997). By reusing this tool, we can transform interface FSMs into free-choice Petri nets with concurrency.

A Petri net is said to be free-choice if and only if for every two transitions, if they share any input place, they share all input places (Chrzastowski-Wachtel, Benatallah et al. 2003). We enforce this restriction to prevent mixing of choice and synchronisation, which is difficult to represent in a higher level modelling language. Once orchestrators are represented as Petri nets, logic is easier to read, but the model clarity can be further improved for business users or developers if displayed in a recognised modelling language such as BPMN or BPEL. We identified value-adding augmentation as a potential use of orchestrators, so we developed a technique to translate Petri nets into BPMN diagrams, so value can be added at a business level by altering the logic in the diagrams.

### 4.2 Petri nets to BPMN

We developed a set of rules to transform Petri nets into BPMN diagrams. These rules are summarised in Figure 11, where each rule identifies a pattern in the graph to be replaced by the output specified in the rule. Dependencies between rules are introduced in order to reduce the complexity of pattern definitions. An overview of the rule patterns and outputs is presented in Figure 12, taking snippets from an extended version of our working example.

The rules are only concerned with the logic inside the BPMN pool artefact representing the orchestrator. Other pools and connecting message flows are not generated by the rules as they do not affect the control-flow logic.



**Figure 11: Rules and their dependencies**

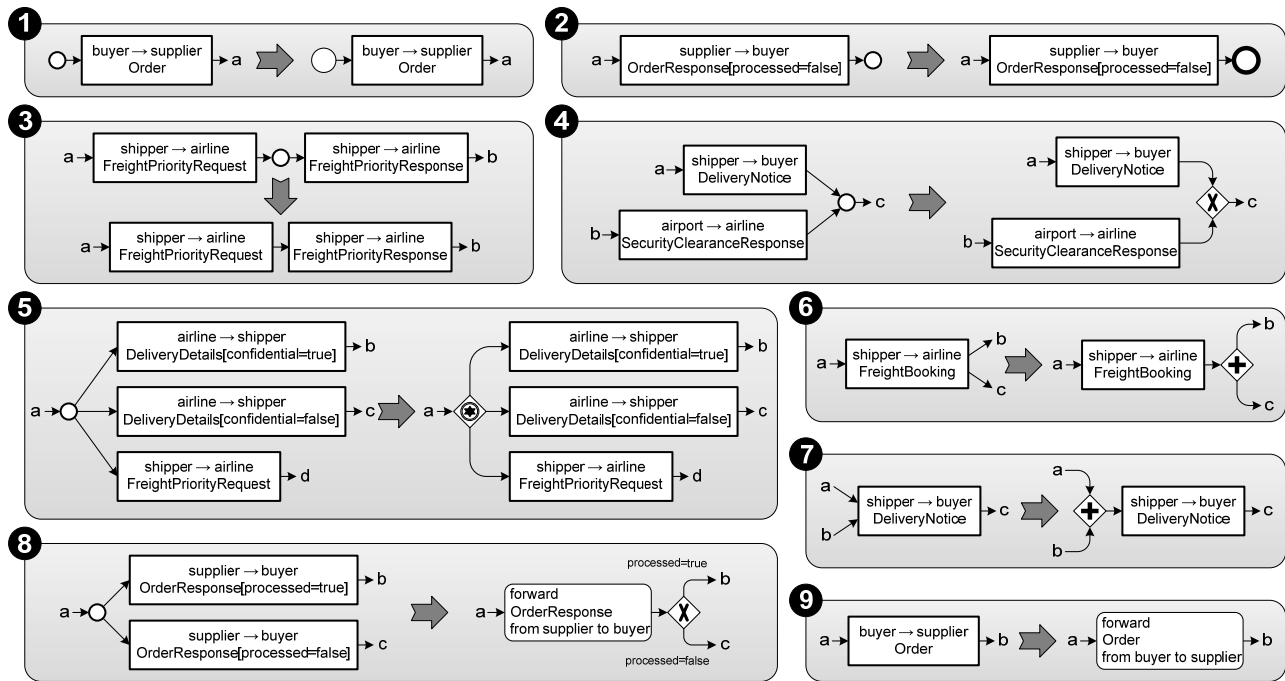


Figure 12: Petri net to BPMN transformation rules

We chose to model all message interactions with BPMN tasks without any BPMN message event elements to reduce the repetition that is prevalent with message forwarding, where a message is sent immediately after it is received. Tasks are therefore used for modelling message receiving, sending, and forwarding. The example cases in Figure 12 consider only orchestrated interactions which forward messages, but the rules are equally applicable to Petri nets describing interactions of individual parties, such as the buyer or the supplier. In such instances, tasks are for sending and receiving only.

## 5 Validation

We developed tooling to validate the synthesis algorithm and rule-based transformations. Utilising the Eclipse Modelling Framework (EMF) we created a graphical editor to design interface FSMs, which were used by an implementation of the algorithm to produce models readable in the same editor. We extracted around two dozen sample choreographies from two industry standards for business-to-business interactions, namely the XML Common business Library (xCBL) (xCBL.org 2000) and the Voluntary Inter-industry Commerce Standard (VICS) (GS1 US 2007). The examples had between two and four participants, arranged in different topologies and with varying numbers of states and transitions per interface FSM. By altering the FSMs of choreography participants, we also generated sample choreographies with deadlocks that the tool was able to detect.

We noticed state machine orchestrators involving four parties became very verbose, as pairs of services may interact independently, thereby confirming the value of representing orchestrators in a higher-level language such as BPMN.

The tool for transforming Petri nets into BPMN diagrams utilises ProM (van Dongen, de Medeiros et al. 2005), a framework for process mining and analysis,

which provides a Petri net object model and an API for invoking the Petrify tool. BPMN diagrams are created through API and then imported into the BPMN modeller included in the SOA Tools Platform Project<sup>1</sup>.

We tested an implementation of the transformation rules using the collection of choreographies mentioned above, and successfully automated the generation of correct BPMN diagrams for orchestrators, and for individual choreography participants.

## 6 Related work

The language for service behaviour modelling proposed in this paper is directly inspired from work in the area of behaviour specification for software components and services (Yellin and Strom 1997; Benatallah, Casati et al. 2006). Our assumption of synchronous communication is also inspired from this prior work. Although this assumption may be seen as inapplicable in some cases, it has been shown that under some conditions, it is possible to transpose results obtained under the synchronous communication assumption to an asynchronous communication medium (Yellin and Strom 1997; Bultan, Su et al. 2006).

Standard notations for specifying orchestrations and choreographies include BPMN and BPEL. BPMN is intended for modelling business processes involving human tasks and/or automated tasks. Automated tasks in BPMN are typically delegated to external services. A BPMN model that includes only service tasks is essentially an orchestration. BPMN has also been shown to be suitable for modelling choreographies (Decker and Barros 2008). BPEL on the other hand, is primarily intended to model orchestrations. However, it is also possible to use BPEL for specifying business protocols (an alternative term for designating behavioural interfaces), and extensions to BPEL have been proposed

<sup>1</sup> <http://www.eclipse.org/stp/>

to make it usable for capturing choreographies (Decker, Kopp et al. 2007).

Both BPMN and BPEL can be translated to Petri nets using existing techniques. A transformation from BPEL to state machines is implemented by the WS-Engineer toolset (Howard, Emmerich et al. 2007). We are not aware of direct transformations from BPMN to state machines, but transformations exist from BPMN to Petri nets (Dijkman, Dumas et al. 2008), which under certain assumptions can then be expanded into state machines.

A transformation from FSMs to BPEL has also been proposed (Zhao, Bryant et al. 2005), however this transformation does not attempt to identify concurrent regions in the FSM in order to obtain a simpler BPEL process definition. Instead, the generated BPEL process definitions are fully sequential (no 'parallel flow' activities). Thus, if the original FSM is complex due to concurrent message exchanges being represented as interleaved sequences of message exchanges, the resulting BPEL process definitions will mirror this complexity.

In this paper, we use techniques from the theory of regions (Cortadella, Kishinevsky et al. 1998) to transform state machines to Petri nets. We then show how these Petri nets can be transformed to BPMN diagrams. Once a BPMN diagram is obtained, other existing techniques can be applied to transform these diagrams into BPEL for implementation purposes (Ouyang, Dumas et al. 2008).

The work presented in this paper can also be related to work on controllability analysis of service protocols (Lohmann et al. 2008). Controllability analysis is concerned with the following question: given a service protocol  $P$ , is there a partner protocol  $P'$  such that the choreography consisting of  $P$  and  $P'$  possesses certain characteristics such as proper termination? Meanwhile, in our work we take a choreography as a starting point, and we derive an orchestrator that is able to interact with all the existing parties in the choreography in order to mediate between all their interactions.

## 7 Conclusion

We have presented a tool chain for synthesising the behaviour of an orchestrator from a service choreography. This tool chain effectively provides a basis for altering the topology of a service-oriented system composed of services that engage in long-running conversations with one another.

The tool chain starts with the assumption that choreographies are represented as communicating FSMs, and that the communication medium is synchronous. We argued that state machines provide a suitable starting point for this tool chain, pointing out that, under reasonable assumptions, it is possible to transform choreographies specified using standard languages such as BPMN and BPEL into FSMs. We also argued that the assumption of synchronous communication provides a suitable basis for studying the problem of synthesising orchestrators from choreographies. Nonetheless, it would be interesting in future work to study the implications of relaxing this assumption.

At the core of the proposed tool chain lies an algorithm that takes as input a choreography captured as a collection of inter-connected FSMs, and synthesises an

orchestrator, also captured as an FSM. Acknowledging that FSMs do not provide a suitable basis for capturing orchestrator interfaces, the tool chain reuses an existing technique to transform the synthesised FSM into a Petri net. We then provide a rules-based transformation from Petri nets to BPMN, thus enabling orchestrator synthesis from choreographies using standard notations.

The proposed tool chain, including the algorithm for orchestrator synthesis and the transformation from Petri nets to BPMN, has been implemented and tested against choreographies extracted from industry standards.

As discussed in Section 1, orchestrators provide a single entry point into a service composition, and as such, they can facilitate the introduction of added value into a service composition. In particular, orchestrators can act as single points of payment or as entry points for tracking a service composition. A direction for future work is to study the organisational implications and the opportunities opened by the possibility of changing the topology of a service composition from a choreographed style to an orchestrated style.

**Acknowledgment:** This work was partly funded by an ARC Linkage Project (LP0669244) co-sponsored by SAP and Queensland Government.

## 8 References

- Benatallah, B., Casati, F., et al. (2006): Representing, analysing and managing web service protocols. *Data Knowledge Engineering* **58**(3):327-357.
- Berardi, D., Calvanese, D., De Giacomo, G., Hull, R. and Mecella, M. (2005): Automatic composition of transition-based semantic web services with messaging. *Proc. 31st VLDB Conference*, Trondheim, Norway, 613-624, VLDB Endowment.
- Bultan, T., Su, J., et al. (2006): Analyzing conversations of web services. *IEEE Internet Computing* **10**(1): 18-25.
- Chrastowski-Wachtel, P., Benatallah, B., Hamadi, R., O'Dell, M. and Susanto, A. (2003): A top-down Petri net-based approach for dynamic workflow modeling. *Proc. Business Process Management 2003*, Eindhoven, The Netherlands, 336-353, Springer-Verlag.
- Cortadella, J., Kishinevsky, M., Kondratyev, A. Lavagno, L. and Yakovlev, A. (1997): Petrify: A tool for manipulating concurrent specifications and synthesis of asynchronous controllers. *IEICE Transactions on Information and Systems*: **3**(E80-D):315-325.
- Cortadella, J., Kishinevsky, M., et al. (1998): Deriving Petri nets from finite transition systems. *IEEE Transactions on Computers* **47**(8): 859-882.
- Decker, G. and Barros, A. (2008): Interaction Modeling using BPMN. In *Business Process Management Workshops*, Lecture Notes in Computer Science, 4928:208-219, Springer-Verlag, Germany.
- Decker, G., Kopp, O., Leymann, F. and Weske, M. (2007): BPEL4Chor: Extending BPEL for modeling choreographies. *Proc. International Conference on Web Services*, Salt Lake City, Utah, USA, 296-303, IEEE.

- Dijkman, R. M., Dumas, M. and Ouyang, C. (2008): Semantics and analysis of business process models in BPMN. *Information and Software Technology*, To appear.
- GS1 US: Voluntary Interindustry Commerce Standard (VICS), [http://www.uc-council.org/ean\\_ucc\\_system/stnds\\_and\\_tech/vics\\_ed1.html](http://www.uc-council.org/ean_ucc_system/stnds_and_tech/vics_ed1.html). Accessed 18 October 2007.
- Hinz, S., Schmidt, K. and Stahl, C. (2005): Transforming BPEL to Petri nets. *Proc. International Conference on Business Process Management*, Nancy, France, 220–235, Springer-Verlag.
- Howard, F., Emmerich, W., Kramer, J., Magee, J., Rosenbalum, D. and Uchitel, S. (2007): Model checking service compositions under resource constraints. *Proc. ESEC/FSE'07*. Cavtat near Dubrovnik, Croatia, 225-234, ACM.
- Lohmann, N., Massuthe, P., Stahl, C. and Weinberg, D. (2008): Analyzing interacting WS-BPEL processes using flexible model generation. *Data Knowledge Engineering* **64**(1):38–54.
- OASIS (2007): Web Services Business Process Execution Language Version 2.0, <http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.pdf>. Accessed 4 February 2008.
- Object Management Group (2008): Business Process Modeling Notation, v1.1, <http://www.omg.org/docs/formal/08-01-17.pdf>. Accessed 24 March 2008.
- Ouyang, C., Dumas, M., ter Hofstede, A. H. M., van der Aalst, W. M. P. (2008): Pattern-based translation of BPMN process models to BPEL web services. *International Journal of Web Services Research* **5**(1):42-61.
- Palsberg, J. and Jay, C. (1998): The essence of the visitor pattern. *Proc. 22nd IEEE International Computer Software and Applications Conference*, Vienna, Austria, 9-15, IEEE.
- Peltz, C. (2003): Web services orchestration and choreography. *IEEE Computer* **36**(10):46-52.
- van der Aalst, W. M. P. (1998): The application of Petri nets to workflow management. *Journal of Circuits, Systems and Computers* **8**(1):21-66.
- van Dongen, B., de Medeiros, A., Verbeek, H., Weijters, A., and van der Aalst, W. (2005): The ProM framework: A New Era in Process Mining Tool Support. *Proc. 26th International Conference on Application and Theory of Petri Nets (ATPN)*, Miami, Florida, 444-454, Springer-Verlag.
- xCBL.org (2003): XML Common Business Library, <http://www.xcbl.org/xcbl40/documentation.shtml>. Accessed 12 September 2007.
- Yellin, D. M. and Strom, R. E. (1997): Protocol specifications and component adaptors. *ACM Transactions on Programming Languages and Systems* **19**(2):292-333.
- Zhao, W., Bryant, B.R., Cao, F., Bhattacharya, K. and Hauser, R. (2005): Transforming Business Process Models: Enabling Programming at a Higher Level. *Proc. IEEE International Conference on Services Computing (SCC)*, Orlando, FL, USA, 173-180. IEEE.



# Towards Accurate Conflict Detection in a VCS for Model Artifacts: A Comparison of Two Semantically Enhanced Approaches

Kerstin Altmanninger

Gabriele Kotsis

Department of Telecooperation  
Johannes Kepler University Linz, Austria  
Email: [kerstin.altmanninger|gabriele.kotsis]@jku.ac.at

## Abstract

In collaborative software development the utilization of Version Control Systems (VCSs) is a must. For this important task some graph-based VCSs for model artifacts already emerged. Optimistic approaches, which are nowadays the designated ones, allow parallel editing of one resource and therefore changes can result in conflicts and inconsistencies. To be flexible for the ever increasing variety of modeling environments and languages VCSs should be independent of the modeling environment and applicable on any modeling language. Those VCS characteristics implicate a lack of information for the conflict detection method by virtue of firstly receiving solely the state of an artifact without concrete editing operations and secondly due to unavailable knowledge about the semantics of a modeling language. In such VCSs inconsistencies would even arise more often. Hence, accurate conflict detection methods are indispensable for the realization of optimistic, environment and language independent VCSs. This can be achieved by providing some understanding about the models's semantics which is possible by specifying machine interpretable formal semantics. Therefore, in this work, a comparison of two semantically enhanced conflict detection approaches is presented with respect to their suitability for the integration in an optimistic, environment and language independent VCS for model artifacts to achieve more accurate conflict reports.

**Keywords:** Version Control System, parallel model development, model evolution, model comparison, conflict detection, model consistency, model-driven engineering.

## 1 Introduction

The shift from code-centric to model-centric software development places models as first class artifacts in model-driven engineering (MDE). A major prerequisite for the wide acceptance of MDE is the availability of proper methods and tools for traditional software development, such as build tools, test frameworks or Version Control Systems (VCSs). Considering the latter, VCSs are particularly essential to enable collaborative editing and sharing of *model artifacts* like UML, ER or domain specific modeling languages (DSML) models.

Different systems use different strategies to provide collaborative editing. With the utilization of

pessimistic VCSs, model developers can work on the same set of model artifacts. Parallel editing of the same artifact is prevented by locking. *Optimistic* VCSs instead are crucial when the development process proceeds in parallel. Those systems enable each model developer to work on a personal copy of a model artifact, which may result in conflicting modifications. Such conflicting modifications need to be resolved and finally merged by appropriate techniques for model comparison, conflict detection, conflict resolution and merging.

Generic VCSs like CVS<sup>1</sup> or Subversion<sup>2</sup> are not applicable to model artifacts since they apply text-based comparison in a line-based manner and therefore cannot provide adequate conflict reports. *Graph-based* techniques instead need to be utilized to preserve the logical structure of model artifacts.

Most current optimistic, graph-based VCSs for model artifacts are coupled to a specific modeling environment e.g., the IBM Rational Software Architect (RSA)<sup>3</sup>. Since modelers evolve models for different system stages and application areas they also utilize different modeling environments e.g., for each purpose or language the most appropriate or preferred one. Environment specific VCSs are not widely applicable and modelers are bounded to a specific modeling environment. Hence, the kind of coupling of the version control functionalities to a model environment is essential. Therefore, standalone VCSs, so-called *environment independent* VCSs, like Odyssey-VCS (Murta et al. 2008) are preferable. Such systems allow modelers to use their modeling environment of preference for editing their model artifacts which leads to a better acceptance of the VCS.

In view of the fact that MDE is not only about UML and in the light of a growing number of DSMLs, VCSs which are solely applicable on specific modeling languages are often not usable. E.g., approaches of VCSs like Cicchetti et al. (2008), Oda & Saeki (2005) and RSA provide solely versioning capabilities for UML models. Hence, the number of supported modeling language is an important characteristic of a VCS. A *modeling language independent* (e.g., MOF-based) VCS like Odyssey-VCS (Murta et al. 2008) is desirable. Summing up, the utilization of an environment and language independent VCS is of interest for model developers since they can choose their preferred modeling environment for editing model artifacts and furthermore can use the VCS for a number of modeling languages.

To achieve a merged, consistent model artifact an *accurate conflict detection method* is essential. To find out the accuracy of the conflict detection method the definition of Leser & Naumann (2006) to deter-

Copyright ©2009, Australian Computer Society, Inc. This paper appeared at the Sixth Asia-Pacific Conference on Conceptual Modelling (APCCM 2009), Wellington, New Zealand, January 2009. Conferences in Research and Practice in Information Technology (CRPIT), Vol. 96, Markus Kirchberg and Sebastian Link, Ed. Reproduction for academic, not-for profit purposes permitted provided this text is included.

<sup>1</sup><http://www.nongnu.org/cvs/>

<sup>2</sup><http://subversion.tigris.org/>

<sup>3</sup><http://www-306.ibm.com/software/awdtools/architect/swarchitect/>

mine the *effectiveness* of the method can be applied. Therefore, the results gained of the conflict detection method and the actual perception in the reality are considered. With the result *true-positive*, a conflict has been detected by the conflict detection method and in reality. Accordingly, the result *true-negative* states that a conflict has neither been detected by the method nor in reality. Those two results correspond to the best case for accurate conflict detection. The accuracy of the method can be reduced by *false-positive* and *false-negative* results. Those are conflicts reported by the method which are actually no conflicts in reality or those which have not been detected by the method.

For dealing with concurrent modifications on models, in an environment independent and language independent VCS, the concentration on the reduction of false-negative and false-positive results is particularly essential and more challenging. Firstly, environment independent VCS can only operate on the state of a model artifact whereas environment specific VCSs can trace the modification performed by the developers. Since environment specific VCSs receive a editing history those systems dispose of more information for the conflict detection method than environment independent VCSs (Antkiewicz & Czarnecki 2007, Dig et al. 2008). Secondly, VCSs for specific languages can provide language specific conflict reports and therefore gain more accuracy in conflict detection opposed to language independent VCS approaches. For language independent VCS the method to detect conflicts must be general for any modeling language and therefore can not rely on the model languages' semantics. Hence, it is necessary not only to consider the logical structure of models in terms of a graph-based representation but also to "*understand*" the model's semantics in order to provide more accurate identification of conflicts. This can be achieved with different techniques by specifying the modeling languages semantics for semantic conflict detection between concurrently edited model versions.

The remainder of this paper is structured as follows: Section 2 motivates the need for semantic specifications in order to provide an accurate conflict detection method. In Section 3, a case study is given which depicts two elaborated semantically enhanced conflict detection approaches by means of a concrete example. Further on, the findings gained out of the case study are discussed. Section 4 presents related work and finally a conclusion and future prospects is given in Section 5.

## 2 Semantics for Accurate Conflict Detection

**Conflict Detection.** In the following, a *scenario in a VCS* in which two developers concurrently edit a copy of one and the same model artifact is considered. Thus, the developer who submits the edited model first can proceed with the check-in without any additional effort. The developer who submits the edited model version to the VCS secondly, needs to go through the basic VCS phases (comparison, conflict detection, conflict resolution and merge). After accomplishing this task (s)he can check-in a merged version of the edited model versions.

Because the history of editing operations is not available in an environment independent system the changes performed by the developers need to be identified in the first phase. In other words, the structural features of the model artifacts, namely the attributes and references of a model element, are inspected by applying a 3-way-comparison (Mens 2002). Therefore the differences between the model artifact ( $V''$ ) which is going to be checked-in and its ancestor version ( $V$ )

and the differences between the last revision ( $V'$ ) in the VCS repository and  $V$  are calculated. The input for the comparison phase constitutes a XMI serialization of the three different versions of the model artifacts. The result of this comparison phase, based on identifying attributes designated in the metamodel, is a **structural diff** comprising added, deleted, and updated elements (Altmanninger 2008). In the second phase, conflict detection, techniques to detect conflicting situations on basis of the two **structural diffs** resulting from the 3-way comparison phase are provided. Conflicts, however, are presented through concurrent operations of one and the same model element like update-update or update-delete operations. Furthermore concurrent create-create operations may also lead to a conflict if they differ in their properties. Those conflicts, also called syntactic conflicts (Altmanninger 2008) because they are detected through a structural comparison of the syntax of the model artifacts, need to be resolved by the model developer. Otherwise the model versions cannot be merged and stored in the repository.

**Problems.** Since editing operations between model versions are not available in an environment independent VCS and the method is not aligned to a specific modeling language, some conflicts between different model artifacts may remain hidden or are no actual conflicts in reality. This insufficiency can be ascribed to the fact that the conflict detection method has too little information about the meaning of the model artifacts under comparison.

For example, if a false-positive result is received it means that a conflict has been detected by the method which does not constitute a conflict in reality. The reason for such a result is raised by structural modifications performed by developers on model artifacts, which do not actually change the meaning of a model. More concrete, some modeling languages offer different ways to express one and the same circumstance. For example, in UML activity diagrams, decision nodes as well as conditional nodes are two equivalent ways to express alternative branches in a process, which could in fact result in a conflict if two developers edit a model concurrently by using different, semantically equivalent modeling concepts. Hence, to avoid a false-positive result, a mechanism is needed to express the semantics of those constructs to identify them as *equivalent concepts*. If an equivalent concept, however, has been detected the model developer has to be informed about this circumstance and should be able to store both "semantically equivalent concepts".

Moreover, the situation can occur that the method does not detect a conflict which is actually one in reality (false-negative). For example, concurrent modifications on a model may not result in an obvious conflict when syntactically different parts of the model (e.g., different model elements) were edited. Nevertheless, they may interfere with each other due to side effects (Thione & Perry 2005, Shao et al. 2007, Mens 2002), thus yielding an actual conflict, which, without considering the model's semantics, would remain hidden. Reasons could be, firstly, the violation of constraints, relationships or context conditions (*static semantics*) which cannot be operated on by utilizing a solely structural difference computation algorithm. Secondly, also concurrent changes of the behavior of a model artifact could affect a merged model artifact not incorporating behavioral side-effects (*behavioral semantics*). Such, also called *semantic conflicts* (Altmanninger 2008) should additionally be reported to the model developer. Opposed to syntactic conflicts, semantic conflicts do not need to be resolved

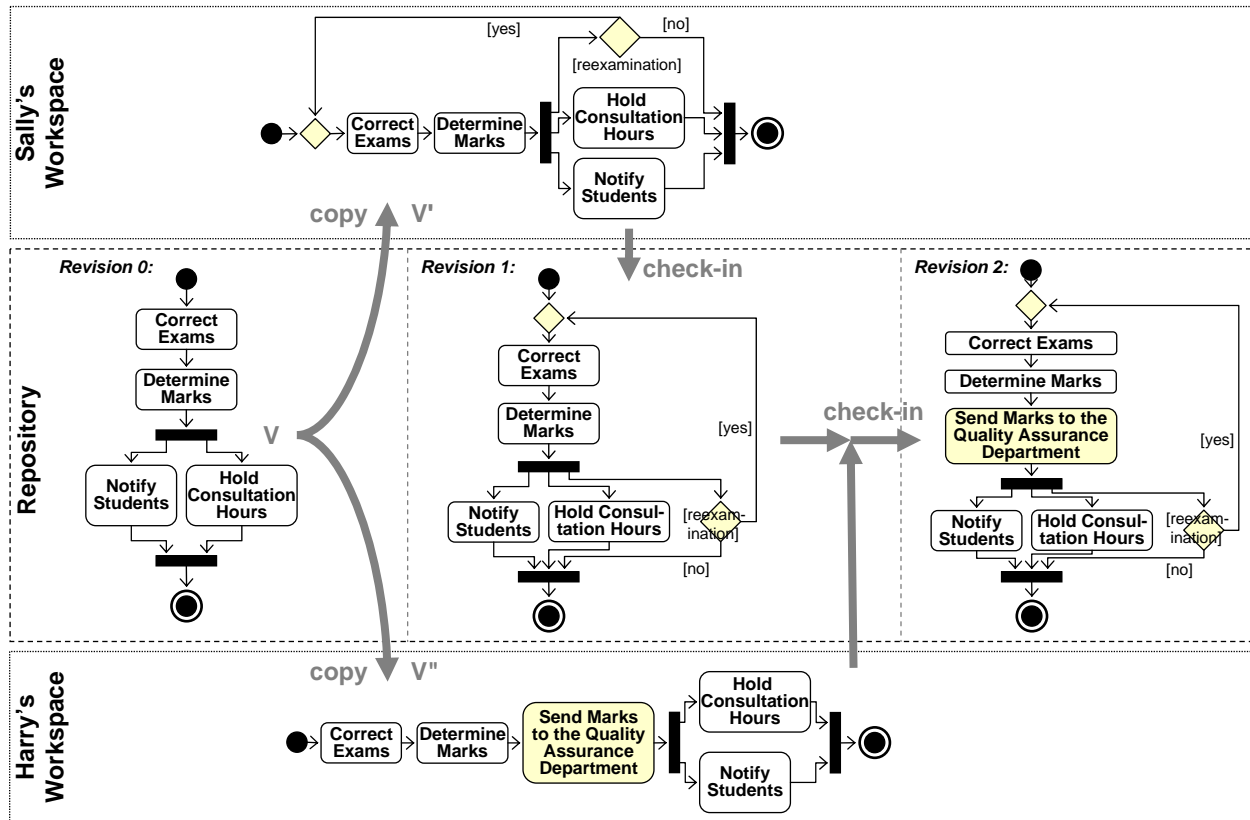


Figure 1: VCS check-in process of UML activity diagram versions.

by the model developer since the model versions can be merged if no syntactic conflict has been detected. Semantic conflicts, therefore, serve as additional notification mechanisms to model developers that the consistency of the resulting model version after the merge cannot be guaranteed without resolving the semantic conflict(s).

As described, to reduce false-positive and false-negative results to achieve an accurate conflict detection method, some understanding about the model's semantics is essential. Concretely, as identified previously, the specification of the three *semantic aspects* (Altmanninger et al. 2007), namely equivalent concepts, static, and behavioral semantics, of modeling languages need to be provided. In order to achieve this goal, a VCS needs an enhancement to cope with these issues. An algorithm, which checks the model versions for semantic interferences with pairwise comparison, would solve the previously mentioned issues to achieve more accurate conflict detection. This technique, however, is not feasible since it may need a lot of computing power if working on large and complex models. Moreover a more abstract approach is desirable to ease the maintenance of the conflict detection method.

**Solution.** To enhance the conflict detection method the meaning about the models semantics need to be specified in a formal manner. Neither implicit semantics, which are not formalized at all, nor informal semantics, solely understandable by humans, can be taken into account for conflict detection in a VCS since they cannot be processed by machines. Instead, *formal semantics*, with which it is possible to define a semantic mapping which is a functional or relational definition that relates both, the elements of the syntax and the elements of the semantic domain, need to be considered. Many different kinds of formal se-

mantics evolved according to various application areas and disciplines (Harel & Rumpe 2004, Uschold 2003, Sheth et al. 2005, Sheth & Gomadam 2007, Slonneger & Kurtz 1995). Those different kinds of formal semantics vary mainly in the used *formalism* to express the semantic domain and the according mapping. Therefore, two possibilities to express semantics exist, which can also be found in a mixed presentation, namely in pure *mathematics* (Broy et al. 2006) or in the *metamodeling language* (Clark et al. 2008). Depending on the purpose of the semantic description, mathematics are utilized when the modeling language, for which semantics should be defined, does not conveniently provide the appropriate mechanisms ones need (e.g., Formal Semantics for UML<sup>4</sup>). To enable more accurate conflict detection both formalisms (mathematics and metamodeling languages) can be utilized.

### 3 Case Study

The case study was conducted solely on behavioral modeling languages since only those are capable to cover all three semantic aspects mentioned before. Therefore, the DSML Web Services Business Process Execution Language (WSBP) (OASIS 2007, Altmanninger et al. 2007) and a subset of the general purpose language UML activity diagram have been deployed. To receive comprehensive comparison results of the applicability of the two semantic formalisms both modeling languages have been applied and examples for all three semantic aspects have been established.

In the following, an extract of the case study is presented in terms of an UML activity diagram example. To start with, the environment and language independent comparison and conflict detection method is

<sup>4</sup><http://www.cs.queensu.ca/~stl/internal/uml2/>

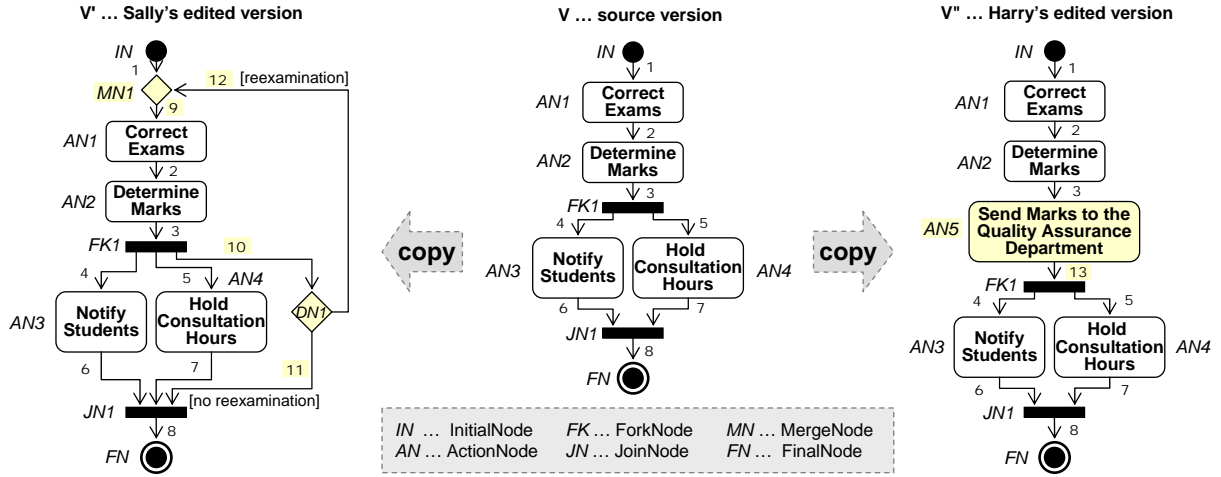


Figure 2: UML activity diagram versions with their according element names.

applied on the UML activity diagram model artifacts. Subsequently, two different semantically enhanced approaches are explained. Both of them can detect an additional semantic conflict between the parallel edited UML activity diagram model artifacts. To elaborate the advantages and disadvantages of those approaches the findings out of the case study are discussed.

### 3.1 Environment and Language Independent Conflict Detection

Fig. 1 illustrates a scenario for a typical check-in process of a VCS where two developers (Sally & Harry) concurrently edited copies of an UML activity diagram. The UML activity diagram in the scenario encapsulates the procedure for assessment of examinations. Sally edits her personal working copy by adding an additional *DecisionNode* reexamination. If a reexamination for the examination is provided, the control flow edge's target is a new inserted *MergeNode* in front of the first activity in the procedure. Otherwise the control flow edge's target is the *JoinNode* at the end of the procedure. Sally submits her edited working copy ( $V'$ ) first to the repository. Since the last version in the repository ( $V$ ) is the one she created a working copy of the check-in process can proceed. Harry has edited his personal working copy by adding an *ActionNode* after the determination of marks. Harry, then wants to check-in his version ( $V''$ ). Since Sally submitted her version first, Harry needs to apply the VCS check-in phases to obtain a finally merged version. The conflict detection method operating on the abstract syntax of the model versions is not capable to understand the semantics e.g., of a control flow, concurrency or data flow which can be expressed by UML activity diagram. Hence, no conflict is reported and the editing operations of both developers are merged.

In Fig. 2 abbreviations for the *ControlNodes* and *ActionNodes* and numbers for the names of the *ActivityEdges* of the UML activity diagrams are introduced for the description of the environment and language independent comparison and conflict detection method.

In Listing 1 the differences and according conflict sets between the three different versions of the UML activity diagrams are stated. Therefore the method described in previous works (Altmanninger 2008, Altmanninger et al. 2007) is applied. The structural differences between the source version ( $V$ ) and the edited versions ( $V'$ ,  $V''$ ) are com-

puted in the sets *Creates*, *Deletes* and *Updates*. The computed sets of differences then serve as input to detect conflicts (*Con*) which origin through concurrent create-create (*CrCon*), update-update (*UpdCon*) and delete-update (*DelCon*) editing operations.

```

Creates' = {V' - V} = {MN1, 9, 10, DN1, 11, 12}
Updates' = {V -> select (e | e.isUpdated (V, V'))}
          = {1 (REFS), AN1 (ROL), FK1 (REFS), JN1 (ROL)}
Deletes' = {V - V'} = {}
Creates'' = {V'' - V} = {AN5, 13}
Updates'' = {V -> select (e | e.isUpdated (V, V''))}
          = {3 (REFS), FK1 (ROL)}
Deletes'' = {V - V''} = {}

CrCon = {Creates' -> intersection (Creates'')
        -> select (e | e.areNotEqual (V', V''))} = {}

UpdCon = {Updates' -> intersection (Updates'')
        -> select (e | e.areNotEqual (V', V''))} = {}

DelCon = {(Updates' -> intersection (Deletes''))
        -> union (Updates'' -> intersection (Deletes''))} = {}

Con = {UpdCon -> union (CrCon -> union (DelCon))} = {}

```

Listing 1: Conflict detection between the UML activity diagram versions.

Between the edited model versions of Sally & Harry no conflict could be detected, with the structural difference computation and conflict detection method, despite the fact that semantically interferences exist. A semantic conflict should be reported in order to make Harry aware of the situation that the source and the set of targets of FK1 have been concurrently edited. This means, if the two versions of Sally & Harry are merged, the resulting UML activity diagram provides a *ForkNode* which has been concurrently updated (update-update conflict) by both developers. The model developer needs to obtain a notification about this semantic conflict in order to be able to react appropriately to prevent an incorrect merging of model versions as it would be the case for this example. Hence, a technique which incorporates the comparison of the control flow is essential. In the following two approaches to tackle this task are presented.

### 3.2 Semantically Enhanced Conflict Detection

**Mathematical-based Approach:** Dingel et al. (2006) present in their work a basic abstract syntax for activity diagrams using simple set-theoretical

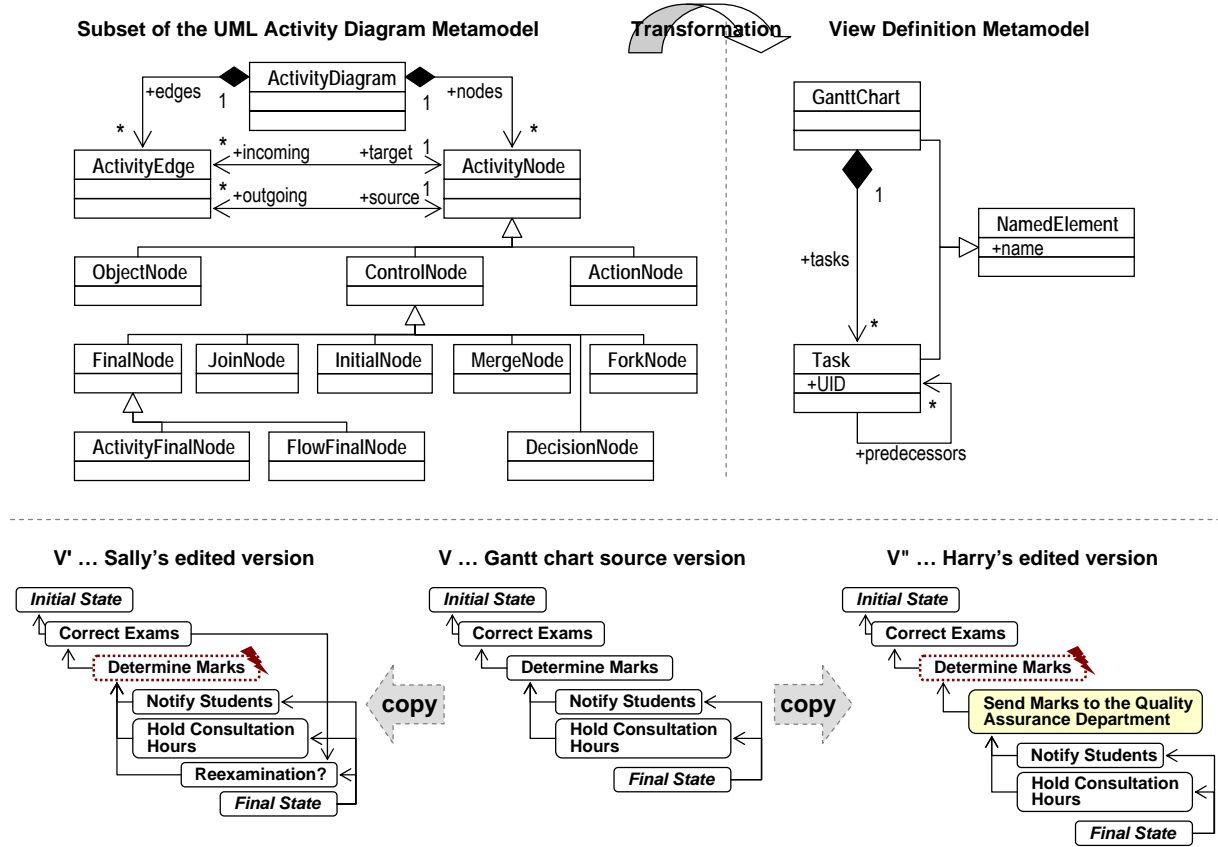


Figure 3: UML activity diagram and Gantt chart metamodels and according transformation to obtain the Gantt chart model versions.

Table 1: Mathematical instances of the UML activity diagram versions.

Sally's edited version	Source version	Harry's edited version
(1,IN,AN1)	(1,IN,AN1)	(1,IN,AN1)
(2,AN1,AN2)	(2,AN1,AN2)	(2,AN1,AN2)
(4,AN2,AN3)	(4,AN2,AN3)	(3,AN2,AN5)→cr
(5,AN2,AN4)	(5,AN2,AN4)	(4,AN5,AN3)→upd
(6,AN3,FN)	(6,AN3,FN)	(5,AN5,AN4)→upd
(7,AN4,FN)	(7,AN4,FN)	(6,AN3,FN)
(10,AN2,DN1)→cr		(7,AN4,FN)
(11,DN1,FN)→cr		
(12,DN1,AN1)→cr		

notation. To detect the conflict between the concurrently edited model versions of Sally & Harry the definition for the control flow from Dingel et al. (2006) can be utilized but need to be modified for the sake of detecting additional conflicts. A control flow is defined as an *ActivityEdge* that starts an *ActivityNode* after the previous one is finished. Neither objects nor data may pass along a control flow edge. Therefore,  $cf \in ControlFlow$  is defined as a 3-tuple  $cf = (name, source, target)$  where

1.  $name \in String$  is the name/identifier of this *ActivityEdge*.
2.  $source \in InitialNode \cup DecisionNode \cup ActionNode$  is the source of this control flow *ActivityEdge*.
3.  $target \in FinalNode \cup DecisionNode \cup ActionNode$  is the target of this control flow *ActivityEdge*.

Applying the definition for the control flow on the three different model versions the results stated in Table 1 can be identified. A conflict detection algorithm computed on those 3-tuple instances of the mathematical definition can now detect a create-create conflict on basis of concurrent specified different targets for control flows for the source node *AN2*.

**Metamodel-based Approach:** To detect a semantic conflict due to concurrent updates of the control flow the problem domain can also be abstracted by means of a metamodel (e.g., a Gantt chart) and according transformation.

In the top of Fig. 3 both metamodels are stated with an associated transformation which maps the root *ActivityDiagram* element to the *GanttChart* element, the *InitialNode* element to a *Task* without predecessors and each *FinalNode*, *DecisionNode* and *ActionNode* element to a *Task* with their according predecessor elements.

```
Creates'={DN1}
Deletes'={}
Updates'={AN1(REFS),FN(REFS),AN2(ROL)}
Creates''={AN5}
Deletes''={}
Updates''={AN3(REFS),AN4(REFS),AN2(ROL)}
CrCon=DelCon={}
UpdCon={AN2(ROL)}
Con={AN2(ROL)}
```

Listing 2: Conflict detection between the Gantt chart model versions.

Applying the transformation on the UML activity diagram versions (cf. bottom of Fig. 3) the conflict detection method, as given in Listing 1 on the UML activity diagram versions, can now be deployed on the Gantt chart models (cf. Listing 2).

Since those model versions make explicit the control flows between the actions/tasks consequently an additional semantic conflict can be reported. Specifically, the conflict occurred through a concurrent update of both developers of the references pointing to the node *AN2*.

This kind of semantic specification for more accurate conflict detection has been realized in preceding work in SMOVer, the “**Semantically Enhanced Model Version Control System**”<sup>5</sup> (Altmanninger 2008). Depending on the type of conflict, which should be avoided or detected, various “semantic view definitions” (target metamodels and transformations) can be specified. Each semantic view definition provided by SMOVer belongs to one of the three semantic aspects and holds a unique name which briefly describes the purpose. How complex and explicit such definitions are designed fully relies on the liability of the SMOVer administrator. He/She has to define a set of semantic view definitions for a specific modeling language all at once when setting-up the VCS or evolutionary so that a group of developers can collaborative edit model artifacts.

### 3.3 Discussion

As experiences and the above mentioned example showed, conflict detection is not restricted to mere structural difference detection, but with sophisticated methods also conflicts on a semantic level can be avoided and additional ones detected.

**Comparison of the different approaches.** For the comparison of the suitability of the approaches for the integration in an optimistic, environment and language independent VCS four important dimensions could be identified (cf. Table 2).

Starting with the first dimension, both approaches for semantically enhanced conflict detection are capable to define all **three semantic aspects** important for conflict detection in order to avoid conflicts due to the definition of equivalent concepts and to detect additional static and behavioral semantic conflicts. To tackle this task, mathematics can be utilized. Those are often called as more *precise* because people who communicate using mathematical terminology and notation tend to define things mathematically and therefore precise as well (Harel & Rumpe 2004, Broy et al. 2006). Nevertheless, sometimes mathematical techniques are not quite intuitive and thus need readers to cope with it. In the contrary, the utilization of transformations to different metamodels for the definition of semantics allows to directly execute those transformations using existing mechanisms. Furthermore no knowledge about a different notation for VCS administrators and model developers is needed. VCS administrators can specify new metamodels and according transformations to guarantee more accurate conflict detection with *well known techniques* (metamodels & transformations) and is therefore fast adaptable in an evolutionary manner.

In order to be able to **compute semantic conflicts**, for the mathematical-based approach, an interpreter of the defined rules has to be developed. By transforming the source model versions in a different probably more abstract representation, with the metamodel-based approach, the target models provide essential advantages compared to the mathematical approach. With this approach, the *same comparison and conflict detection method* as utilized for the source model versions can be applied on the target

Table 2: Comparison of the two semantic specification approaches.

Approaches	Definition of Semantic Aspects	Semantic Conflict Detection Method	Visualization of Conflicts	Reuse for Sem. Specifications
<i>Mathematical-based</i>	+	–	–	partially
<i>Metamodel-based</i>	+	+	+	partially

models. Moreover, the conflicts computed between the target model versions can be simply traced back to the source models by the model elements IDs.

For the mathematical-based approach a **presentation of the conflicting situation** as visualized in Table 1 would not be sufficient for model developers because it cannot be assumed that all model developers understand the mathematical instances of the model artifacts. Therefore an additional technique need to be developed to present the reason of the detected semantic conflict. In the metamodel-based approach, the model developers benefit from the automatically received presentation of the model versions in the semantic views which can be displayed to the developers without enormous additional effort. The model versions are represented in an abstract manner in each semantic view and therefore serve additionally for a better understanding about the conflicting situation.

For the definition of the semantic aspects for accurate conflict detection formal mathematical semantic specification, like those from Dingel et al. (2006) and Broy et al. (2006), can be reused to some extent as visualized in the preceding example. **Reuse** can also be enabled for the metamodel-based semantic specification approach. For instance, examples of the Atlas Transformation Language (ATL) Website<sup>6</sup> (Allilaire et al. 2006), a metamodel of a dependency graph (Altmanninger et al. 2007), a object life cycle (Ryndina et al. 2006) or Petri Nets (Farooq et al. 2007) can be applied for the purpose of making explicit specific aspects of a modeling language. In the UML activity diagram example out of the case study (cf. Section 3) e.g., the transformation of a UML activity diagram in a MSProject model of the ATL Transformation Website has been slightly adapted and reused for a different purpose, for making explicit the control flow for the sake of conflict detection.

Summing up, the metamodel-based approach, to gain accurate conflict detection, can be identified as *less time-consuming and complex to implement*. No additional comparison and conflict detection algorithm has to be invented for the models in the semantic view. Furthermore, the representation of the conflicts can be visualized on the models in the semantic view and can also be easily traced back to the original model versions contrary to the mathematical-based approach. Moreover, the *model developers gain more support* by the metamodel-based approach since they can view the detected semantic conflicts in the according semantic views, without the need of additional knowledge like the understanding of mathematical rules.

**Expedient employment of the semantically enhanced approaches.** The case study on different modeling languages (DSMLs and a subset of UML)

<sup>5</sup><http://smover.tk.uni-linz.ac.at/>

<sup>6</sup><http://www.eclipse.org/m2m/at1/at1Transformations/>



has revealed that different modeling languages provide different results of the conflict detection method without incorporating the semantics of a model. For a small DSML with few concepts like a metamodel of a Gantt chart (comprising tasks and references on predecessor tasks) no equivalent concepts can be defined to avoid conflicts and the abstraction level of the metamodel cannot be raised to find additional conflicts. Hence, the conflict detection method already computes almost solely true-positive results and therefore cannot be made more accurate. In the opposite if utilizing a language independent VCS with e.g., the general purpose language UML, the effectiveness of the language independent conflict detection method may not be satisfactory. UML, compared to DSMLs, provides much richer concepts and is more ambiguous. The more concepts a language provides the more likely it is to have equivalent concepts. Furthermore constraints need to be considered to control the models conforming to the language. Consequently, an effective conflict detection method for complex DSMLs or UML models needs information about the models' semantics unlike for simple DSMLs.

#### 4 Related Work

Besides SMOVer currently no optimistic, environment and language independent VCS with semantic enhancements for more accurate conflict detection on model artifacts exists.

Only two optimistic, environment independent but language specific approaches support semantically enhanced conflict detection to gain more accurate conflict reports. In the area of model engineering, Cicchetti et al. (2008) present an approach, which has not yet been implemented, for providing semantic awareness for the conflict detection method for UML models. Concretely, Cicchetti et al. (2008) propose to leverage conflict detection and resolution by adopting design-oriented descriptions endowed with custom conflict specifications. Hence, several conflicting situations, which can not be captured by a priori structural conflict detection mechanism can be specified that they refer to as "domain specific conflicts". The developers, however, are forced to enumerate all wrong cases in form of weaving models, which negatively affects the usability and scalability of the approach. Therefore, in the work of Cicchetti et al. (2008), each modification, which is not allowed to preserve a design pattern and the design pattern itself have to be specified in a weaving pattern (as they exemplified for the singleton design pattern). The approach of Cicchetti et al. (2008) focuses on the detection of previously undiscovered conflicts in terms of domain specific conflicts only, whereas behavioral semantic conflicts and the detection of previously falsely indicated conflicts as provided by SMOVer are not considered.

In the area of ontology engineering, *SemVersion* (Völkel 2006) performs semantic difference calculations on the basis of the semantics of the used ontology language. *SemVersion* is based on the Resource Description Framework (RDF), proposing the separation of language specific features (e.g., semantic difference) from general features (e.g., structural difference or branch and merge). Therefore, assuming using an RDF Schema as the ontology language and two versions (A and B) of an RDF Schema ontology, *SemVersion* uses RDF Schema entailment on model A and B and infers all possible triples. Now, a structural difference on A and B can be calculated in order to obtain the semantic difference. *SemVersion*, however, is limited to RDF based languages and therefore

does not provide the flexibility of being reused in the modeling domain.

#### 5 Conclusion and Future Work

In this paper, different semantically enhanced conflict detection approaches (for an optimistic, environment and modeling language independent VCS) which are capable to reduce the occurrence of false-positive and false-negative results have been elaborated, exemplified, and discussed.

The case study showed that the metamodel-based opposed to the mathematical-based approach needs less effort for implementation and also provides adequate support for model developers. In more detail the advantages of the metamodel-based approach opposed to purely mathematical specification of formal semantics are firstly the utilization of well known MDE constructs (metamodels & transformations) for the semantic enhancement. Secondly, the use of one and the same comparison and conflict detection method for the model versions in the syntax as well as semantic views. Thirdly, as a consequence of this transformation, developers are provided with a graphical presentation of the model versions in the respective semantic views. Hence, the conflict resolution phase can be completed by model developers with a better understanding about the conflicts detected.

Future work in this area will focus on a comprehensive evaluation of the effectiveness of the conflict detection method in terms of accuracy, which makes use of metamodel-based semantic specifications, realized in SMOVer. Therefore SMOVer is going to be compared to other VCSs for model artifacts like Odyssey-VCS (Murta et al. 2008) and RSA. Since the comparison of the effectiveness of the conflict detection method of SMOVer to other VCSs for models should be applied on as many tools as possible the subset of the general purpose modeling language UML activity diagram was chosen for the evaluation. Firstly because the majority of language specific VCSs provide versioning support for UML and secondly UML activity diagrams belong to behavioral languages and therefore all three semantic aspects, proposed in the context of semantically enhanced versioning (Altmanninger et al. 2007), can be exploited. Furthermore, it will be investigated in realizing a second release of the implementation of SMOVer which encapsulates an advanced representation of the result of the conflict detection phase. This is done by tracing back the falsely indicated syntactic conflicts and semantic conflicts from the semantic views.

In a longer prospect, it is planned to integrate the functionality to version a model in different languages (Störrle 2007) and support for metamodel versioning in SMOVer. Moreover, research in the area of VCSs for model artifacts will focus on building a VCS called AMOR (Adaptable Model Versioning) which comprises the characteristics of SMOVer and additional mechanisms (Altmanninger et al. 2008). Firstly, to further improve the accuracy of the conflict detection method, AMOR will provide supplementary to semantically enhanced conflict detection an operation-based conflict detection mechanism with which logged operations can be imported to the environment independent VCS. Secondly, AMOR will also support intelligent conflict resolution support, specifically aiming at techniques for the representation of differences between model versions and relieving modelers from repetitive tasks by suggesting proper resolution strategies, thus enhancing productivity and consistency of versioning.



## References

- Allilaire, F., Bézin, J., Jouault, F. & Kurtev, I. (2006), ATL – eclipse support for model transformation, in 'Eclipse Technology eXchange workshop (eTX) at the ECOOP 2006 Conference, Nantes, France'.
- Altmanninger, K. (2008), Models in conflict – towards a semantically enhanced version control system for models, in H. Giese, ed., 'Models in Software Engineering, Workshop and Symposia at MoDELS 2007, Nashville, TN, Reports and Revised Selected Papers', number 5002 in 'LNCS', Springer, pp. 293–304.
- Altmanninger, K., Bergmayr, A., Kotsis, G. & Schwinger, W. (2007), Semantically enhanced conflict detection between model versions in SMOVer by example, in 'Int. Workshop on Semantic-Based Software Development in conjunction with the Int. Conf. on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA)'.
- Altmanninger, K., Kappel, G., Kusel, A., Retschitzegger, W., Schwinger, W., Seidl, M. & Wimmer, M. (2008), AMOR - towards adaptable model versioning, in '1st Int. Workshop on Model Co-Evolution and Consistency Management (MCCM) at the ACM/IEEE 11th Int. Conf. on Model Driven Engineering Languages and Systems (MoDELS)'.
- Antkiewicz, M. & Czarnecki, K. (2007), Design space of heterogeneous synchronization, in 'Submitted to post-proceedings of Summer School on Generative and Transformational Techniques in Software Engineering (GTTSE)'.
- Broy, M., Crane, M. L., Dingel, J., Hartman, A., Rumpe, B. & Selic, B. (2006), 2nd UML 2 semantics symposium: Formal semantics for UML, in T. Kühne, ed., 'MoDELS 2006 Workshops', Vol. 4364 of LNCS, Springer, pp. 318–323.
- Cicchetti, A., Ruscio, D. D. & Pierantonio, A. (2008), Managing model conflicts in distributed development, in K. Czarnecki, I. Ober, J.-M. Bruel, A. Uhl & M. Völter, eds, 'Model Driven Engineering Languages and Systems, 11th International Conference, MoDELS 2008, Toulouse, France, September 28 - October 3', number 5301 in 'LNCS', Springer, pp. 311–325.
- Clark, T., Sammut, P. & Willans, J. (2008), *Applied Metamodeling: A Foundation for Language Driven Development*, second edition edn, Ceteva.
- Dig, D., Manzoor, K., Johnson, R. & Nguyen, T. (2008), 'Effective software merging in the presence of object-oriented refactorings', *IEEE Transactions on Software Engineering* **34**(3), 321–335.
- Dingel, J., Crane, M. L. & Diskin, Z. (2006), Activity diagrams: Abstract syntax and mapping to system model, Technical report, School of Computing, Queen's University, Kingston, Ontario, Canada. Draft – Version 0.0.
- Farooq, U., Lam, C. P. & Li, H. (2007), Transformation methodology for UML 2.0 activity diagram into colored petri nets, in '3rd Int. Conf. on Advances in Computer Science and Technology (ACST), Phuket, Thailand', ACTA Press, pp. 128–133.
- Harel, D. & Rumpe, B. (2004), 'Meaningful modeling: What's the semantics of "semantics"?' , *Computer* **37**(10), 64–72.
- Leser, U. & Naumann, F. (2006), *Informationsintegration: Architekturen und Methoden zur Integration verteilter und heterogener Datenquellen*, Dpunkt Verlag.
- Mens, T. (2002), 'A state-of-the-art survey on software merging', *IEEE Transactions on Software Engineering* **28**(5), 449–462.
- Murta, L., Corrêa, C., Prudêncio, J. & Werner, C. (2008), Towards Odyssey-VCS 2: Improvements over a UML-based version control system, in 'Int. Workshop on Comparison and Versioning of Software Models (CVSM)', ACM, pp. 25–30.
- OASIS (2007), 'Web Services Business Process Execution Language (WSBPPEL) Standard Version 2.0', <http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.pdf>.
- Oda, T. & Saeki, M. (2005), Generative technique of version control systems for software diagrams, in '21st IEEE Int. Conf. on Software Maintenance'.
- Ryndina, K., Küster, J. M. & Gall, H. (2006), Consistency of business process models and object life cycles, in T. Kühne, ed., 'Models in Software Engineering, Workshop and Symposia at MoDELS 2006, Genova, Italy, Reports and Revised Selected Papers', number 4364 in 'LNCS', Springer, pp. 80–90.
- Shao, D., Khurshid, S. & Perry, D. E. (2007), Evaluation of semantic interference detection in parallel changes: an exploratory experiment, in '23rd IEEE Int. Conf. on Software Maintenance, Paris, France'.
- Sheth, A. P. & Gomadam, K. (2007), The 4x4 semantic model: Exploiting data, functional, non-functional and execution semantics across business process, workflow, partner services and middleware services tiers, in '9th Int. Conf. on Enterprise Information Systems, Volume DISI, Funchal, Madeira, Portugal', pp. 5–12.
- Sheth, A., Ramakrishnan, C. & Thomas, C. (2005), 'Semantics for the semantic web: The implicit, the formal and the powerful', *Int. Journal on Semantic Web & Information Systems* **1**(1), 1–18.
- Slonneger, K. & Kurtz, B. (1995), *Formal Syntax and Semantics of Programming Languages: A Laboratory Based Approach*, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.
- Störrle, H. (2007), A formal approach to the cross-language version management of models, in 'Nordic Workshop on Model Driven Engineering'.
- Thione, G. L. & Perry, D. E. (2005), Parallel changes: Detecting semantic interferences, in '29th Annual Int. Computer Software and Applications Conf. (COMPSAC)', Vol. 1, IEEE Computer Society, pp. 47–56.
- Uschold, M. (2003), 'Where are the semantics in the semantic web?', *AI Magazine* **24**(3), 25–36.
- Völkel, M. (2006), 'D2.3.3.v2 SemVersion – versioning RDF and ontologies', <http://www.aifb.uni-karlsruhe.de/Publicationen/showPublikation?publid=1163>.

## Author Index

Alhadi, Arifah Che, 79  
Altmanninger, Kerstin, 139

Chen, Xing, 7

Dumas, Marlon, 129

Fagin, Ronald, 3

Ghose, Aditya K., 29  
Grün, Katharina, 107

Hasegawa, Ryo, 87  
Hausser, Roland, 17

John, Mathias, 39

Kühne, Thomas, 71  
Kaiya, Haruhiko, 87  
Kirchberg, Markus, iii  
Kitamura, Motohiro, 87  
Kiyoki, Yasushi, 7  
Koliadis, George, 29  
Kotsis, Gabriele, 139

Liegl, Philipp, 59  
Link, Sebastian, iii

Ma, Hui, 49  
McIlvenna, Stephen, 129  
Menzies, Alex, 29  
Morrison, Evan D., 29

Nečaský, Martin, 117  
Neumayr, Bernd, 107  
Noah, Shahrul Azman, 79

Saeki, Motoshi, 87  
Schäfer, Andreas, 39  
Schewe, Klaus-Dieter, 49  
Schrefl, Michael, 107

Thalheim, Bernhard, 49  
Thies, Gunnar, 97

Vossen, Gottfried, 97

Wynn, Moe Thandar, 129

Zakaria, Lailatulqadri, 79

## Recent Volumes in the CRPIT Series

ISSN 1445-1336

Listed below are some of the latest volumes published in the ACS Series *Conferences in Research and Practice in Information Technology*. The full text of most papers (in either PDF or Postscript format) is available at the series website <http://crpit.com>.

- |   |   |
|---|---|
| <b>Volume 67 - Conceptual Modelling 2007</b><br>Edited by John F. Roddick and Annika Hinze.<br>January, 2007. 978-1-920682-48-4.  | Proc. Fourth Asia-Pacific Conference on Conceptual Modelling (APCCM2007), Ballarat, Victoria, Australia, January 2007.  |
| <b>Volume 68 - ACSW Frontiers 2007</b><br>Edited by Ljiljana Brankovic, Paul Coddington, John F. Roddick, Chris Stekette, Jim Warren and Andrew Wendelborn. January, 2007. 978-1-920682-49-1. | Proc. ACSW Workshops - The Australasian Information Security Workshop: Privacy Enhancing Systems (AISW), the Australasian Symposium on Grid Computing and Research (AUSGRID), and the Australasian Workshop on Health Knowledge Management and Discovery (HKMD), Ballarat, Victoria, Australia, January 2007. |
| <b>Volume 69 - Safety Critical Systems and Software 2006</b><br>Edited by Tony Cant. February, 2007. 978-1-920682-50-7.   | Proc. 11th Australian Conference on Safety Critical Systems and Software, August 2006, Melbourne, Australia.  |
| <b>Volume 70 - Data Mining and Analytics 2007</b><br>Edited by Peter Christen, Paul Kennedy, Jiuyong Li, Inna Kolyshkina and Graham Williams. December, 2007. 978-1-920682-51-4.              | Proc. 6th Australasian Data Mining Conference (AusDM 2007), Gold Coast, Australia. December 2007.   |
| <b>Volume 72 - Advances in Ontologies 2006</b><br>Edited by Mehmet Orgun and Thomas Meyer. December, 2006. 978-1-920682-53-8.   | Proc. Australasian Ontology Workshop (AOW 2006), Hobart, Australia, December 2006.  |
| <b>Volume 73 - Intelligent Systems for Bioinformatics 2006</b><br>Edited by Mikael Boden and Timothy Bailey. December, 2006. 978-1-920682-54-5.   | Proc. AI 2006 Workshop on Intelligent Systems for Bioinformatics (WISB-2006), Hobart, Australia, December 2006.   |
| <b>Volume 74 - Computer Science 2008</b><br>Edited by Gillian Dobbie and Bernard Mans. January, 2008. 978-1-920682-55-2.  | Proc. 31st Australasian Computer Science Conference (ACSC2008), Wollongong, NSW, Australia, January 2008.   |
| <b>Volume 75 - Database Technologies 2008</b><br>Edited by Alan Fekete and Xuemin Lin. January, 2008. 978-1-920682-56-9.  | Proc. 19th Australasian Database Conference (ADC2008), Wollongong, NSW, Australia, January 2008.  |
| <b>Volume 76 - User Interfaces 2008</b><br>Edited by Beryl Plimmer and Gerald Weber. January, 2008. 978-1-920682-57-6.  | Proc. 9th Australasian User Interface Conference (AUIC2008), Wollongong, NSW, Australia, January 2008.  |
| <b>Volume 77 - Theory of Computing 2008</b><br>Edited by James Harland and Prabhuram Manyem. January, 2008. 978-1-920682-58-3.  | Proc. 14th Computing: The Australasian Theory Symposium (CATS2008), Wollongong, NSW, Australia, January 2008.   |
| <b>Volume 78 - Computing Education 2008</b><br>Edited by Simon and Margaret Hamilton. January, 2008. 978-1-920682-59-0.   | Proc. 10th Australasian Computing Education Conference (ACE2008), Wollongong, NSW, Australia, January 2008.   |
| <b>Volume 79 - Conceptual Modelling 2008</b><br>Edited by Annika Hinze and Markus Kirchberg. January, 2008. 978-1-920682-60-6.  | Proc. 5th Asia-Pacific Conference on Conceptual Modelling (APCCM2008), Wollongong, NSW, Australia, January 2008.  |
| <b>Volume 80 - Health Data and Knowledge Management 2008</b><br>Edited by James R. Warren, Ping Yu, John Yearwood and Jon D. Patrick. January, 2008. 978-1-920682-61-3.                       | Proc. Australasian Workshop on Health Data and Knowledge Management (HDKM 2008), Wollongong, NSW, Australia, January 2008.  |
| <b>Volume 81 - Information Security 2008</b><br>Edited by Ljiljana Brankovic and Mirka Miller. January, 2008. 978-1-920682-62-0.  | Proc. Australasian Information Security Conference (AISC 2008), Wollongong, NSW, Australia, January 2008.   |
| <b>Volume 82 - Grid Computing and e-Research</b><br>Edited by Wayne Kelly and Paul Roe. January, 2008. 978-1-920682-63-7.   | Proc. Australasian Workshop on Grid Computing and e-Research (AusGrid 2008), Wollongong, NSW, Australia, January 2008.  |
| <b>Volume 83 - Challenges in Conceptual Modelling</b><br>Edited by John Grundy, Sven Hartmann, Alberto H.F. Laender, Leszek Maciaszek and John F. Roddick. December, 2007. 978-1-920682-64-4. | Contains the tutorials, posters, panels and industrial contributions to the 26th International Conference on Conceptual Modeling - ER 2007.   |
| <b>Volume 84 - Artificial Intelligence and Data Mining 2007</b><br>Edited by Kok-Leong Ong, Wenyuan Li and Junbin Gao. December, 2007. 978-1-920682-65-1.                                     | Proc. 2nd International Workshop on Integrating AI and Data Mining (AIDM 2007), Gold Coast, Australia. December 2007.   |
| <b>Volume 86 - Safety Critical Systems and Software 2007</b><br>Edited by Tony Cant. December, 2007. 978-1-920682-67-5.   | Proc. 12th Australian Conference on Safety Critical Systems and Software, August 2006, Adelaide, Australia.   |
| <b>Volume 87 - Data Mining and Analytics 2008</b><br>Edited by John F. Roddick, Jiuyong Li, Peter Christen and Paul Kennedy. November, 2008. 978-1-920682-68-2.                               | Proc. 7th Australasian Data Mining Conference (AusDM 2008), Adelaide, Australia. December 2008.   |
| <b>Volume 90 - Advances in Ontologies</b><br>Edited by Thomas Meyer and Mehmet Orgun. September, 2008. 978-1-920682-71-2.   | Proc. Knowledge Representation Ontology Workshop (KROW 2008), Sydney, September 2008.   |